Using FreeBSD to Promote Open Source Development Methods

Brooks Davis, Michael AuYeung, Mark Thomas
The Aerospace Corporation
El Segundo, CA
{brooks,mauyeung,mathomas}@aero.org

Abstract

In this paper we present AeroSource, an initiative to bring open source software development methods to internal software developers at The Aerospace Corporation. Within AeroSource, FreeBSD is used in several key roles. First, we run most of our tools on top of FreeBSD. Second, the ports collection (both official ports and custom internal ones) eases our administrative burden. Third, and most importantly the FreeBSD project serves as an example and role model for the results that can be achieved by an open source software projects. We discuss the development infrastructure we have built for AeroSource based largely on BSD licensed software including FreeBSD, PostgreSQL, Apache, and Trac. We will also discuss our custom management tools including our system for managing our custom internal ports. Finally, we will cover our development successes and how we use projects like FreeBSD as exemplars of open source software development.

1 Introduction to Aerospace

The Aerospace Corporation operates a Federally Funded Research and Development Center for National Security Space. From the corporate web site[Aerospace]:

> Since 1960 The Aerospace Corporation has operated a federally funded research and development center in support of national-security, civil and commercial space programs. We're applying the leading technologies and the brightest minds in the industry to meet the challenges of space.

The company employs approximately 2400 engineers on a wide range of disciplines. In today's engineering climate, a large potion of these engineers write software, up to thousands of programs by some counts.

Due in part to the fact that these engineers are not trained software developers, the quality of software and software development methods varies widely. Since Aerospace helps oversee the development of massive software projects, we have a significant number of people who are trained to develop these types of software. They represent one of two historical groups of developers at Aerospace. They use big, heavy development processes which produce reliable software suitable for all sorts of applications, but require significant numbers of full-time developers and large paper trails.

The other camp takes a laissez-faire approach to software development. They tend to use little or no processes to the point that one of the more advanced groups was using a shared file system for development with a white board to lock files before the AeroSource team started working with them. As would be expected, this approach to development yields highly variable results. A number of pieces of software are very useful and some are even distributed outside the company, but even with those we've heard reports of problems like features disappearing between releases.

Past attempts to encourage developers of the more important pieces of software to adopt more rigorous development practices have met with limited success. One problem is that these developers quite reasonably fear the more heavy weight processes they see employed to build big systems. In addition to the process overhead of these methods developers worry about the cost of tools and the need to learn new tools. Other problems include inertia in the face of demanding schedules.

AeroSource is our current attempt to bring modern software development methods to the more ad-hoc development projects within Aerospace. We are promoting the idea that using tools and methods from open source software development provides a useful midpoint between big, expensive software methods and current practices. In addition to promoting open

source software and development methods, we are also promoting the open source development philosophy within the company. We call this internal open source, enterprise source software. Enterprise source software enshrines principles of open source, but is restricted to the enterprise. Users of enterprise source are free to read the source code, build and run it, make changes to it, and redistribute modified versions of it as long as they do so within the bounds of the company. External software distribution remains governed by existing processes.[1]

In the rest of this paper we discuss our experiences designing, developing and promoting AeroSource and the enterprise source concept. We discuss our use of FreeBSD throughout, both as the foundation of our infrastructure and as an example of both what can be achieved with open source methods and one set of highly effective methods. In the next section we discuss open source and enterprise source software. We then discuss our efforts to promote the enterprise source concept and the reactions we have encountered. Coming from the open source software world, we often find it hard to credit the issues people raise, but we have found it is critical to do so if we are going to convince people to support enterprise source. As part of this section we discuss our implementation of AeroSource, a resource for collaborative software development using FreeBSD and other open source technologies. We also talk about our successes and failures in recruiting projects to use it. Finally we conclude with a discussion of future directions for AeroSource.

