Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study

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

Abstract

Socio-technical processes have come to the forefront of recent analysis of the open source software development (OSSD) world. Interest in making these processes explicit is mounting, from industry and the software process community, as well as among those who may become contributors to OSSD organization. This paper serves to close this gap by providing an analysis of the role migration and project career advancement process, and role-sets within, that we have observed through comparative case studies within three large OSSD project organizations: Mozilla.org, Apache.org, and NetBeans.org.

1. Introduction

In recent years, organizations producing both open and closed software have sought to capitalize on the perceived benefits of open source software development (OSSD) methodologies and technologies. Recent years have seen a substantial number of full-time developers employed to contribute to, extend, customize, or provide support for the bigger named OSSD projects. Whether a full-time contributor, an occasional hobbyist, or an end-user eager to report a bug, people new to an OSS project face the same challenge: learning how to participate in the project. Although most developers do not typically join a project with the intent to lead it, understanding the joining and participation processes can be as large an entry-barrier as the technical contribution itself.

Studies of OSSD processes are increasing in number (e.g., [26, 27]), while OSSD organizational structures, technical roles, and career opportunities are much less studied. Ye and Kishida [31] and also Crowston and Howison [7] observe that OSSD project members gravitate towards central roles over time, which they depict with "onion" diagrams such as in Figure 1.

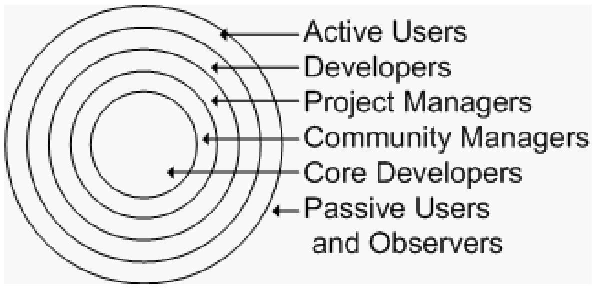


Figure 1. An "onion" diagram representing a generic OSSD project organizational hierarchy

Precedent for this sort of layered organizational depiction is well established outside of OSSD, such as in management, economics, and software engineering literature [22, 2, 8]. These diagrams indicate multi-layer organizational hierarchies across communities, but do not indicate how participants might transition between layers, or what roles are available at each layer. Moreover, the onion model as presented fails to draw out the presence of multiple tracks of project career advancement through different role-sets, suggested in Figure 2. Much like their development processes, OSSD communities typically provide little insight into role migration and advancement processes.

What guidance is provided is often directed at recruitment- initial steps to get people in the door. Guidance for attaining more central roles is often characterized as being meritocratic, depending on the governance structure of the project [10, 25]. Nevertheless, these development roles and how developers move between them seem different from and more diverse than those offered by the traditional view of software engineering, where developers seem to be limited to roles like requirements analyst, software designer, programmer, or code tester, and where there is little/no expected movement between roles during a development project (except perhaps in small projects).

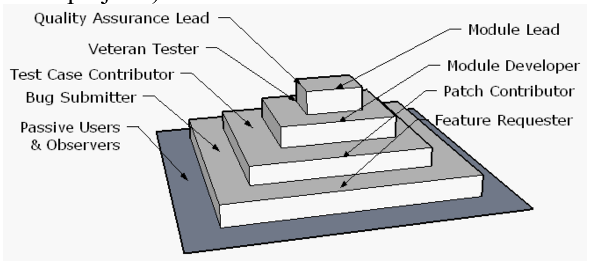


Figure 2. An "onion" pyramid representation of a generic OSSD project organizational hierarchy with multiple role-sets and advancement tracks.

Christie and Staley [4] argue that social and organizational processes, such as those associated with moving between different developer roles in a project, are important in determining the outcome of software development processes. In previous studies, we have examined software development processes within and across OSSD communities [15, 24, 25, 26]. Here, we examine two related socio-technical processes used in OSSD as a way of merging the social/cultural and technical/developmental OSSD activities. Specifically, we'll focus on the *role migration* and *project career advancement* processes of developers from end-users/observers towards roles more central roles within the Mozilla, Apache, and NetBeans OSSD project communities. Such processes characterize both the hierarchy of roles that OSS developers play (cf. [12]), as well as how developers move through or become upwardly mobile within an OSSD project (cf. [30]). While anecdotal evidence of these processes exists, the lack of precision in their description serves as a barrier to project entry, continuous process improvement, and process adoption by other organizations. A goal of our work is to provide process transparency through explicit modeling of such process in ways that enable increased participation, process improvement, and more widespread adoption.

In the remaining sections, we outline details about these role migration and advancement processes found, while analyzing cases within and across each of these three OSSD project communities.

2. Background

Role migration and career advancement has previously been studied in both education and management literature. In the former, it falls under the category of tenure. The latter brings us a wealth of topics from gender specific issues, such as glass ceilings in the workplace [18] to promotion and tenure in academia [23]. Management literature has long argued over the dual ladder system of role advancement within corporations [1]. However, there may only be one technical track of advancement available. In such cases, technical people can be forced into managerial positions in order to advance in the organization, denoting a migration in career tracks.

