Modeling Recruitment and Role Migration Processes in OSSD Projects

Chris Jensen and Walt Scacchi
Institute for Software Research
Bren School of Information and Computer Sciences
University of California, Irvine
Irvine, CA USA 92697-3425
{cjensen, wscacchi}@ics.uci.edu

Abstract

Socio-technical processes have come to the forefront of recent analyses of the open source software development (OSSD) world. Though there many anecdotal accounts of these processes, such narratives lack the precision of more formal modeling techniques, which are needed if these processes are going to be systematically analyzed, simulated, or re-enacted. Interest in making these processes explicit is mounting, both from the commercial side of the industry, as well as among spectators who may become contributors to OSSD organization. Thus, the work we will discuss in this paper serves to close this gap by analyzing and modeling recruitment and role transition processes across three prominent OSSD communities whose software development processes we've previously examined: Mozilla.org, the Apache community, and NetBeans.

Keywords: Project recruitment, membership, process modeling, open source, Mozilla, Apache, NetBeans

Introduction

In recent years, organizations producing both open and closed software have sought to capitalize on the perceived benefits of open source software development methodologies. This necessitates examining the culture of prominent project communities in search of ways of motivating developers. Although the ensuing studies have provided much insight into OSSD culture, missing from this picture was the process context that produced the successes being observed. Ye and Kishida (2003) and Crowston and Howison (2005) observe that community members gravitate towards central roles over time represented with "onion" diagrams such as in figure 1. These depictions indicate a similar number of layers in organizational hierarchies across communities, but do not suggest how one might transition between layers and what roles are available at each layer. Much like their development processes, OSSD communities typically provide little insight into role migration 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 community. Nevertheless, these development roles and how developers move between them seems to lie outside of 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 movement between roles (except perhaps in small projects).

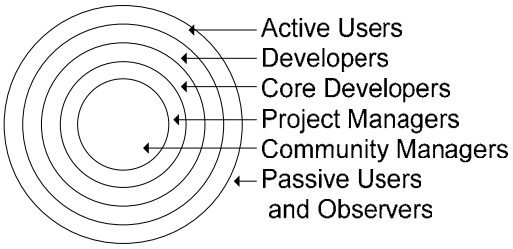

Figure 1. An "onion" diagram representation of an open source community organizational hierarchy

Christie and Staley (2000) 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 (Jensen and Scacchi, 2005, Scacchi 2002, 2004, 2005). Here, we take a look at 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 recruitment and migration of developers from end-users or infrequent contributors towards roles more central to the community, like core developer, within projects such as the Mozilla, Apache community, and NetBeans projects. Such processes characterize both the hierarchy of roles that OSS developers play (cf. Gacek and Arief 2004), as well as how developers move through or become upwardly mobile within an OSSD project (Sim and Holt 1998). While anecdotal evidence of these processes exists, the lack of precision in their description serves as a barrier to community entry,

continuous improvement, and process adoption by other organizations. The goal of our work here thus serves to provide process transparency through explicit modeling of such processes in ways that may enable increased community participation, more widespread process adoption, and process improvement.

In the remaining sections, we outline details about recruitment and role migration as membership processes as found while examining each of these three OSSD project communities. At the ProSim'05 Workshop we will present a variety of semi-structured and formal models that enable more rigorous analysis and simulated re-enactment using tools and techniques we have previously developed and employed (cf. Noll and Scacchi 2001, Jensen and Scacchi 2005)

Membership Processes in Mozilla.org

Developer recruitment in Mozilla was difficult at the start. The opening of the Netscape browser source code offered developers a unique opportunity to peek under the hood of the once most dominant Web browser in use. Nevertheless, the large scale of the application (millions of lines of source code) and the complex/convoluted architecture scared developers away. These factors, combined with the lack of a working release and the lack of support from Netscape led one project manager to quit early on (Mockus, *et. al*, 2002). However, with the eventual release of a working product, the Mozilla project garnered users who would become developers to further the cause.

The Mozilla Web site lists several ways for potential developers and non-technical people to get involved with the community (Getting Involved with Mozilla.org, 2005). The focus 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 sites that do not display properly under Mozilla. 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, they are initially assigned to a default developer for correction. It is not uncommon for community developers and would-be developers to become frustrated with an outstanding issue within the bug repository and submit a patch, themselves.

The next task is to recruit others to accept the 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 developer be granted access by the project drivers. In rare cases, such a developer may even be offered ownership of a particular module 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[1].

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 (cf. Fielding 1999), developers may transition from being a QA module contact to a QA owner. Similar occasions exist on the project level for becoming a module source reviewer.

