The Release Engineering of 4.3BSD

Marshall Kirk McKusick

Michael J. Karels

Keith Bostic

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

This paper describes an approach used by a small group of people to develop and integrate a large software system. It details the development and release engineering strategy used during the preparation of the 4.3BSD version of the UNIX† operating system. Each release cycle is divided into an initial development phase followed by a release engineering phase. The release engineering of the distribution is done in three steps. The first step has an informal control policy for tracking modifications; it results in an alpha distribution. The second step has more rigid change mechanisms in place; it results in a beta release. During the final step changes are tracked very closely; the result is the final distribution.

1. Introduction

The Computer Systems Research Group (CSRG) has always been a small group of software developers. This resource limitation requires careful software-engineering management as well as careful coordination of both CSRG personnel and the members of the general community who contribute to the development of the system.

Releases from Berkeley alternate between those that introduce major new facilities and those that provide bug fixes and efficiency improvements. This alternation allows timely releases, while providing for refinement, tuning, and correction of the new facilities. The timely followup of ‘‘cleanup’’ releases reflects the importance CSRG places on providing a reliable and robust system on which its user community can depend.

The development of the Berkeley Software Distribution (BSD) illustrates an *advantage* of having a few principal developers: the developers all understand the entire system thoroughly enough to be able to coordinate their own work with that of other people to produce a coherent final system. Companies with large development organizations find this result difficult to duplicate. This paper describes the process by which the development effort for 4.3BSD was managed.[Leffler1989a]

2. System Development

The first phase of each Berkeley system is its development. CSRG maintains a continuously evolving list of projects that are candidates for integration into the system. Some of these are prompted by emerging ideas from the research world, such as the availability of a new technology, while other additions are suggested by the commercial world, such as the introduction of new standards like POSIX, and still other

†UNIX is a registered trademark of AT&T in the US and other countries.

projects are emergency responses to situations like the Internet Worm.

These projects are ordered based on the perceived benefit of the project as opposed to its difficulty; the most important are selected for inclusion in each new release. Often there is a prototype available from a group outside CSRG. Because of the limited staff at CSRG, this prototype is obtained to use as a starting base for integration into the BSD system. Only if no prototype is available is the project begun in-house. In either case, the design of the facility is forced to conform to the CSRG style.

Unlike other development groups, the staff of CSRG specializes by projects rather than by particular parts of the system; a staff person will be responsible for all aspects of a project. This responsibility starts at the associated kernel device drivers; it proceeds up through the rest of the kernel, through the C library and system utility programs, ending at the user application layer. This staff person is also responsible for related documentation, including manual pages. Many projects proceed in parallel, interacting with other projects as their paths cross.

All source code, documentation, and auxiliary files are kept under a source code control system. During development, this control system is critical for notifying people when they are colliding with other ongoing projects. Even more important, however, is the audit trail maintained by the control system that is critical to the release engineering phase of the project described in the next section.

Much of the development of BSD is done by personnel that are located at other institutions. Many of these people not only have interim copies of the release running on their own machines, but also have user accounts on the main development machine at Berkeley. Such users are commonly found logged in at Berkeley over the Internet, or sometimes via telephone dialup, from places as far away as Massachusetts or Maryland, as well as from closer places, such as Stanford. For the 4.3BSD release, certain users had permission to modify the master copy of the system source directly. People given access to the master sources are carefully screened beforehand, but are not closely supervised. Their work is checked at the end of the beta-test period by CSRG personnel who back out inappropriate changes. Several facilities, including the Fortran and C compilers, as well as important system programs, for example, **telnet** and **ftp**, include significant contributions from people who did not work directly for CSRG. One important exception to this approach is that changes to the kernel are made only by CSRG personnel, although the changes are often suggested by the larger community.

The development phase continues until CSRG decides that it is appropriate to make a release. The decision to halt development and transition to release mode is driven by several factors. The most important is that enough projects have been completed to make the system significantly superior to the previously released version of the system. For example, 4.3BSD was released primarily because of the need for the improved networking capabilities and the markedly improved system performance. Of secondary importance is the issue of timing. If the releases are too infrequent, then CSRG will be inundated with requests for interim releases. Conversely, if systems are released too frequently, the integration cost for many vendors will be too high, causing them to ignore the releases. Finally, the process of release engineering is long and tedious. Frequent releases slow the rate of development and cause undue tedium to the staff.

3. System Release

Once the decision has been made to halt development and begin release engineering, all currently unfinished projects are evaluated. This evaluation involves computing the time required to complete the project as opposed to how important the project is to the upcoming release. Projects that are not selected for completion are removed from the distribution branch of the source code control system and saved on branch deltas so they can be retrieved, completed, and merged into a future release; the remaining unfinished projects are brought to orderly completion.