Here, we show that the two ladder system does not hold for role migration processes in OSS development organizations. Rather, we see several patterns at work dictated by the organizational configuration of the community. Moreover, although the traditional assumption of two ladder theory follows that individuals move up the ladder towards to positions of greater authority open source project participants ascend and descend the organizational hierarchy, but it is not uncommon to see lateral movements to other tracks of community involvement. Moreover, participation in OSSD projects is volunteer-driven and advancement is usually meritocratic: participants advance by proving themselves in the responsibilities of the position, while proving the social commitment to project success by facilitating others OSSD work, and by mediating others conflicts [27, 28]. There is a growing body of work in modeling OSS processes, and enumerating roles in such projects, however, the literature is lacking in detail of how advancement in rank is achieved in OSS development, and how participation changes over time through role migration and advancement.

3. Research approach and methods

To help explore the preceding themes, we present findings from empirical studies of cases within three large OSSD project that we have investigated longitudinally starting in 2002.We do not claim that what we report here is meant to describe all/most OSSD projects, since they probably don't. They do however represent case studies from three of the largest OSSD projects whose software products constitute a core software infrastructure for the World Wide Web. The goal of our effort is to comparatively examine and explicate semi-structured models of interesting socio-technical processes found in OSSD projects that appear to be key to overall project success and longevity. These processes and the roles people play within them guide and coordinate large-scale OSSD activities without a traditional software project management regime. As such, we have engaged a variety of qualitative and ethnographic research methods (including participant interviews, collection and cross-coding of OSSD artifacts, semi-automated Web site data mining, and multi-mode modeling (cf. [24, 28, 29, 30]) to discover and analyze role migration and advancement processes in this sample of OSSD projects, as well as the set of roles through which participants advance. Our focus here is to reveal a sample of (otherwise invisible) processes we have found through our studies with these methods, since these projects do not provide explicit descriptions or models for how such processes operate. However, we note that the processes we describe are not static, and so they evolve and adapt over time, much like most effective work processes supported by evolving computing technologies.

4. Case 1: Role-sets, role migration and advancement process in Mozilla.org

Developer recruitment in Mozilla has always been difficult. The opening of the Netscape Web browser source code a few years ago offered OSS developers a unique opportunity to peek under the hood of the once dominant Web browser in use. Nevertheless, the large scale of this software application (millions of lines of source code) and the complex/convoluted architecture scared many developers away. These factors, combined with the lack of a working release or support from Netscape led one project manager to quit early on [19]. However, with the eventual release of a working product, the Mozilla project garnered users who would later become developers to further the cause.

The Mozilla.org project Web site lists several ways for potential developers and non-technical people to get involved with the community [13]. One focuses on quality assurance and documentation reflects a community focus on maturing, synchronizing, and stabilizing updates to the source code base. Technical membership roles and responsibilities currently listed include bug reporting, screening, confirming, and fixing, writing documentation, and contacting Web sites that do not display properly within Mozilla/Firefox browsers. Compared to more central roles, these activities do not require deep knowledge of the Mozilla source code or system architecture, and serve to allow would-be contributors to get involved and participate in the overall software development process.

When bugs are submitted to the Bugzilla (the bug reporting system used to support Mozilla development), they are initially assigned to a component, which developers look at. On occasion, community members will submit patches for outstanding issues within the bug repository (often attached to comments within the defect discussion thread) if module developers have not taken action. This phenomenon can be seen especially in instances where community members wish to share solutions that have been rejected by the committers or module owner for inclusion in the source tree.

The next task is to recruit others to accept the software repair/modification (i.e., patch) and incorporate it into the source tree. Recruitment of patch review is best achieved through emailing reviewers working on the module for which the patch was committed or reaching out to the community via the Mozilla IRC chat. By repeatedly demonstrating competency and dedication writing useful code within a section of the source, would-be developers gain a reputation among those with commit access to the current source code build tree. Eventually, these committers recommend that the promising developer be granted access by people who direct or coordinate overall project activities. These are called the project drivers. In rare cases, such a developer may even be offered to become a module owner if s/he is the primary developer of that module and it has not been blocked for inclusion into the trunk of the source tree (see https://bugzilla.mozilla.org/show_bug.cgi?id=18574).

Once a project contributor is approved as a source code contributor, there are several roles available to community members. Most of these are positions requiring greater seniority or record of demonstrated accomplishments within the community. As module developers and owners establish themselves as prominent community members, other opportunities may open up. In meritocratic fashion, developers may transition from being a QA module contact to a QA owner (cf. [10]). Similar occasions exist on the project level for becoming a module source reviewer.

Super-reviewers attain rank by demonstrating superior expertise for discerning quality and effect of a given section of source on the remainder of the source tree. If a reviewer believes that s/he has done this appropriately, s/he must convince an existing super-reviewer of such an accomplishment. This super-reviewer will propose the candidate to the remainder of the super-reviewers. Upon group consensus, the higher rank is bestowed on the reviewer [20].

Project drivers are usually either contributing company employees or module owners who are interested in setting the technical direction of the project per release. Their primary role is to encourage developers to fix specific high priority or high impact bugs critical to a release. In that sense, they act as release managers in a similar capacity to those of the NetBeans.org project described later. At the time of this writing, there are seventeen drivers listed on the Mozilla project management information page (see http://www.mozilla.org/about/drivers), though the recent development branch (version 1.8.1) is directed by five drivers (see http://developer.mozilla.org/devnews/index.php/2006/06/20/181-branch-now-under-branch-driver-control).

