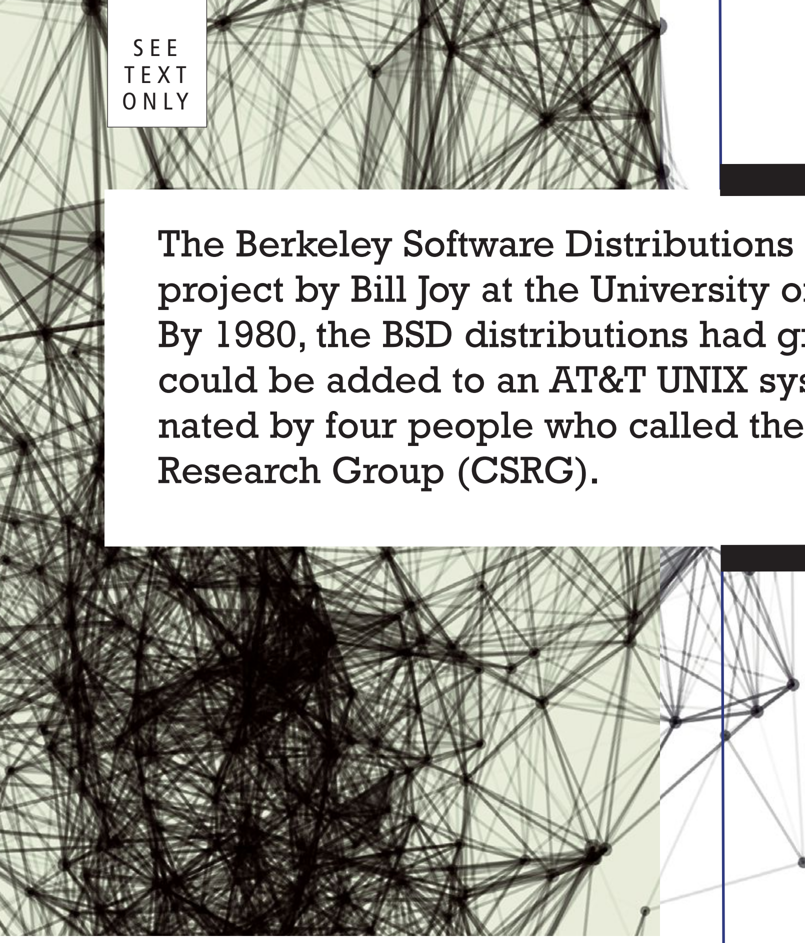

The Evolution Of FreeBSD Governance

BY MARSHALL KIRK MCKUSICK AND BENNO RICE

The Berkeley Software Distributions (BSD) was started as a one-man project by Bill Joy at the University of California at Berkeley in 1977. By 1980, the BSD distributions had grown from a few programs that could be added to an AT&T UNIX system to a complete system coordinated by four people who called themselves the Computer Systems Research Group (CSRG).

At that time, BSD development began being managed using SCCS, the first source-code control system. By 1983, the socket interface had been designed, and TCP/IP implemented underneath it allowing a small set of trusted external contributors to log into the CSRG development machines over the ARPAnet (which later became the Internet) and directly update the sources using SCCS. The CSRG staff could then use SCCS to track changes and verify them before doing distributions. This structure formed the basis for the current BSD-based projects once BSD was spun off from the university as open source.

The Formation of the FreeBSD Project

The final release from Berkeley was an open-source version of BSD called Networking Release 2 which was later rereleased as 4.4BSD-Lite. The release was missing six kernel files that still contained AT&T proprietary code. Bill Jolitz wrote replacements for these six missing files and released a system called 386BSD that ran on the commodity PC hardware. Frustrated with the slow development pace of Bill's 386BSD work, a set of developers forked 386BSD to start the FreeBSD Project.