2 Open Source and Enterprise Source Software

According to the Open Source Initiative "open source is a development method for software that harnesses the power of distributed peer review and transparency of process."[OSI] Many definitions of open source exist including the OSI *Open Source Definition*[OSD]. For our purposes we define an open source project as one that allows the four freedoms defined by the Free Software Foundation[Wikipedia] (the wording below is ours):

- The freedom to run the software, for any purpose
- The freedom to study how the software works, and adapt it to your needs
- The freedom to redistribute copies
- The freedom to improve the software, and release the improvements to anyone for the benefit of all

Advocates of the open source development mode argue it has numerous benefits. Those benefits include "better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in"[OSI]. These benefits derive directly from the four freedoms listed above. The quality and reliability claims derive from the idea that with more people working on the code, bugs are more likely to be discovered and fixed. There is a common idea in the open source community that "given enough eyeballs, all bugs are shallow"[CatB]. In our experience this is true for simpler bugs, but for very complex issues, there often are not enough people who understand the problem for this to work. One case where we do find quality to be better is adherence to unified code styles. In our experience and that of others we have talked to, large open source projects tend to have cleaner, more readable source code that internally developed code. Flexibility and protection from lock-in derive from the fact that users can modify the software themselves or hire someone to make the changes they want. As a result users can adapt to unforeseen software needs and add the functionality they want rather than things the developers' marketing department thinks they can sell. Lower cost is obvious since the software is free.

In addition to these benefits, open source development methods provide other advantages within the enterprise. Because open source developers often have another day job, they generally can not be bothered with excessively involved procedures. Thus, open source projects tend to use processes that are low friction. By adopting these processes, developers can build higher quality software without resorting to traditional, high overhead methods. Another benefit within the enterprise is that if people publicly share their code and others can find it, duplication of effort can be reduced. For example the world only needs so many tools to parse the same telemetry format.

Much of the software produced at Aerospace that would benefit from the wider exposure open source development brings is not possible or practical to release to the general public for a range of technical, legal, and political reasons. When promoting open source methods, we discovered we needed a term to describe the internal use of those methods since simply talking about open source or internal open source often lead people to think we would be posting their code to Source Forge or another public site. To capture this concept we coined the term *enterprise source software*. Enterprise source software is everything that open source software is, but restricted to an enterprise. All of the four freedoms hold for enterprise source soft-

[1] We have ambitions to encourage the release of more Aerospace code as open source. Promoting enterprise source the first of several steps in that direction.

ware, but with the added restriction that it must stay within the organization. At Aerospace this means that enterprise source source software may leave the company only though official software release channels.

We believe that the growth of enterprise source at Aerospace will improve the quality of the software we develop and increase the skill of our software developers.

3 Promoting Enterprise Source

In an effort to improve the development practices used by the less formal software projects at Aerospace we are working to promote the enterprise source concept for internal use. Our efforts center on encouraging the internal publication of source code and the use of open source tools and methods to develop that software. AeroSource, our internal collaborative software development environment, lies as the heart of our efforts. It allows users to "get their feet wet" without all the effort of maintaining their own systems. We discuss AeroSource in detail later in this section.

3.1 Promotion Efforts

Our promotional efforts are targeted in several different directions. We work to educate Aerospace employees on the benefits of open source software and development methods and encourage them to adopt them where practical. We also work to convince management of these benefits to support them from above in addition to our more grass roots efforts.

The most basic level of advocacy is using open source software or open source derivatives. Most people in our organization use BSD, Linux, or Mac OS exclusively and all our department servers are hosted on open source OSes. We also host a variety of semi-official corporate services including open source software mirrors, a list server, and a number of wikis.

The next level of advocacy is formal open source education. We have given a number of lectures on benefits of open source and open source development methods at internal forums. We also developed a tutorial on open source development methods which we presented at the Ground System Architecture Workshop in 2007[GSAW]. In these talks we promote the variety of great software available as open source both for it's own sake and to demonstrate that the non-traditional development efforts involved can and do produce top quality software.

FreeBSD is a key component of this promotion effort. We use it extensively in our infrastructure and because we are extremely familiar with it's development process, we can speak with authority on the processes involved. This is helpful in convincing people that we really do know what we are talking about with regard to open source development. The availability to resources such as the *FreeBSD Developers Handbook*[GSAW] and Robert Watson's *How the FreeBSD Project Works*[Watson] talk helps in this regard. Other projects we use as examples include Ganglia, K Desktop Environment (KDE), and Linux.

The most specific form of advocacy is AeroSource. With AeroSource we give developers the tools they need and help train them in the tools and best practices for using them. We help with things like repository layout, process, and usage. Eventually we hope to provide continuous integration tools like tinderboxes.

3.2 An Overview of AeroSource

AeroSource provides collaborative tools to software projects including tightly integrated version control, bug tracking, and a wiki. We also provide e-mail lists that can be integrated with the bug tracking and version control systems. This functionality is provided by Trac and GNU Mailman with version control provided by Subversion. Trac is one of several projects that aim to create a complete, web-based collaborative environment for open source development. Trac is open source (BSD licensed) and is built on top of a large stack of other open source software. In our installation we use Subversion for version control, PostgreSQL as the database, Apache for the web servers, and FreeBSD for the operating system.