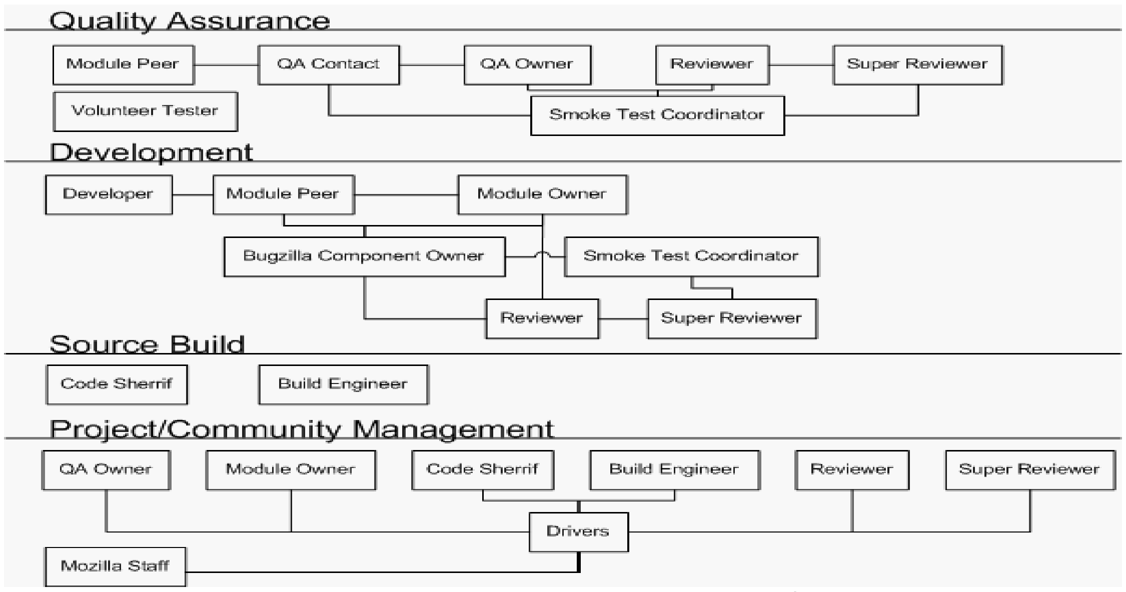


Figure 3. Role-sets, role hierarchies, and advancement paths (left-to-right) in Mozilla.org

There is little evidence to offer generalizations about how one becomes a driver, however what is visible from the calendar sub-project shows a developer informally nominated at a project status meeting by other project leads and developers (see http://wiki.mozilla.org/Calendar:Status_Meetings:2006-07-26_MeetingLog). This individual agreed to serve and was affirmed by the others present online. The change in semantics for developer roles (i.e. project lead developer) appears characteristic to the sub-project, rather than extending to flagship projects such as Firefox and Thunderbird.

Community level roles include smoke-test coordinator, code sheriff, and build engineer, although no process is prescribed for such transitions. As individual roles, they are held until vacated, at which time, the position is filled by appointment from the senior community members and Mozilla Foundation staff.

The participation tracks and respective hierarchy are given in Figure 3. This is how the Mozilla project worked for some period of time. It appears that notions of module ownership and a formal quality assurance process have diminished in recent years. The quality assurance track, in particular, appears to have collapsed into three roles: the defect submitter, the benevolent committer, who may take corrective and administrative actions on the defect report, and the patch contributor sharing his/her solution with others (which may or may not be included in the source tree). This is not to suggest that quality assurance is not performed, but rather to the contrary, that the review process and the associated role structure, have become subsumed by other processes and roles over time. When Mozilla was connected with Netscape, the super reviewership process was created in order to balance contributions from Netscape developers and the public community and ensure their quality. As Mozilla matured (and later separated from Netscape), the number of expert developers increased. For reasons as technical as social/political (as suggested by [6]), several developers split off to create Firefox with more lightweight development processes and a small, though heavily gated, role hierarchy. With Firefox becoming the dominant browser project, code reviewing and many of the related quality assurance tasks have merged into module development processes. Similarly, the reviewer and smoke test coordinator roles in the development track have similarly faded, along with code sheriff in the build and project management tracks. Moreover, since the legal incorporation of the Mozilla Foundation in 2005, build and release roles are assigned to Foundation employees. Incorporation further added a management layer including the board of directors and standing committees. The (now deprecated) role migration process for reviewers in the Mozilla.org community appears in Figure 4.

5. Case 2: Role-sets, role migration and advancement process in Apache.org

The Apache Software Foundation (ASF) has been established to nurture and nominally coordinate a multi-project software ecosystem that surrounds the Apache Web server effort. ASF has laid out a linear (and meritocratic) path for involvement, as shown by the role hierarchy in Figure 5. Individuals start out as end-users (e.g., Web site administrators), and then proceed to developer status, committer status, project