Following the CSRG model, the FreeBSD Project used a CVS source-code repository to manage the code. The only distribution method for their early releases was agonizingly slow 14.4K dialup modems. As the FreeBSD 1.0 release approached, they were looking for a way to more quickly reach a larger audience. Jordan Hubbard approached Walnut Creek CD-ROM whose

business model was creating CD-ROM distributions of open-source software that they could then sell. The FreeBSD developers' goal of expanding FreeBSD's distribution matched well with Walnut Creek CD-ROMs business model, so Walnut Creek CD-ROM was happy to pick up FreeBSD as one of their distributions. For its part, Walnut Creek CD-ROM provided the high-powered development machines needed by the FreeBSD developers to host the CVS repository and to manage the release engineering of distributions that Walnut Creek CD-ROM sold. As the popularity of FreeBSD grew, Walnut Creek CD-ROM hired several of the FreeBSD developers to work on FreeBSD full-time.

As the use of FreeBSD expanded, so did the number of software packages included with the distributions. To keep the size of the FreeBSD distribution from exploding, the ports collection was created.

The base FreeBSD system has just the critical programs and libraries. The ports collection (which currently has over 25,000 packages) can then be used to supplement the base system with the programs needed to complete a system's functionality. So, a desktop machine installs a window manager, web browser, and a mail client from the ports collection. A machine providing a web server installs a program like Apache from ports.

Initially, everyone working on FreeBSD could commit to the CVS repository, but as the number of developers involved grew, that became untenable. With the move to Walnut Creek CD-ROM, a core team was created to make the commits and to decide who else should be able to commit to the repository. GNATS was brought up to do bug tracking.

The FreeBSD Project Moves into Companies

As FreeBSD expanded in size and needs, and became core technology at more companies, the source code repository and main development machines moved from Walnut Creek CD-ROM to Yahoo, whose entire company ran on FreeBSD. Realizing that having the project dependent on the munificence of a single company was undesirable, Justin Gibbs created the FreeBSD Foundation in 2000 in the hope that it could eventually garner enough support to fully support the project infrastructure.

It took a decade before the FreeBSD Foundation was fully able to support the project resources. Today it provides many things, including staff to head the marketing and release engineering teams; a staff person to oversee both developers receiving grants from the foundation; and other foundation staff working on projects that tackle needed parts of the system development that volunteers do not have the time to do or are not interested in doing.

Initially, the core team was permanently appointed, eventually approaching nearly 20 members. By 2000, only about a third were consistently active in the project while another third were not participating at all. This deadwood caused the business of the core team to grind almost to a halt. There was growing frustration among the committers that decisions were not being made in a timely manner and/or that the core team members acted abruptly and radically in ways that others felt bordered on impunity.

As a result, some key developers, both in and out of core, increasingly took matters into their own hands. Nobody wanted to relinquish their perceived prestigious core-title voluntarily. To gain better accountability, promote faster decision making, and to have a natural mechanism that cleared out the deadwood, a group of central developers proposed letting the developers elect the core team.

Warner Losh together with Poul-Henning Kamp, Wes Peters, and others drew up a set of bylaws that created the current structure where the core team is nominated from and elected by committers every two years. The underlying philosophy was that if core cannot be trusted, the project is doomed, but at the same time you cannot legislate common sense. Creating core by electing a bunch of random people into the role seemed unlikely to bring about project unity or even consensus on important matters. But it was agreed that democracies are the worst form of government except for all the others. So, an elected core was deemed to be the best solution.

With some arm-twisting, the original core team adopted the bylaws thus bringing in the first elected core team of nine members. Unsurprisingly, only a few of the original core members were carried over to the first elected core. The net effect was generally agreed to be that there was little change in the effectiveness of the core team. However, overall contentment of the developers improved as it was a lot harder to argue with the implicit authority of an elected body that appears to have the support of a majority of the electorate.

The core team is tasked with keeping the FreeBSD Project running. It approves new committers, resolves differences between committers, and manages committer discipline using such mechanisms as suspension of commit privileges. They also handle any other top-level issues that arise within the project.