Super-reviewers attain rank by demonstrating superior faculty 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 (Mozilla Code Review FAQ, 2005). The same follows for Mozilla drivers, who determine the technical direction of the project per release.

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. Role hierarchy and a flow graph of the migration process for transitioning from reviewer to super-reviewer are provided in figure 2 as an example of those we have modeled for this community. In the flow graph, rectangles refer to actions, whereas ovals

[1] https://bugzilla.mozilla.org/show_bug.cgi?id=18574

refer to resources created or consumed by the associated action, as determined by the direction of the arrow linking the two. Transitions from one role to another are depicted with a dashed arrow from an action performed by one role to the title of another. We have also used dashed lines to differentiate social or role transitioning activities and resources from strictly technical, developmental resources.

Membership Processes in the Apache Community

Role migration in the Apache community is linear. The Apache Software Foundation (ASF) has laid out a clear path for involvement in their meritocracy. Individuals start out as end-users (e.g., Web site administrators), proceed to developer status, then committer status, project management committee (PMC) status, ASF membership, and lastly, ASF board of directors membership (How the ASF Works, 2005). 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 accepted or rejected 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 committee 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 (Fielding, et. al, 2002). 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 (Bylaws of the Apache Software Foundation, 2005). 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 3.

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.

Membership Processes in the NetBeans.org Community

Roles in the NetBeans.org community for developing the Java-based NetBeans interactive development environment are observable on five levels of project management (Oza, et. al 2002) just as in Apache. 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 (Contributing to the NetBeans Project, 2005). 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 catalogue have become more managed to eliminate an abundance of abandoned modules. Also as in 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, 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 ensures that issues brought up on 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 4.

Discussion

In both NetBeans and Mozilla, recruitment consists of listing 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.

Recruitment and role migration processes aren't something new; since they describe the actions and transition passages involved in moving along career paths. Like career paths described in management literature (e.g., Lash and Sein 1995), movement in the organizational structure may be horizontal or vertical. Most large OSSD project communities are hierarchical, even if here are few layers to the hierarchy and many members exist at each layer.

In the communities we have examined, we found different paths (or tracks) towards the center of the developer role hierarchy as per the focus of each path. Paths we've 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--these are generally taken up by more senior members of the community. 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 require demonstrated skills to ensure the agents responsible won't put the software source code into a state of disarray).

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. This is not to suggest that a module manager of a branch of source code will automatically and immediately gain core developer 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 open source communities as concentric, onion-like circles speaks to the fact that those in the outer periphery have less direct control or knowledge of the community's current state and its social and technical direction compared to those in the inner core circle. Unlike their industrial counterparts, open source community hierarchies are dynamic. Although changes in the number of layers stabilizes early in the community formation, the size of each layer (especially the 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 (Jensen and Scacchi 2005b). If too pronounced, these changes can lead to breakdowns of the technical processes.

As a general principle, meritocratic role migration processes such as those we have observed 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. However, in the Mozilla and NetBeans communities, 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.

Conclusion

Social or organizational processes that affect or constrain the performance of software development processes have had comparatively little investigation. This is partially because some of these processes may 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 a variety of ways in each instantiation. The purpose of our examination and modeling study of recruitment and role migration processes is to help reveal how these socio-technical processes are intertwined with conventional software development processes, and thus constrain or enable how software processes are performed in practice. In particular, we have examined and modeled these processes within a sample of three OSSD projects that embed the Web information infrastructure. Lastly, we have shown where and how they interact with existing software development processes found in our project sample.

References

Bylaws of the Apache Software Foundation, available online at http://www.apache.org/foundation/bylaws.html accessed 7 February 2005

Christie, A. and Staley, M. "Organizational and Social Simulation of a Software Requirements Development Process" *Software Process Improvement and Practice* 2000; 5: 103–110 (2000)

Contributing to the NetBeans Project, available online at http://www.netbeans.org/community/contribute/ accessed 7 February 2005

Coward, Anonymous. "About Firefox and Mozilla" Comment on Slashdot.org forum "Firefox Developer on Recruitment Policy," available online at http://developers.slashdot.org/comments.pl?sid=137815&threshold=1&commentsort=0&tid=154&tid=8&mode=thread&cid=11527647, 31 January, 2005.

Crowston, K. and Howison, J. 2005. The Social Structure of Free and Open Source Software Development, First Monday, 10(2). February. Online at http://firstmonday.org/i8ssues/issue10_2/crowston/index.html

Elliott, M., The Virtual Organizational Culture of a Free Software Development Community, *Proceedings of the Third Workshop on Open Source Software*, Portland, Oregon, May 2003.