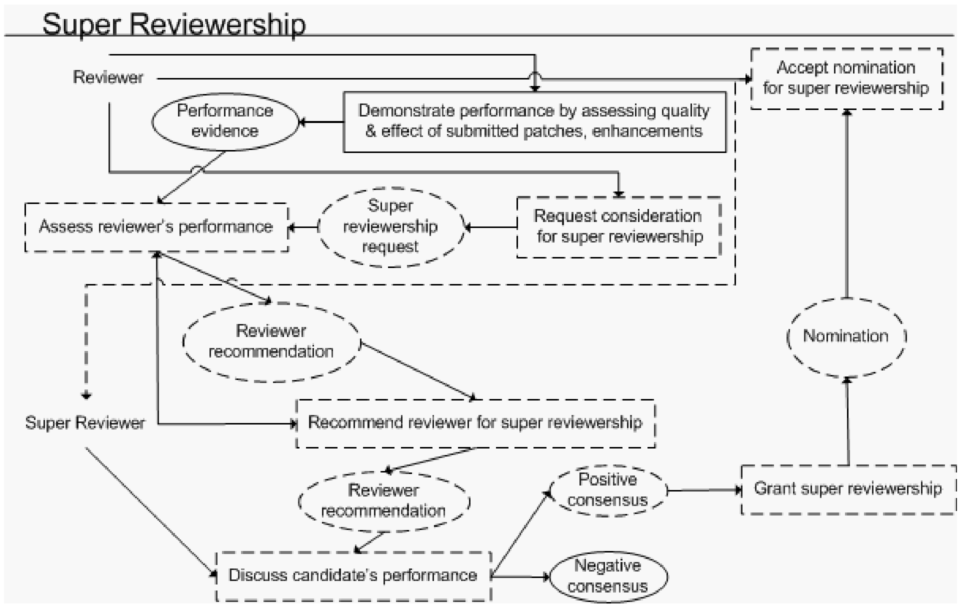

Super Reviewership
Reviewer
Accept nomination for super reviewership
Performance evidence
Demonstrate performance by assessing quality & effect of submitted patches, enhancements
Assess reviewer's performance
Super reviewership request
Request consideration for super reviewership
Reviewer recommendation
Nomination
Super Reviewer
Recommend reviewer for super reviewership
Reviewer recommendation
Positive consensus
Grant super reviewership
Discuss candidate's performance
Negative consensus

Figure 4. Deprecated role migration process for reviewers in Mozilla.org

management committee (PMC) status, ASF membership, and lastly, ASF board of directors' membership [14]. Much as in advancement in the Mozilla community, Apache membership is by invitation only. As the name suggests, the Apache server is comprised of patches submitted by developers. These patches are reviewed by committers and either rejected or accepted into the source tree.

In addition to feature patches, developers are also encouraged to submit defect reports, project documentation, and participate on the developer mailing lists. When the PMC is satisfied with the developer's contributions, they may elect to extend an offer of "committership" to the developer, granting him/her write access to the source tree. To accept committership, the developer must submit a contributor license agreement, granting the ASF license to the intellectual property conveyed in the committed software artifacts.

PMC membership is granted by the ASF. To become a PMC member, the developer/committer must be nominated by an existing ASF member and accepted by a majority vote of the ASF membership participating in the election [11]. Developers and committers nominated to become PMC members have demonstrated commitment to the project, good judgment in their contributions to the source tree, and capability in collaborating with other developers on the project. The PMC is responsible for the management of each project within the Apache community. The chair of the PMC is an ASF member elected by his/her fellow ASF members who initially organizes the day-to-day management infrastructure for each project, and is ultimately responsible for the project thereafter. ASF membership follows the same process as PMC membership- nomination and election by a majority vote of existing ASF members.

ASF members may run for office on the ASF board of directors, as outlined by the ASF bylaws [3]. Accordingly, the offices of chairman, vice chairman, president, vice president, treasurer (and assistant), and secretary (and assistant) are elected annually. A flow graph of the role migration process appears in Figure 6.

Although, there is one path of advancement in the Apache community, there are several less formal committees that exist on a community (as opposed to project) scale. These include the conference organizing committee, the security committee, the public relations committee, the Java Community Process (JCP) committee, and the licensing committee. Participation in these committees is open to all committers (and higher ranked members) and roles are formalized on an as-needed basis (e.g. conference organization). Non-committers may apply for inclusion in specific discussion lists by sending an email to the board-mailing alias explaining why access should be granted. Thus, processes associated with these committees are ad hoc and consist of one step.

6. Case 3: Role-sets, role migration and advancement processes in NetBeans.org

Roles in the NetBeans.org community for developing the Java-based NetBeans interactive development environment are observable on six levels of project management [21], as demonstrated by the role-sets and hierarchy in Figure 7.