To streamline management of various areas, the core team has created other teams or responsible people (referred to as "hats") who own certain aspects of the project. These teams include:

• The port manager team that oversees the 217 ports committers who maintain the ports tree.
• The documentation team that oversees the 126 documentation committers who develop the FreeBSD documentation and prod other committers if their documentation needs updating.
• The Security Officer, along with a team that handles security issues and oversees the release of security alerts and updates.
• The seven-member system administration team that maintains the FreeBSD infrastructure.
• The release engineering team that consists of Glen Barber and about 10 other committers who assist him with FreeBSD releases.
• The quality assurance team that runs continuous integration builds and creates an ever-growing set of regression tests. Additionally, the FreeBSD Foundation assists with advocacy and marketing. The group is headed by Anne Dickison who works with members of the FreeBSD community to provide promotion, outreach, and social networking for FreeBSD.

The FreeBSD Project Today

Project collaboration was initially handled using a single mailing list. Over time the traffic on this list grew until it became necessary to split it out into multiple lists, each focused on a given topic area such as networking, file systems, ports, documentation, announcements, and general questions. Eventually, the proliferation of mailing lists made it difficult to deal with issues that spanned several areas. Some collaboration happened via bug tracking, initially in GNATS and later in Bugzilla, but these tools were lacking when it came to reviewing larger changes, particularly if involving developers outside of FreeBSD.

In 2014 an instance of Phabricator, an open-source collaboration tool written and released by Facebook, was set up to allow detailed pre-commit review of larger changes. Phabricator facilitates detailed review and discussion of a proposed change somewhat similar to GitHub "pull requests." Phabricator has created an easier venue for non-committers to propose changes to FreeBSD as they are able to create their own Phabricator accounts, post their recommended changes, and have Phabricator automatically suggest reviewers or otherwise notify appropriate FreeBSD developers that a change needs review.

FreeBSD Source-Code Control

When the FreeBSD project began, the founders chose to use the CVS source-control system. With the release of newer source code management tools like Subversion in 2000, the pressure to move to a more modern tool began to increase. This pressure only increased with the release of the newer wave of tools such as Git and Mercurial. The branching models and commit atomicity of all of these tools were highly attractive along with the fact that all of them were generally easier to deal with than CVS. Eventually, after a lot of discussion, test conversions, and verification, the project moved to Subversion with the rationale that it was fairly close in operation to the CVS system, and that Git and Mercurial could both function on top of it if needed.

Despite the conversion to Subversion, discussion has continued about moving to something newer. There are many FreeBSD developers actively using Git, and with the advent of GitHub and its pull request model, there are ongoing discussions on whether FreeBSD should adopt GitHub or something like it. The FreeBSD Project has a presence on GitHub but it is purely read-only. Pull requests and issues opened on GitHub can only be addressed by having them moved over to Phabricator or Bugzilla before being committed into Subversion.

FreeBSD Workflow

As the project grew, more formal structures were needed to ensure smooth workflow without inhibiting innovation. Outside developers produce bug fixes and updates to FreeBSD. Using mailing lists or consulting the source-code-control logs, they identify an appropriate committer with whom to work to get their changes incorporated into FreeBSD. Another option is to create a Phabricator account to raise an issue and have the Phabricator infrastructure identify the appropriate committer or group with whom to work.

Committers (of which there are currently 371) are authorized to commit changes to specific parts of the system. These system parts are broken into three main committer groups: documentation, ports, and source. Many committers are in more than one group. Committers normally work in a self-defined subset of the groups in which they are a member. All changes (other than trivial ones) require review by at least one other committer.

Historically, committers could simply make any changes they saw fit. This policy led to broken infrastructure, especially when changes were made to systems that were more central, such as the virtual memory subsystem. To combat these problems, developers were encouraged to seek review of their changes before committing. These changes were formalized by requiring tags to be added to commit log messages indicating:

• which other project member had reviewed the changes,
• the sponsoring organization (e.g., the project member's employer),
• the bug report number from which it was identified,
• the Phabricator thread on which it was discussed,
• when to send a reminder to merge the change to older stable/release branches, and,
• if appropriate, an acknowledgement that the commit fixes an earlier mistake made by the committer (the "Pointy Hat" tag).

Guidelines on How to Work and Play Together

Though the project had long had a set of guidelines on how members should interact with each other, it did not have explicit rules and procedures to be followed when the guidelines were violated. Rules and procedures were added in response to some developer disagreements and misbehavior, and detailed clear behavioral expectations. These initial rules were primarily based around interactions within CVS.