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

Fielding, R., Hann, I-H., Roberts, J and Sandra Slaughter, S. "Delayed Returns to Open Source Participation: An Empirical Analysis of the Apache HTTP Server Project," Presented at the Conference on Open Source: Economics, Law, and Policy, Toulouse, France June 2002.

Gacek, C. and Arief, B., The Many Meanings of Open Source, *IEEE Software*, 21(1), 34-40, January/February 2004.

Getting Involved with Mozilla.org, Web page available online at http://www.mozilla.org/contribute/ 3 November 2004

How the ASF works, available online at http://www.apache.org/foundation/how-it-works.html, accessed 7 February 2005

Jensen, C. and Scacchi, W., Process Modeling Across the Web Information Infrastructure, *Software Process Improvement and Practice*, to appear, 2005.

Jensen, C. and Scacchi, W. Collaboration, Leadership, Control, and Conflict Negotiation Processes in the NetBeans.org Open Source Software Development Community. working paper, Institute for Software Research, March 2005

Lash, P.B. and Sein, M.K. Career Paths in a Changing IS Environment: A Theoretical Perspective, Proc. SIGCPR 1995, 117-130. Nashville, TN

Mockus, A., Fielding, R., and Herbsleb, J. "Two Case Studies of Open Source Software Development: Apache and Mozilla," ACM Transactions on Software Engineering and Methodology, 11(3):309-346, 2002

Mozilla Code Review FAQ, available online at http://www.mozilla.org/hacking/code-review-faq.html, accessed 7 February 2005

Noll, J. and Scacchi, W., Specifying Process-Oriented Hypertext for Organizational Computing, *J. Network and Computer Applications*, 24(1), 39-61, 2001.

Oza, M. Nistor, E. Hu, X., Jensen, C., Scacchi, W. "A First Look at the NetBeans Requirements and Release Process." June 2002, updated February 2004 available online at http://www.isr.uci.edu/~cjensen/papers/FirstLookNetBeans/.

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

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

Scacchi, W., 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, 2005.

Sim, S.E. and Holt, R.C., The Ramp-Up Problem in Software Projects: A Case Study of How Software Immigrants Naturalize, *Proc. 20th Intern. Conf. Software Engineering*, Kyoto, Japan, 361-370, 1998.

Ye, Y. and Kishida, K. Towards an Understanding of the Motivation of Open Source Software Developers, *Proc. 25th Intern. Conf. Software Engineering*, Portland, OR, 419-429, 2003.