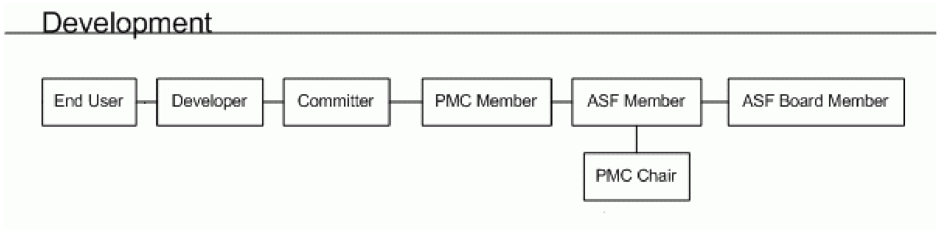


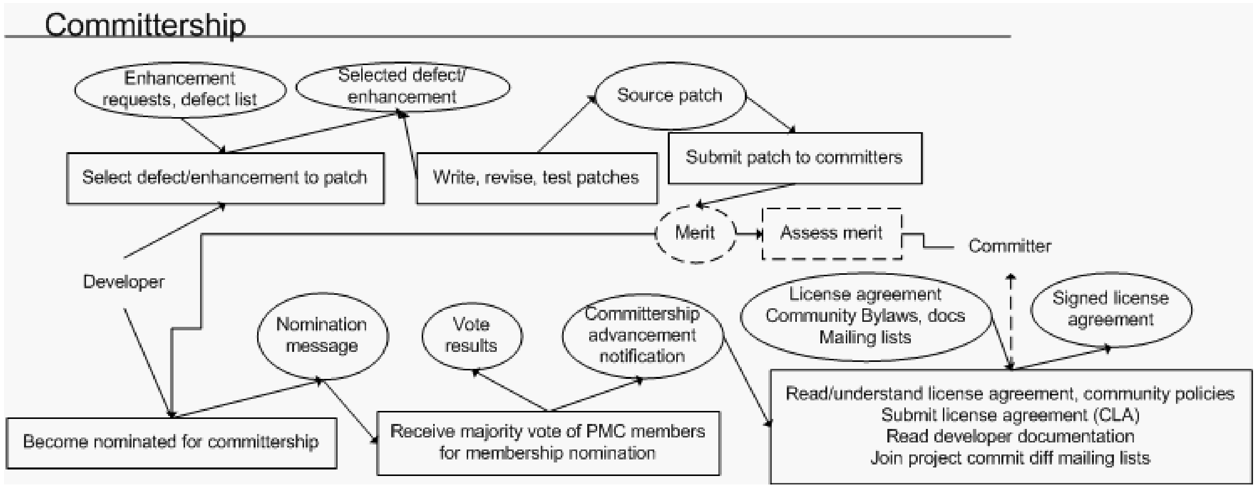
Figure 5. Role-set, role hierarchy, and advancement paths in Apache.org

These range from users to source contributors, module-level managers, project-level managers, and community-level managers. The NetBeans community's core members are mostly Sun Microsystems employees, the community's primary sponsor, and are subject to the responsibilities set on them by their internal organizational hierarchy. As such, (and unlike the cases of Apache and Mozilla), not all roles are open to volunteer and third-party contributors. Non-Sun employed community members wanting to participate beyond end-usage are advised to start out with activities such as quality assurance (QA), internationalization, submitting patches, and documentation [5]. As in the case with Mozilla, until they have proven themselves as responsible, useful, and dedicated contributors, developers must submit their contributions to developer mailing lists and the issue repository, relying on others with access to commit the source. However, unlike Mozilla, developers are also encouraged to start new modules.

While the community was more liberal with module creation early in the project's history, as the community has matured, additions to the module catalog have become more managed to eliminate an abundance of abandoned modules. Also as in Apache and Mozilla, developers are subjected to the proving themselves before being granted committer status on a portion of the source tree. Additionally, they may gain module owner status be creating a module or taking over ownership of an abandoned module that they have been the primary committer for. With module ownership comes the responsibility to petition the CVS manager to grant commit access to the source tree to developers working on the module, thereby raising their role status to "committer."

Rising up to the project-level roles, the Sun-appointed CVS source code repository manager is responsible for maintaining the integrity of the source tree, as well as granting and removing developer access permissions. In contrast, the release manger's role is to coordinate efforts of module owners to plan and achieve timely release of the software system. Theoretically, any community member may step in at any time and attempt to organize a release. In practice, this rarely occurs. Instead, most community members passively accept the roadmap devised by Sun's NetBeans team. In the latter case, the previous release manager puts out a call to the community to solicit volunteers for the position for the upcoming cycle. Assuming there are no objections, the (usually veteran) community member's candidacy is accepted and the CVS manager prepares the source tree and provides the new release manager permissions accordingly. Alternatively, a member of Sun may appoint a member of their development team to head up the release of their next development milestone.

At the community-management level, the community managers coordinate efforts between developers and ensure that issues brought up on



Committership
Enhancement requests, defect list
Selected defect/ enhancement
Source patch
Select defect/enhancement to patch
Write, revise, test patches
Submit patch to committers
Merit
Assess merit
Committer
Developer
Nomination message
Vote results
Committership advancement notification
License agreement Community Bylaws, docs Mailing lists
Signed license agreement
Become nominated for committership
Receive majority vote of PMC members for membership nomination
Read/understand license agreement, community policies
Submit license agreement (CLA)
Read developer documentation
Join project commit diff mailing lists

Figure 6. Role migration process for committership in the Apache.org community, highlighting the sequence of a developer becoming a committer.

