Matt Jibson
ECE 501
Research Paper

OpenBSD's Lifecycle

Abstract

Although OpenBSD is not a standard commercial or military organization, their product aligns well to systems engineering principles. This paper will show how these principles correlate with OpenBSD's lifecycle and method.

Introduction

The OpenBSD project is a free 4.4BSD-based UNIX-like operating system developed by volunteers who emphasize goals of portability, standardization, correctness, proactive security, and integrated cryptography [1]. Although this project is not bound by external schedules or costs, internal deadlines and monetary donations create the same restrictions. Since the contributors to the project are all volunteers, there is never any guaranteed work that will be accomplished. Yet, the schedule is always met, and the quality is always high. One possible explanation of this success is the unofficial adherence to principles of systems engineering. OpenBSD's lifecycle and will be analyzed to show how it applies to systems engineering.

Before describing this, though, here is some background on the project. OpenBSD was derived from NetBSD 1.0 in October of 1995 [2], which is derived from 386/BSD, with roots back to AT&T UNIX [4]. Although OpenBSD was initially formed due to personnel differences [3], it now exists for a variety of goals, including providing the best development platform possible, finding and proactively fixing security problems, and cryptographic integration [5].

Main Body

To show adherence to systems engineering principles, OpenBSD's lifecycle can be used. There is a release every six months [6], a schedule that has been successfully sustained for 23 releases [7]. Because of this, the development cycle is well known and has been divided into three phases: (1) heavy development (which directly succeedes a release) where large, dangerous changes enter [8], about two months, (2) normal development, about three months, and (3) bug-fixes only [9], the final month. Release are generally supported for one year after release [10].

Concept Development

There are eight development phases as described by the text, each of which has a parallel in OpenBSD. The first phase listed is needs analysis. In OpenBSD this is done on a personal basis when a developer has a need (e.g., preventing a dDoS attack [11]) or desire [12] for a feature. Those two reasons satisfy the "Is there a valid need for a new system?" question defined by the text. Sometimes, however, there is no way to satisfy the second question ("Is there a practical approach to satisfying such a need?"). For example, the developers function largely autonomously [12] and so will not push others with ability to implement a specific feature [13]. Thus, if a developer wants something and is able to do it himself, then a feature will pass this phase. Occasionally developers are recruited and attracted to specific projects, however, but never begrudgingly.

Next is concept exploration. To address the question of system performance to meet the perceived need, the developers will consult among their peers and occasionally ask for help, or await features from others as precursors to their own. Generally, though, if something needs to change they will do it themselves. The second question of this phase regards affordable feasability. The only affordability constraint is hardware, documentation, and time. If the developer has the tools he needs and is willing to spend the time, it is affordable. Certainly time is most precious, and so if a feature is deemed to difficult and will take much time, it may be passed over or slowly worked on over a large time, interleaved with other work.

In the concept definition phase, there are generally two branches considered: (1) incorporate the work of another [14] and (2) roll a fully-custom solution [15]. These two branches are the result of considering the most beneficial balance between capability, operational life, and cost. Unless there is another solution that has been proven to meet the goals of the project (technically excellent, legally free to use, secure, etc. [5]), it is not considered as a candidate. If it is too time consuming to complete, they will either abandon the idea or try to attract other developers with similar interests to assist.

Engineering Development

Now let us assume the proposed feature has met all goals and requirements above, and move on to the engineering development stage. The first phase of this stage is advanced development. This phase is generally marked by deep thought and research by a developer to fully understand the system they wish to change [16]. Developers will, of course, consult each other about their ideas to determine the extent to which their change will affect other systems.

Next is the engineering design phase. This phase is often never seen by others (i.e., the interested public), or only by few of technical ability who are able to contribute ideas and reviews. Developers will often remain in this stage for years before any code is made public [16]. Frequent formal design reviews are conducted by evaluating and testing code-in-progress, which is usually submitted to a larger group for review [17]. The "ilities" are considered during reviews, and are of course built-in and expected to be of high quality, as per the project goals [5]. The role of systems engineer during this phase becomes vague. Those who have expertise in certain systems asses impacts and contribute where they are able, acting as a systems engineer in that they attempt to understand a broader picture. There are perhaps five developers of which at least one is always consulted before any substantial change.

Integration and evaluation are done continuously by providing daily snapshot releases [10]. These are tested by the developers and community-at-large who report bugs. In this

way experimental features (which are committed towards the beginning of a release cycle) get more exposure and testing. It is not uncommon for features to be disabled if they have not been deemed good enough for the formal release. As our text points out, a system must be fully engineered and built before it can be accurately assessed, which snapshots do not provide. Hence, directly before a release is made, calls for testing are done to the community where there is a last chance to report problems on the final snapshots[18].