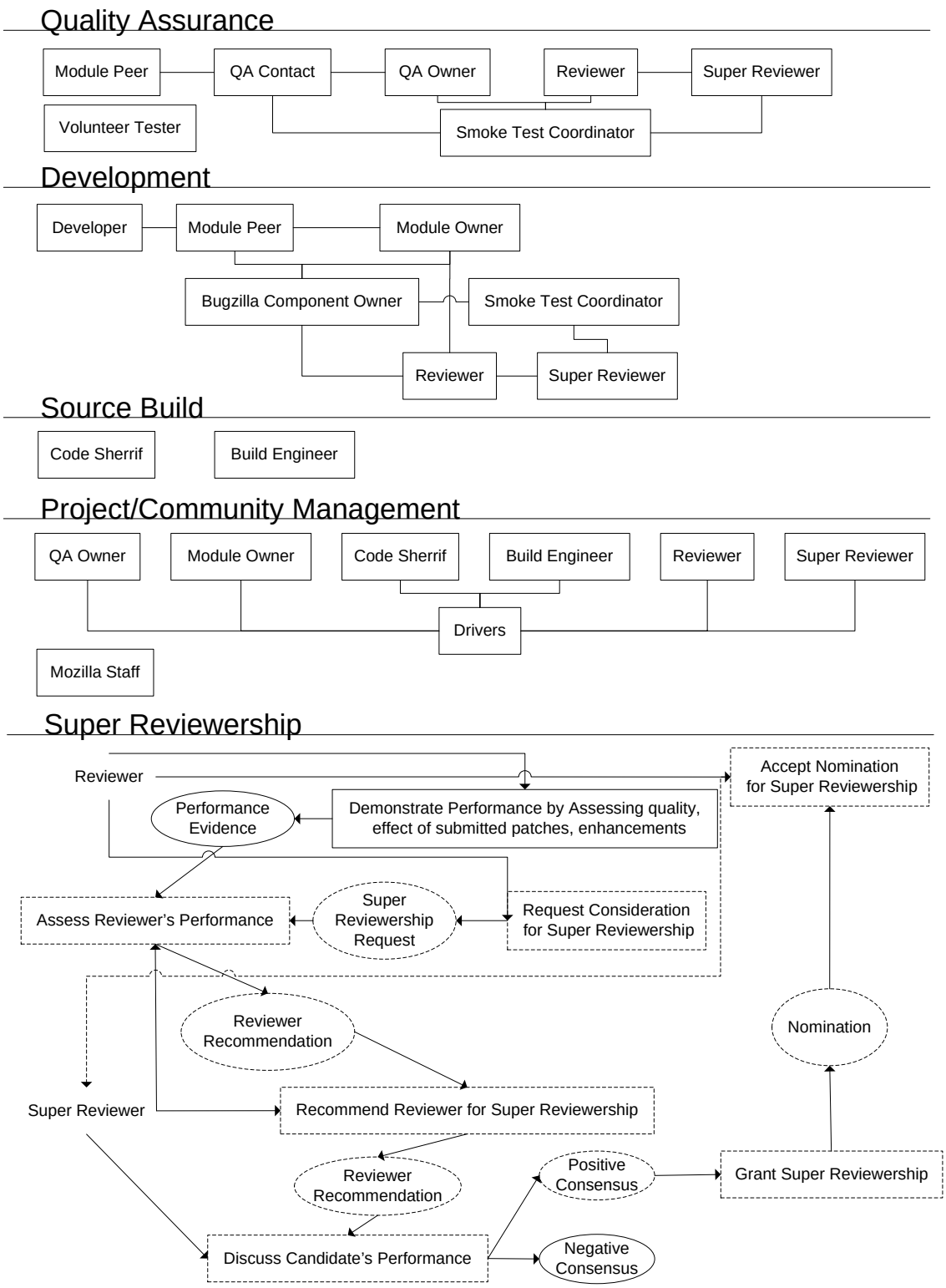

Figure 2. Role hierarchy and super reviewership migration in the Mozilla community

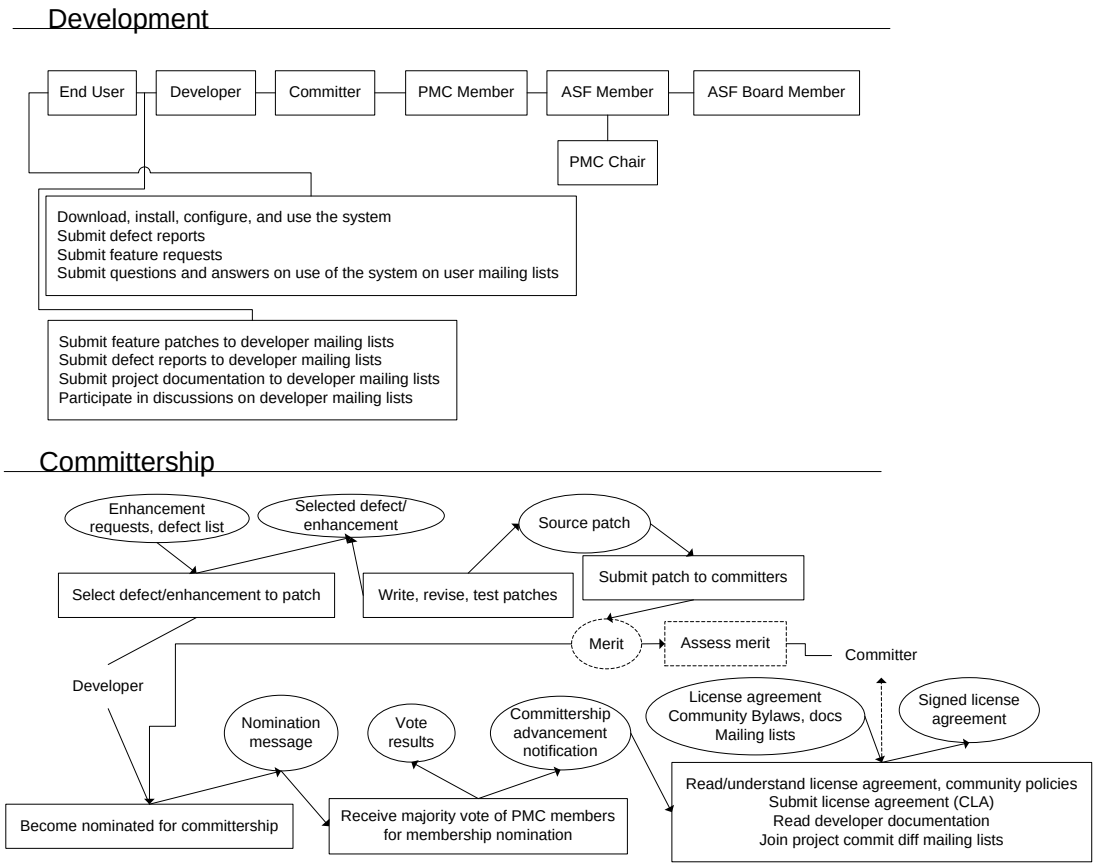

Figure 3. Role hierarchy and committership migration in the Apache community, highlighting the sequence of a developer becoming a committer