Developments from CSRG are released in three steps: alpha, beta, and final. Alpha and beta releases are not true distributions—they are test systems. Alpha releases are normally available to only a few sites, usually those working closely with CSRG. More sites are given beta releases, as the system is closer to completion, and needs wider testing to find more obscure problems. For example, 4.3BSD alpha was distributed to about fifteen sites, while 4.3BSD beta ran at more than a hundred.

3.1. Alpha Distribution Development

The first step in creating an alpha distribution is to evaluate the existing state of the system and to decide what software should be included in the release. This decision process includes not only deciding what software should be added, but also what obsolete software ought to be retired from the distribution. The new software includes the successful projects that have been completed at CSRG and elsewhere, as well as some portion of the vast quantity of contributed software that has been offered during the development period.

Once an initial list has been created, a prototype filesystem corresponding to the distribution is constructed, typically named **/nbsd**. This prototype will eventually turn into the master source tree for the final distribution. During the period that the alpha distribution is being created, **/nbsd** is mounted read-write, and is highly fluid. Programs are created and deleted, old versions of programs are completely replaced, and the correspondence between the sources and binaries is only loosely tracked. People outside CSRG who are helping with the distribution are free to change their parts of the distribution at will.

During this period the newly forming distribution is checked for interoperability. For example, in 4.3BSD the output of context differences from **diff** was changed to merge overlapping sections. Unfortunately, this change broke the **patch** program which could no longer interpret the output of **diff**. Since the change to **diff** and the **patch** program had originated outside Berkeley, CSRG had to coordinate the efforts of the respective authors to make the programs work together harmoniously.

Once the sources have stabilized, an attempt is made to compile the entire source tree. Often this exposes errors caused by changed header files, or use of obsoleted C library interfaces. If the incompatibilities affect too many programs, or require excessive amounts of change in the programs that are affected, the incompatibility is backed out or some backward-compatible interface is provided. The incompatibilities that are found and left in are noted in a list that is later incorporated into the release notes. Thus, users upgrading to the new system can anticipate problems in their own software that will require change.

Once the source tree compiles completely, it is installed and becomes the running system that CSRG uses on its main development machine. Once in day-to-day use, other interoperability problems become apparent and are resolved. When all known problems have been resolved, and the system has been stable for some period of time, an alpha distribution tape is made from the contents of **/nbsd**.

The alpha distribution is sent out to a small set of test sites. These test sites are selected as having a sophisticated user population, not only capable of finding bugs, but also of determining their cause and developing a fix for the problem. These sites are usually composed of groups that are contributing software to the distribution or groups that have a particular expertise with some portion of the system.

3.2. Beta Distribution Development

After the alpha tape is created, the distribution filesystem is mounted read-only. Further changes are requested in a change log rather than being made directly to the distribution. The change requests are inspected and implemented by a CSRG staff person, followed by a compilation of the affected programs to ensure that they still build correctly. Once the alpha tape has been cut, changes to the distribution are no longer made by people outside CSRG.

As the alpha sites install and begin running the alpha distribution, they monitor the problems that they encounter. For minor bugs, they typically report back the bug along with a suggested fix. Since many of the alpha sites are selected from among the people working closely with CSRG, they often have accounts on, and access to, the primary CSRG development machine. Thus, they are able to directly install the fix themselves, and simply notify CSRG when they have fixed the problem. After verifying the fix, the affected files are added to the list to be updated on **/nbsd**.

The more important task of the alpha sites is to test out the new facilities that have been added to the system. The alpha sites often find major design flaws or operational shortcomings of the facilities. When such problems are found, the person in charge of that facility is responsible for resolving the problem. Occasionally this requires redesigning and reimplementing parts of the affected facility. For example, in 4.2BSD, the alpha release of the networking system did not have connection queueing. This shortcoming prevented the network from handling many connections to a single server. The result was that the networking interface had to be redesigned to provide this functionality.

The alpha sites are also responsible for ferreting out interoperability problems between different utilities. The user populations of the test sites differ from the user population at CSRG, and, as a result, the utilities are exercised in ways that differ from the ways that they are used at CSRG. These differences in usage patterns turn up problems that do not occur in our initial test environment.

The alpha sites frequently redistribute the alpha tape to several of their own alpha sites that are particularly interested in parts of the new system. These additional sites are responsible for reporting problems back to the site from which they received the distribution, not to CSRG. Often these redistribution sites are less sophisticated than the direct alpha sites, so their reports need to be filtered to avoid spurious, or site dependent, bug reports. The direct alpha sites sift through the reports to find those that are relevant, and usually verify the suggested fix if one is given, or develop a fix if none is provided. This hierarchical testing process forces bug reports, fixes, and new software to be collected, evaluated, and checked for inaccuracies by first-level sites before being forwarded to CSRG, allowing the developers at CSRG to concentrate on tracking the changes being made to the system rather than sifting through information (often voluminous) from every alpha-test site.