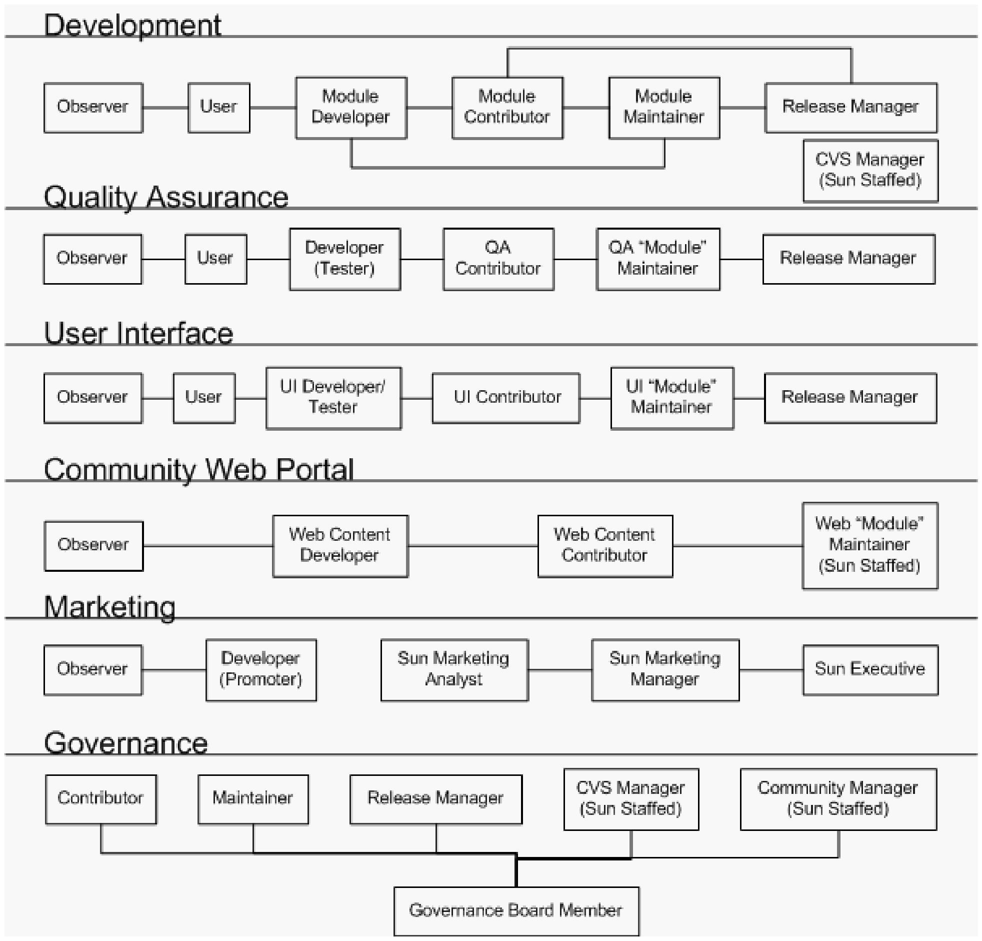


Figure 7. Role-sets, role hierarchies, and advancement paths in NetBeans.org

mailing lists are addressed fairly. At the inception of the NetBeans project, an employee of CollabNet (the company hosting the NetBeans Web portal) originally acted as community manager and liaison between CollabNet and NetBeans. However, it was soon transferred to a carefully selected Sun employee (by Sun) who has held it since. As community members have risen to more central positions in the NetBeans community, they tend to act similarly, facilitating and mediating mailing list discussions of a technical nature, as well as initiating and participating in discussions of project and community direction.

Lastly, a committee of three community members, whose largely untested responsibility is to ensure fairness within the community, governs the NetBeans project. One of the three is appointed by Sun. The community at large elects the other two members of the governance board. These elections are held every six months, beginning with a call for nominations by the community management. Those nominees that accept their nomination are compiled into a final list of candidates to be voted on by the community. A model of the product development track role migration process is shown in Figure 8.

7. Comparative case analysis

Role migration and project advancement in most OSS projects is usually passive and does not extend beyond a project's own Web site. In the projects observed, recruitment consisted of multiple ways for users and observers to get involved. Such activities include submitting defect reports, test cases, source code and so forth. These activities require a low degree of interaction with other community members, most notably decision makers at the top of the organizational hierarchy. Our observation has been that the impact of contributions trickles up the organizational hierarchy whereas socio-technical direction decisions are passed down. As such, activities that demonstrate capability in a current role, while also coordinating information between upstream and downstream (with respect to the organizational hierarchy) from a given developer are likely to demonstrate community member capability at his/her current role, and therefore good candidates for additional responsibilities.

Several themes are present in the role migration processes of these three projects. In the communities we have examined, we found different paths (or