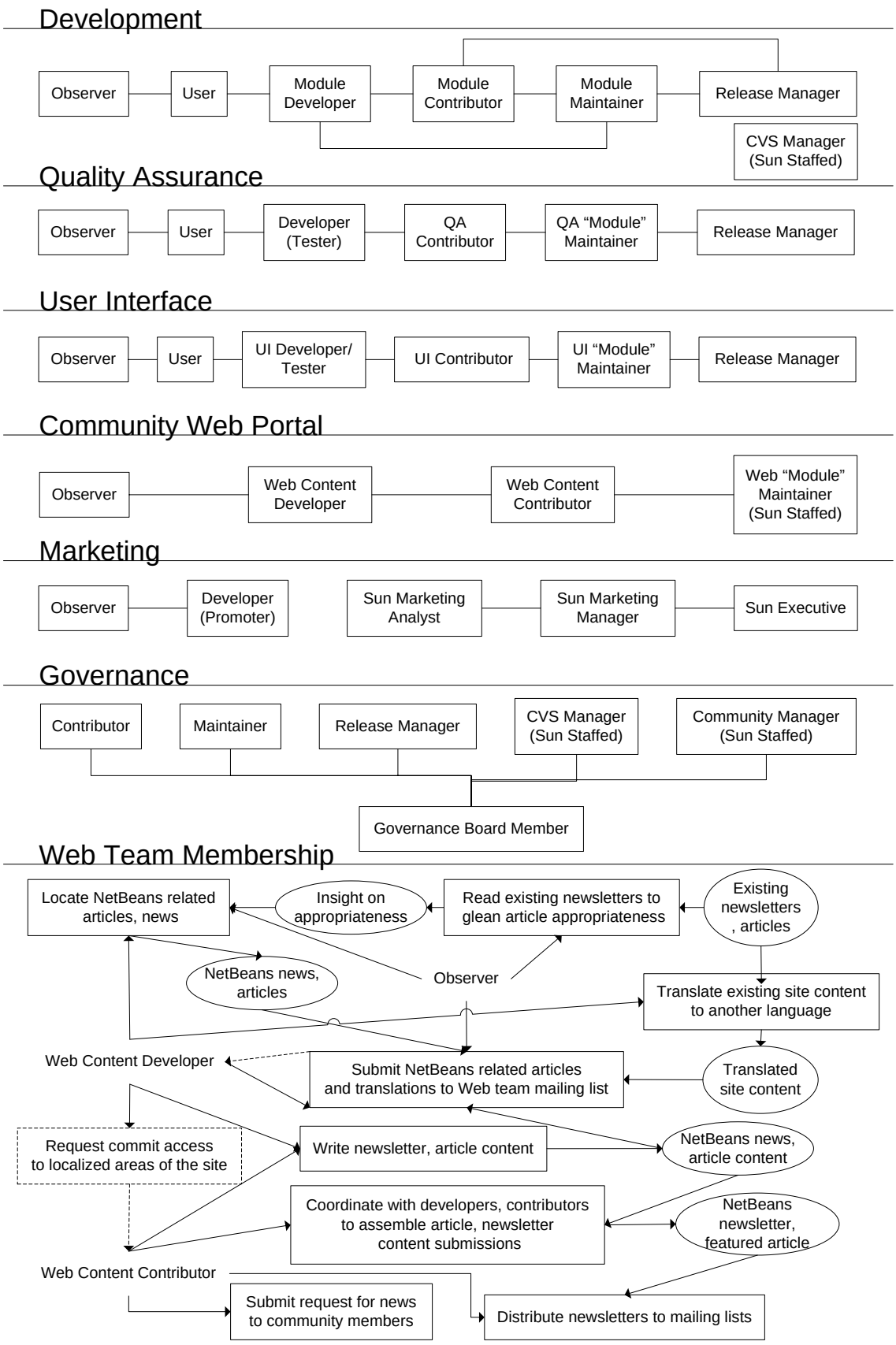

Figure 4. Role hierarchy and Web Team membership migration in the NetBeans open source community