When developing AeroSource we looked at several alternatives including GForge, SourceForge, and building our own system. GForge was rejected due to prior experience: it worked, but upgrades were time consuming and difficult. SourceForge was not an option because we wanted to keep software internal and we were not prepared to purchase the commercial version. After finding Trac we concluded that any benefits from building our own system were likely to be minimal compared to starting with an already working system. Systems we did not consider at the time but would consider today include CollabNet and Retrospectiva.

Today, AeroSource contains over 50 projects ranging from small repositories of scripts to large established projects. Our most prominent win is is the Satellite Orbital Analysis Program (SOAP), a cross platform (MacOS, UN*X, and Windows) 3D orbit visualization

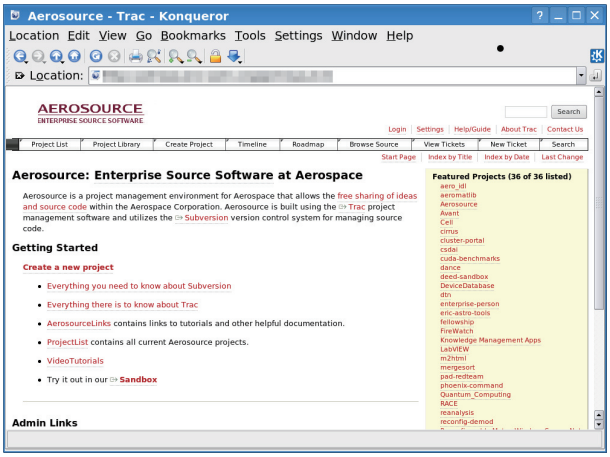

Figure 1: AeroSource.aero.org

and analysis program. SOAP has been under development for more than a decade in numerous forms and is one of Aerospace's crown jewels, so winning the development team over was a major milestone for AeroSource. Other projects include collections of Perl, IDL, and Matlab scripts and configuration files for a number of internal systems including AeroSource it self.

Users seem generally pleased with Trac and Subversion, but we have encountered a few problems. The most severe one is that the wiki implementation has no support for simultaneous edits. In it's current form, if two users edit the same page, the second user loses all their work when they submit. This is arguably the worst of all possible behaviors. Otherwise, Trac is working fairly well for us. The only other significant issue we have found is that some parts of Trac are more easily customized that others. There is a solid plugin framework, but if what you want cannot be accomplished through it maintaining modifications can be complex.

3.3 Maintaining AeroSource

With AeroSource, we do our best to "eat our own dog food" and use Trac and Subversion as much as possible to aid in maintenance. We store configuration, custom Trac modules, and administration scripts in AeroSource. The project homepage is a Trac instance and we use the ticket system to track most maintenance operations. The AeroSource front page can be seen in Figure 1.

The maintenance operations of AeroSource are fairly normal, but a few things stand out. We use `freebsd-update` to keep the base system up to date and install most of our software using the FreeBSD ports collection.

One one very useful customization we have developed is a set of local ports stored in an AeroSource hosted subversion repository. We call this collection AeroPorts. We check our ports out under `/usr/ports/aero` and ports live in `<category>/<port>` subdirectories. Figure 2 shows the top-level `Makefile` and Figures 3 and 4 show an example of the make files for each `<category>` sub directory. With this setup, we can easily maintain local ports of things that are not useful to the general public, or custom modifications of existing ports to perform non-standard tasks. One example of this is a custom version of the `security/pam_ldap` port that authenticates based on LDAP queries on userid's instead of usernames. Another is the `misc/aero-bootstrap` port which is a meta-port we use to install basic administrative tools on our FreeBSD machines. This method of incorporating local ports in the ports tree is based on a suggestion by Scot Hetzel on the freebsd-ports mailing list[Hetzel].

To simplify management of these local ports we have a wrapper for the `portsnap` and `svn` commands call `apt` (Aerospace Ports Tool). The `apt` command performs an `svn update` and `portsnap update` using the "`-l descfile`" option to refresh the ports tree and build combined `ports/INDEX*` files as needed. This yields functionality virtually identical to that of `portsnap`, but with full integration of our local ports.