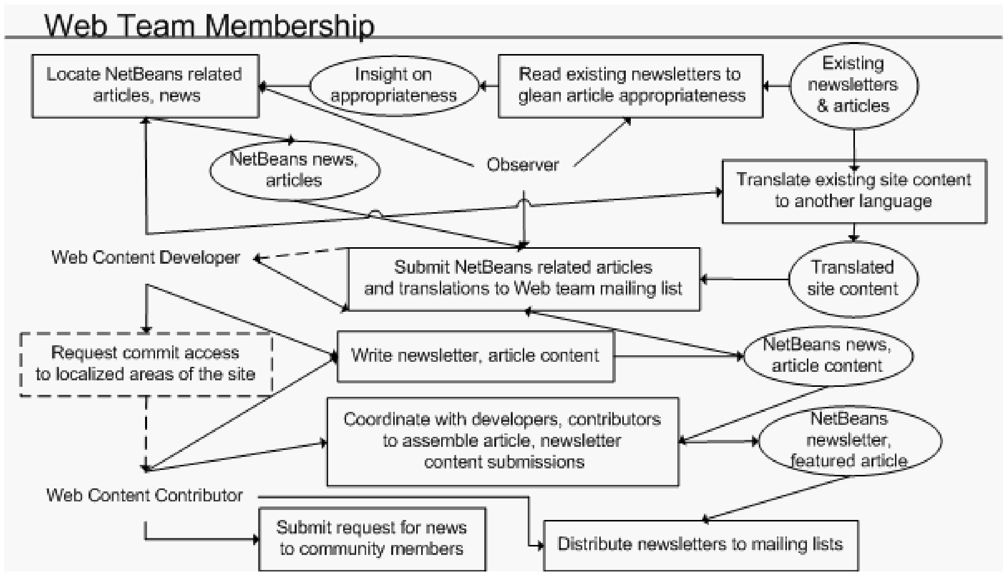

Web Team Membership
Locate NetBeans related articles, news
Insight on appropriateness
Read existing newsletters to glean article appropriateness
Existing newsletters & articles
NetBeans news, articles
Observer
Translate existing site content to another language
Web Content Developer
Submit NetBeans related articles and translations to Web team mailing list
Translated site content
Request commit access to localized areas of the site
Write newsletter, article content
NetBeans news, article content
Coordinate with developers, contributors to assemble article, newsletter content submissions
NetBeans newsletter, featured article
Web Content Contributor
Submit request for news to community members
Distribute newsletters to mailing lists

Figure 8. Role migration process for Web team membership in NetBeans.org

tracks) towards the center of the developer role hierarchy as per the focus of each path. Paths we have identified include project management (authority over technical issues) and organizational management (authority over social/infrastructural issues). Within these paths, we see tracks that reflect the different foci in their software processes. These include quality assurance roles, source code creation roles, and source code versioning roles (e.g. CVS manager, CVS committer, etc), as well as role paths for usability, marketing, and licensing. There are roles for upstream development activities (project planning). More senior members of the community generally take these up. This is due in part that developers working in these roles can have an impact on the system development commensurate with the consequences/costs of failure, and requires demonstrated skills to ensure the agents responsible will not put the software source code into a state of disarray.

The presence of corporate and non-profit organizations as the core of OSS organizations, employing project members in a full time fashion has become common. As a salient example: the Mozilla project began as a proprietary company product. For several years after the source was opened, Mozilla existed as a non-profit foundation that now recently has incorporated (though still with a non-profit status). With a full time staff, many of the project management roles (e.g. build engineering) are not positions available to community members from the general public, but rather staffed positions. The same is true of NetBeans. Role migration and advancement processes for these positions follow from the corporate or staffed organization. These processes are consequently more opaque than those of community positions. In terms of recruitment, organizations often extend employment offers to veteran community members. Moreover, though Mozilla, Apache, and NetBeans all have well defined organizational structures, we also note the existence a number of unofficial roles, such as for conference organization in Apache. These roles are emergent and their migration processes may be one-off in nature. Their existence may be temporary (e.g. conference organizing) but may formalize over time. Table 1 summarizes the types of role migration processes we have observed.

Recruitment and role migration processes are not a newly observed phenomenon. Like career paths described in management literature [17], movement in the organizational structure may be vertical or horizontal. Most large OSSD project communities are hierarchical, even if they consist of only a few layers with many members exist at each layer. It is also common for project members to wear multiple hats, so to speak. Source code developers almost unfailingly submit defect reports though their primary focus is creating new code, rather than managing test cases, test suites, or defect repository as they are the focus of the higher levels of the quality assurance that we have discussed here. Quality assurance and development are two tracks that naturally align themselves well. With further empirical study, we may be able to identify additional patterns of alignment.

In comparison to traditional software development organizations, tracks of advancement in open source communities are much more fluid. A developer contributing primarily to source code generation may easily contribute usability or quality assurance test cases and results to their respective community teams. These is not to suggest that a module manager of a branch of source code will automatically and immediately gain core developer

Table 1. Types of role migration processes observed in OSSD projects

Role Acquisition Method	*Description*
Implicit	Acquired by performing a task.
Earned/Granted	An individual or body of authority grants the rank to the community member. This may require that the community member apply for the position, or that he or she is nominated or sponsored by a higher ranking member, possibly involving a vote from the granting body.
Elected	An individual is voted into a position by the community at large or a subcommittee.
Appointed/Assigned	An individual or body of authority appoints the community member to a position.

privileges, responsibilities, and respect from those teams. However, industrial environments tend towards rigid and static organizational hierarchies with highly controlled growth at each layer.

The depiction of role hierarchies in OSSD project communities as concentric, onion-like circles speaks to the fact that those in the outer periphery have less direct control or knowledge of the project's current state and its social and technical direction compared to those in the inner core circle. As observed with Mozilla, organizational hierarchies and recruitment and role migration processes are not static. Unlike their industrial counterparts, OSSD organizational hierarchies tend towards a higher degree of agility. Although changes in the number of layers stabilize early in community formation, the size of each layer (especially outer layers) is highly variable. Evolution of the organizational structure may cause or be caused by changes in leadership, control, conflict negotiation, and collaboration in the community, such as those examined elsewhere [16]. If too pronounced, changes can lead to breakdowns of the technical processes.

Overall, meritocratic role migration and advancement processes, such as presented here, consist of a sequence of establishing a record of contribution in technical processes in collaboration with other community members, followed by certain "rights of passage" specific to each community. For Apache, there is a formal voting process that precedes advancement. In Mozilla and NetBeans, these are less formal. The candidate petitions the appropriate authorities for advancement or otherwise volunteers to accept responsibility for an activity. These authorities will either accept or deny the inquiry.