Post Development

In the post development process, the production phase is generally supported by the project leader who oversees printing of the CDs and other release items. Maintainers of FTP mirrors are notified, as well, as they are involved in this process. Show-stopper problems are not often found, but when they are workarounds are determined [19].

Operation and support is done by mailing lists [20] and official bug reports [21]. Users are able to submit questions regarding the system, and the mailing list will provide help. Frequently asked questions are added to the FAQ [22]. When problems are found and corrected errata are posted [23]. Releases are supported for one year past release date [10]. In addition, copious documentation is provided with the system release and on the website.

Conclusion

Since the OpenBSD project does not specifically use systems engineering principles (but only happens to follow them, perhaps because both systems engineers and the project leaders are wise and experienced), it is prudent to asses the success of the project. On their webpage they tout to have had "[o]nly two remote holes in the default install, in more than 10 years" [1], a bold claim and challenge that no other modern operating system can make or approach. The popular OpenSSH suite, which is produced by OpenBSD, is used by 80% of SSH distributors [24]. This is further strengthened by the list of users of this software, which includes major systems including all BSD and Linux systems, Solaris, Irix, AIX, HP-UX, Juniper devices, Cisco devices, and others [25]. As this software (SSH) is the protocol to securely communicate

with other devices on the hostile Internet, it is clear that the world-at-large trusts OpenBSD.

Although the systems engineering principles are nowhere listed on the official website, and the developers are under no constraint to follow any set of guidelines, it is clear that OpenBSD adheres to these sound principles. This has helped the project achieve success and prestige in its field.

References

[1] http://openbsd.org/.

[2] http://www.openbsd.org/cgi-bin/cvsweb/src/Makefile?rev=1.1.1.1&content-type=text/x-cvsweb-markup.

[3] http://www.theos.com/deraadt/coremail.html.

[4] M. K. McKusick. Open Sources: Voices from the Open Source Revolution. http://oreilly.com/catalog/opensources/book/kirkmck.html.

[5] OpenBSD Project Goals. http://openbsd.org/goals.html.

[6] Introduction to OpenBSD - When is the next release of OpenBSD? http://openbsd.org/faq/faq1.html#Next.

[7] T. d. Raadt. 'Re: Rolling release?'. http://marc.info/?l=openbsd-misc&m=120891491814252&w=2.

[8] T. d. Raadt. 'Development at hackathon'. http://marc.info/?l=openbsd-misc&m=121316165810469&w=2.

[9] P. Valchev. 'ports tree "soft lock" '. http://marc.info/?l=openbsd-ports&m=111073355609501&w=2.

[10] OpenBSD's Flavors. http://www.openbsd.org/faq/faq5.html#Flavors.

[11] H. Brauer. 'What I did in April'. http://marc.info/?l=openbsd-misc&m=108370702404264&w=2.

[12] M. Espie. 'Re: OpenBSD kernel janitors'. http://marc.info/?l=openbsd-misc&m=119385449100584&w=2.

[13] M. Peereboom. 'Re: What is our ultimate goal??'. http://marc.info/?l=openbsd-misc&m=120327720917583&w=2.

[14] http://marc.info/?l=openbsd-cvs&m=99774412221528&w=2.

[15] OpenBSD release song lyrics - 3.5: "CARP License" and "Redundancy must be free". http://openbsd.org/lyrics.html#35.

[16] Hackathon Summary Part 1.
http://undeadly.org/cgi?action=article&sid=20080618203134.

[17] http://www.drijf.net/openbsd/malloc/.

[18] T. d. Raadt. 'Pre-release tests'.
http://marc.info/?l=openbsd-misc&m=120468537309557&w=2.

[19] A. Hook. 'Re: Marginal boot CD #1 in OpenBSD 4.2 sets'.
http://marc.info/?l=openbsd-misc&m=119386241612594&w=2.

[20] OpenBSD Mailing lists. http://openbsd.org/mail.html.

[21] OpenBSD problem reports. http://openbsd.org/report.html.

[22] OpenBSD Frequently Asked Questions. http://cvs.openbsd.org/faq/.

[23] Applying patches in OpenBSD. http://openbsd.org/faq/faq10.html#Patches.

[24] SSH usage profiling. http://openssh.org/usage/graphs.html.

[25] Systems using OpenSSH. http://openssh.org/users.html.