The initial rules did not approach the clarity of behavioral expectation contained in a more modern Code of Conduct. The initial rules did provide examples of the types of sanctions that the core team could impose in response to breaches of those rules. The initial rules sufficed until 2015 when there were some serious cases of project member misbehavior that spread beyond the bounds of the project itself. This event led to efforts to augment the initial rules with a modern Code of Conduct and a set of attendant processes to deal with issues like this event in the future.

FreeBSD Recruitment

As with all successful open-source projects, developers lose interest or have insufficient time and leave the project. To avoid deadwood, it is important to have metrics to gauge when developers have left. The FreeBSD Project uses the metric of one year without doing a commit to drop an individual's commit privileges.

To keep the project viable, new developers must be recruited and brought into the project. There are several ways new contributors come into contact with the project. One is simply by discovering the project and becoming involved directly. Another is by coming into contact via a university or college course. A third is by working at a company that uses FreeBSD in its products or services. FreeBSD also takes part in the Google Summer of Code and has gained many contributions through this program.

To provide a welcoming and easily entered community, it is important to make the project visible and to provide new developers with mentors to help them learn the policies and procedures used by the project. Committers working with active developers can nominate them to be brought into the project as a new committer. Core is responsible for deciding whether to admit new committers. To ease the transition and to ensure that new committers understand the culture, procedures, and rules of the project, they are assigned a mentor (usually the person that nominated them) to review their changes and ensure that they get proper external review. Once they have gotten up to speed, typically in six to twelve months, their mentor deems them ready to work independently.

FreeBSD Development Model

The project has been quite good at identifying areas for change. When the changes are small and/or contained to a small area of the project, architecting and developing those new areas has gone smoothly. However large or highly impactful changes have been difficult. Two examples stand out: the shift from a single-threaded to a multi-threaded kernel and the move from CVS to

Subversion.

The shift away from CVS was first suggested in 1999. It was recommended that the project move to BitKeeper, which was then in public beta. This suggestion did not pan out, but it, and subsequent discussions around other tools, all fit a general pattern where someone would suggest moving to one tool, others would object and/or suggest moving to other tools, yet others would either vocally support or object to one or more of the previous suggestions, and in the end the discussion would die out with no real conclusion. In 2008, Peter Wemm, in his role as both a member of the cluster administration team and as one of the CVS repository managers, cleared the deadlock by dint of simply picking Subversion, doing all the work necessary to perform and validate the migration, and making it all happen. Subversion was chosen due to its close match to the semantics of CVS, its relative maturity, and the ability of the two other most commonly cited competitors, Git and Mercurial, to interoperate with Subversion.

Moving the kernel from single-threaded to multithreaded was a similarly large task, but, in this case, the problem was one of personalities and disagreements over architecture. Berkeley Software Design Inc. (BSDi) purchased Walnut Creek CDROM in 2000. This purchase provided developers employed by Walnut Creek access to the source code of BSD/OS, BSDi's commercial BSD derivative. One of these developers, John Baldwin, used ideas from BSD/OS and Solaris to create an architecture for a multithreaded FreeBSD kernel. Another developer, Matthew Dillon, preferred a different architecture that was similar to the approach taken in the Amiga kernel. Significant conflicts arose between these two over how the multithreading project should continue. These conflicts were exacerbated by the core team not wanting to actively pick sides in a technical debate. In the end, due to other reasons, Matthew's commit access was removed and he left the project to found the DragonFlyBSD project.

To try to avoid situations like these two, core introduced the "FreeBSD Community Process", a more formalized mechanism for proposing and deciding on important or contentious changes within the project. The idea is to avoid discussions degenerating into an interminable argument on the mailing lists with ultimately no action being taken.

The FreeBSD Community Process is modeled on similar ideas in other projects, particularly the Python Enhancement Process (https://www.python.org/dev/peps/pep-0001/), the Joyent RFD Process (https://github.com/joyent/rfd/blob/master/README.md), and even the venerable IETF RFC Process (https://www.ietf.org/about/standards-process.html).