Once the major problems have been attended to, the focus turns to getting the documentation synchronized with the code that is being shipped. The manual pages need to be checked to be sure that they accurately reflect any changes to the programs that they describe. Usually the manual pages are kept up to date as the program they describe evolves. However, the supporting documents frequently do not get changed, and must be edited to bring them up to date. During this review, the need for other documents becomes evident. For example, it was during this phase of 4.3BSD that it was decided to add a tutorial document on how to use the socket interprocess communication primitives.

Another task during this period is to contact the people that have contributed complete software packages (such as **RCS** or **MH**) in previous releases to see if they wish to make any revisions to their software. For those who do, the new software has to be obtained, and tested to verify that it compiles and runs correctly on the system to be released. Again, this integration and testing can often be done by the contributors themselves by logging directly into the master machine.

After the stream of bug reports has slowed down to a reasonable level, CSRG begins a careful review of all the changes to the system since the previous release. The review is done by running a recursive **diff** of the entire source tree—here, of **/nbsd** with 4.2BSD. All the changes are checked to ensure that they are reasonable, and have been properly documented. The process often turns up questionable changes. When such a questionable change is found, the source code control system log is examined to find out who made the change and what their explanation was for the change. If the log does not resolve the problem, the person responsible for the change is asked for an explanation of what they were trying to accomplish. If the reason is not compelling, the change is backed out. Facilities deemed inappropriate in 4.3BSD included new options to the directory-listing command and a changed return value for the *fseek*() library routine; the changes were removed from the source before final distribution. Although this process is long and tedious, it forces the developers to obtain a coherent picture of the entire set of changes to the system. This exercise often turns up inconsistencies that would otherwise never be found.

The outcome of the comparison results in a pair of documents detailing changes to every user-level command[McKusick1986a] and to every kernel source file.[Karels1986a] These documents are delivered with the final distribution. A user can look up any command by name and see immediately what has changed, and a developer can similarly look up any kernel file by name and get a summary of the changes to that file.

Having completed the review of the entire system, the preparation of the beta distribution is started. Unlike the alpha distribution, where pieces of the system may be unfinished and the documentation incomplete, the beta distribution is put together as if it were going to be the final distribution. All known problems are fixed, and any remaining development is completed. Once the beta tape has been prepared, no further changes are permitted to **/nbsd** without careful review, as spurious changes made after the system has been **diff**ed are unlikely to be caught.

3.3. Final Distribution Development

The beta distribution goes to more sites than the alpha distribution for three main reasons. First, as it is closer to the final release, more sites are willing to run it in a production environment without fear of catastrophic failures. Second, more commercial sites delivering BSD-derived systems are interested in getting a preview of the upcoming changes in preparation for merging them into their own systems. Finally, because the beta tape has fewer problems, it is beneficial to offer it to more sites in hopes of finding as many of the remaining problems as possible. Also, by handing the system out to less sophisticated sites, issues that would be ignored by the users of the alpha sites become apparent.

The anticipation is that the beta tape will not require extensive changes to either the programs or the documentation. Most of the work involves sifting through the reported bugs to find those that are relevant and devising the minimal reasonable set of changes to fix them. After throughly testing the fix, it is listed in the update log for **/nbsd**. One person at CSRG is responsible for doing the update of **/nbsd** and ensuring that everything affected by the change is rebuilt and tested. Thus, a change to a C library routine requires that the entire system be rebuilt.

During this period, the documentation is all printed and proofread. As minor changes are made to the manual pages and documentation, the affected pages must be reprinted.

The final step in the release process is to check the distribution tree to ensure that it is in a consistent state. This step includes verification that every file and directory on the distribution has the proper owner, group, and modes. All source files must be checked to be sure that they have appropriate copyright notices and source code control system headers. Any extraneous files must be removed. Finally, the installed binaries must be checked to ensure that they correspond exactly to the sources and libraries that are on the distribution.

This checking is a formidable task given that there are over 20,000 files on a typical distribution. Much of the checking can be done by a set of programs set to scan over the distribution tree. Unfortunately, the exception list is long, and requires hours of tedious hand checking; this has caused CSRG to develop even more comprehensive validation programs for use in our next release.

Once the final set of checks has been run, the master tape can be made, and the official distribution started. As for the staff of CSRG, we usually take a brief vacation before plunging back into a new development phase.

References

Karels1986a.

M. J. Karels, “Changes to the Kernel in 4.3BSD” in *UNIX System Manager’s Manual, 4.3 Berkeley Software Distribution, Virtual VAX-11 Version,* p. 13:1−32, USENIX Association, Berkeley, CA (1986).

Leffler1989a.

S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.3BSD UNIX Operating System,* Addison-Wesley, Reading, MA (1989).

McKusick1986a.

M. K. McKusick, J. M. Bloom, and M. J. Karels, “Bug Fixes and Changes in 4.3BSD” in *UNIX System Manager’s Manual, 4.3 Berkeley Software Distribution, Virtual VAX-11 Version,* p. 12:1−22, USENIX Association, Berkeley, CA (1986).