3.4 Results

Thus far, our efforts have met with a number of successes, but we still have some work to do. As we mentioned in Section 3.2 we have recruited over 50 projects to AeroSource thus far. We have also had some projects that were not able to become enterprise source software express interest in the tools.

The import of the Satellite Orbital Analysis Program (SOAP) to AeroSource represents a major win and were in fact funded to make the transition. The first release has not yet been cut, but development is well under way and the developers have made significant progress in using the tools.

Some other projects have resisted the idea for a variety of reasons. Some want absolute control over the code they perceive ownership of. Reasons for wanting that control range from not wanting others to see their code to wanting to ensure that no one releases a modified version lest they be blamed for bugs introduced by someone else. We have had some success with the first case and a bit with the second, but we have not won all

```
COMMENT=        Ports specific to Aerospace Corp

SUBDIR+=        archivers
SUBDIR+=        astro
SUBDIR+=        misc
SUBDIR+=        net
SUBDIR+=        science
SUBDIR+=        shells
SUBDIR+=        sysutils

descfile:
        @cd ${.CURDIR}; ${MAKE} describe | grep -v '^===>' > descfile

.include <bsd.port.subdir.mk>
```

Figure 2: aero/Makefile

```
COMMENT=        Local Aerospace system utilities

SUBDIR+=        apt
SUBDIR+=        diskprep-aero
SUBDIR+=        macports
SUBDIR+=        powerctl

.include <bsd.port.subdir.mk>
```

Figure 3: aero/sysutils/Makefile

```
# This file needs to be copied into every aero/*/ subdirectory to set
# common variables.

# Used to set the origin of the local port
PKGORIGIN=      aero/${PKGCATEGORY}/${PORTDIRNAME}

# Used in the local ports tree to set dependencies on other local ports.
AEROPORTSDIR=  ${PORTSDIR}/aero

#Uncomment if you want your local packages to have a "-aero" suffix.
#PGKNAMESUFFIX?= -aero
```

Figure 4: aero/sysutils/Makefile.inc

the arguments. In once case we have even heard that developers have threatened to quit if forced to open their code. A variant of the argument that only the current developers know enough to modify the code is that only the developers can use the code properly. We agree this can happen, but think that is not in and of itself a good reason not to open the code. These were all arguments we expected to some extent based on past experiences. We also ran into a couple we were not expecting. In one case some people felt other developers should implement a certain algorithm as a right of passage. We weren't sure how we felt about that one. In another case, developers were concerned that people might like their code so much they should improve it and then the developers would have to incorporate the improvements. We thought seemed like a good thing rather than a problem.

4 Future Directions & Conclusions

We are generally pleased with our progress in introducing open source software development methods to Aerospace. We a large organization that is largely staffed by engineers with decades of experience, we do not expect to convert everyone over night. We feel many pieces of software within Aerospace could also benefit from full, open source release, but for now we are content with modernizing internal development efforts.

AeroSource itself is functioning very well. We hope to continue to incremental improve the management processes to make project setup easier and to enhance the ability of our users to perform their own project maintenance. We will also continue to monitor Trac development and the development of competing systems to provide our users with the best environment we can. Other future work includes more tutorial materials and more streamlined startup processes for new projects.

References

[Aerospace] The Aerospace corporate web site. October 11, 2007. `http://www.aero.org/`

[CatB] Eric S. Raymond. *The Cathedral and the Bazaar*. September 11, 2000. `http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/`

[GSAW] The FreeBSD Project. *The FreeBSD Developers' Handbook*. `http://www.freebsd.org/doc/en_US.ISO8859-1/books/developers-handbook/index.html`

[GSAW] Brooks Davis, Sam Gasster, Jorge Seidel, Mark Thomas. *Open Source Software Methods in Ground Systems*.

[Hetzel] E-mail to the `freebsd-ports@freebsd.org` mailing list. November 14, 2006. `http://docs.freebsd.org/cgi/mid.cgi?db=irt&id=790a9fff0611141011q4bd9ee97h9357e6d959f95abb@mail.gmail.com`

[OSD] The Open Source Initative's Open Source Definition. July 7, 2006. `http://www.opensource.org/docs/osd`

[OSI] The Open Source Initative web site. January 23, 2008. `http://www.opensource.org/`

[Watson] Robert N. M. Watson. *How the FreeBSD Project Works*. In Proceedings, 2006 EuroBSDCon, Milan, Italy.

[Wikipedia] Wikipedia article on free software. January 23, 2008. `http://en.wikipedia.org/wiki/Free_software`