8. Conclusions

Social or organizational processes that affect the performance of software development processes have had comparatively little investigation. This is partially because some of these processes are perceived to be well understood (e.g., project management processes like scheduling or staffing), while others are often treated as "one-off" or *ad hoc* in nature, executing in different ways in each instantiation. The purpose of our comparative case study examination role migration and project career advancement processes is to help reveal how these socio-technical processes are intertwined with conventional software development processes, and thus constrain or enable how OSSD is performed in practice. In particular, we have examined and modeled these processes within a comparative sample of three large OSSD projects that embed the Web software infrastructure. As a result, we were able to identify different types of methods that OSSD projects employ to migrate and advance their participants, from peripheral roles to core leadership positions. Lastly, we have shown where and how they interact with existing software development processes found in our project sample.

Overall, we have found that comparative studies of socio-technical processes found with even a small sample of cases, such studies do in fact provide sufficient substance and detail to reveal the richness of processes, practices, and roles that shape open source software development projects.

9. References

[1] Allen, T.J., & Katz, R. (1985). The Dual Ladder: Motivational Solution or Managerial Delusion? (Tech. Rep. WP1692-85). Massachusetts Institute of Technology (MIT), Sloan School of Management.

[2] Baldwin R.E. (2001). The Core-Periphery Model with Forward-looking Expectations. *Regional Science and Urban Economics*, 31(1), 21-49.

[3] Bylaws of the Apache Software Foundation, Retrieved February 7, 2005 from http://www.apache.org/foundation/bylaws.html

[4] Christie, A., & Staley, M. (2000). Organizational and Social Simulation of a Software Requirements Development Process. *Software Process--Improvement and Practice,* 5, 103–110

[5] *Contributing to the NetBeans Project.* Retrieved February 7, 2005 from http://www.netbeans.org/community/contribute/

[6] Coward, A. (2005). About Firefox and Mozilla. *Comment on Slashdot.org forum: Firefox Developer on Recruitment Policy,* Retrieved January 31, 2005 from http://developers.slashdot.org/comments.pl?sid=137815&threshold=1&commentsort=0&tid=154&tid=8&mode=thread&cid=11527647

[7] Crowston, K. & Howison, J. (2005). The Social Structure of Free and Open Source Software Development, *First Monday*, 10(2). February. Retrieved 27 April 2006 http://www.firstmonday.org/issues/issue10_2/crowston/index.html

[8] Curtis, B., Krasner, H., & Iscoe, N. (1988). A Field Study of the Software Design Process for Large System. *Communications of the ACM,* 31(11), 1268–1287.

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

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

[11] Fielding, R., Hann, I-H., Roberts, J., & Slaughter, S. (2002). *Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project*, Paper presented at the Conference on Open Source: Economics, Law, and Policy, Toulouse, France.

[12] Gacek, C., & Arief, B. (2004). The Many Meanings of Open Source, *IEEE Software*, 21(1), 34-40.

[13] Getting Involved with Mozilla.org. Retrieved 3 Nov 2004, http://www.mozilla.org/contribute

[14] How the ASF works, Retrieved 7 Feb. 2005 http://www.apache.org/foundation/how-it-works.html

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

[16] Jensen, C., & Scacchi, W. (2005b). *Collaboration, Leadership, Control, and Conflict Negotiation Processes in the NetBeans.org Open Source Software Development Community*. Proc. 38th. Hawaii Intern, Conf. Systems Science, Waikola Village, HI, January 2005.

[17] Lash, P.B., & Sein, M.K. (1995). Career Paths in a Changing IS Environment: A Theoretical Perspective, *Proceedings of SIGCPR 1995*, 117-130. Nashville, TN

[18] Liff, S., & Ward, K. (2001). Distorted views through the glass ceiling, *Gender, Work, and Organization*, 8(1), 9-36.

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

[20] Mozilla Code Review FAQ, Retrieved 7 Feb. 2005 http://www.mozilla.org/hacking/code-review-faq.html

[21] Oza, M., Nistor, E., Hu, X., Jensen, C., & Scacchi, W. (2004). *A First Look at the NetBeans Requirements and Release Process*. Unpublished report, Bren School of Information and Computer Sciences, University of California, Irvine. Retrieved 28 Mar. 2006 http://www.isr.uci.edu/~cjensen/papers/FirstLookNetBeans/.

[22] Rosen, S. (1982). Authority, Control, and the Distribution of Earnings. *Bell Journal of Economics*, 13(2), 311-323.

[23] Saaty, T.L., & Ramanujam, V. (1983). An objective approach to faculty promotion and tenure by the analytic hierarchy process. *Research in Higher Education*, 18(3), 311-331.

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

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

[26] 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. New York, Springer Science+Business Media Inc.

[27] Scacchi, W., Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K. (2006). Understanding Free/Open Source Software Development Processes, *Software Process--Improvement and Practice*, 11(2), 95-105, March/April.

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

[29] Seaman, C.B., (1999). Qualitative Methods in Empirical Studies of Software Engineering, *IEEE Trans. Software Engineering*, 25(4), 557-572, July/August.

[30] Sim, S.E., & Holt, R.C., (1998). The Ramp-Up Problem in Software Projects: A Case Study of How Software Immigrants Naturalize, In *Proc. 20th Intern. Conf. Softw. Eng,* 361-370. Kyoto, Japan.

[31] Ye, Y., & Kishida, K. (2003). Towards an Understanding of the Motivation of Open Source Software Developers, In *Proc. 25th Intern. Conf. Softw. Eng.* 419-429. Portland, OR