Committers who want to make a change that will result in a nontrivial effect on the FreeBSD user base, or retrospectively, anyone having backed out a change after running into contention over something that turned out less trivial than they anticipated, writes down what they propose to change. Their proposal describes the problem they are trying to solve, outlines how they propose to solve it, and indicates any consequential impacts the proposal may have. After being vetted by a FreeBSD Community Process editor, the document is added to the FreeBSD Community Process index, committed into the FreeBSD Community Process repository, and published for discussion. Each FreeBSD Community Process proposal is a living document and can be updated to reflect any conclusions resulting during the discussion.

Once consensus has been achieved, or the discussion has gone on for enough time, the core team votes on accepting the proposal. The core team is expected to vote according to the mood of the discussion around the proposal.

FreeBSD Core Team Interaction with the FreeBSD Committers

Historically, the core agenda was private, and all communications within the core team, whether by email, IRC, or the monthly video conference, were kept private. Communication of the core team's activities to the committers was limited to a monthly report on the actions that they had taken that was prepared by the core secretary. The core secretary was not a member of core but managed core's agenda and handled many of the communications with core. The report consisted of a brief summary of the actions taken by core and discussed only actions that had already been taken. Because of the monthly report's retrospective perspective, there was little opportunity for committers to participate in core's deliberations.

After years of prodding by the committers, the core team recently began working to be more transparent and to provide an opportunity for committers to have more input to core's deliberations. The core team started providing its agenda to developers before their meetings. Core is also looking into allowing committers to attend their video-conference meetings. Attending video-conference meetings will require identifying agenda items that require core-only deliberations that may need careful handling such as disputes between developers.

FreeBSD Security Team

The role of the Security Officer has evolved over the years. Initially it was simply a title for the person tasked with looking after security-related issues for the project. In 2002, an official charter (https://www.freebsd.org/security/charter.html) was adopted that also acknowledged that there was more work than one person could handle and that there would be a Security Team that reported to the Security Officer. In the last few years it has also become apparent that finding someone with the background and time to be the Security Officer is no small feat. In response to this expansion of responsibility, the core team has recast the Security Officer role to be more of a managerial one with the Security Team acting as a pool of people who can do work as needed to address security issues, draft advisories, and ensure patches get into all the relevant branches.

Summary and Conclusions

Like many open-source projects, FreeBSD started out with the Benevolent Dictator(s) For Life governance model. While this approach can be effective, it often leads to stagnation and an aging out and/or burning out of those in charge. For these reasons, FreeBSD moved first to a core team and later to an elected core team.

In its 24-year history there have been four major changes in leadership, each of which has been beneficial and allowed the project to move forward and tackle new problems. It has also helped the project avoid aging out; the median age of the committers has consistently remained in the mid to high 30s. Young enough to have the time and energy to push projects forward, but old enough to have the necessary wisdom to avoid rat holes, and with the patience and experience to avoid technical debt by doing things right rather than settling for a quick hack.

The creation of the FreeBSD Foundation has also been important in ensuring that the FreeBSD Project has the resources that it needs to be successful. Importantly, the FreeBSD Foundation has recognized that its role in the FreeBSD ecosystem is to let the core team and the committers determine the technical direction of the FreeBSD Project while supporting it with infrastructure, marketing, and outreach.

Governance is mundane yet critical to the success of an open-source project. Too much and the project becomes stifled. Too little and the project can go off the rails either as a success disaster or by getting bogged down from lack of enough structure to get things done. To date, the FreeBSD Project has gotten governance right, but keeping it humming requires constant tuning. ●

DR. MARSHALL KIRK MCKUSICK writes books and articles, teaches classes on UNIX- and BSD-related subjects, and provides expert-witness testimony on software patent, trade secret, and copyright issues particularly those related to operating systems and filesystems. He has been a developer and commiter to the FreeBSD Project since its founding in 1993. While at the University of California at Berkeley, he implemented the 4.2BSD fast filesystem and was the Research Computer Scientist at the Berkeley Computer Systems Research Group (CSRG) overseeing the development and release of 4.3BSD and 4.4BSD

BENNO RICE is a software engineer for Dell EMC's Isilon division and also a committer and core team member for FreeBSD.