Sam Auciello
December 2011

My Experiences with Open Source Software

This semester, in an attempt further my education in the art of making great software, I began looking at how software is made in the open source world. I chose to look at this field for several reasons. I've been quite interested in open source software as long as I've known about it. I appreciate the social/economic benefits of software that is made by individuals who want it to be of quality as opposed to corporations that simply want to make money (and are therefore frequently motivated to compromise quality for business reasons e.g. planned obsolescence etc.). My experience with academic vs. professional programming also leads me to believe that people are far more likely to cut corners when they're getting paid. Another reason for looking into open source was that large-scale software is often made in teams. I had had very little experience working on software with a group and, being at tiny isolated liberal arts school with a similarly tiny computer science department, I realized that this was a key missing component of my education. This would also give me an opportunity to compare my coding to others' so as to get a sense of how I need to improve.

I began the semester thinking I would be contributing to Mozilla Firefox[1] and I spent the first few weeks looking into this. In the process I was exposed the XUL[2] and was able to build a very simple Firefox extension as well and getting a glimpse at the guts of Firefox. I quickly realized that my lack of familiarity with XUL, C++, and Firefox development in general was going to make contributing difficult and that I would likely spend more time exploring these technologies than learning about open source. For this reason, I switched my focus to

---

[1] http://www.mozilla.org/

[2] Extensible Markup Language, a language not unlike HTML for building extensible portable user interfaces, https://developer.mozilla.org/En/XUL

Wordpress[3], a project I was already much more familiar with. I had previously dabbled in creating plug-ins for Wordpress so contributing to the core in PHP (a language I am quite familiar with) seemed far less daunting. Although I would still love to someday learn C++ and XUL and explore the Mozilla community, I'm glad I made this decision because it allowed me to actually engage with a community the maintains a popular CMS used millions of bloggers. I made several small contributions to the swiftly approaching release of Wordpress 3.3[4] (as of the time of this writing there is a single ticket impeding the release; it involves adding polish to the admin toolbar icons[5]). My contributions included three major patches and a bug report. I got the opportunity to interact with several of the movers and shakers of the community and the experience was, on a whole, intimidating yet gratifying. I intend to continue contributing to Wordpress.

I also spent a some time exploring some of the tools of open source software such as version control systems. Although Wordpress uses Subversion, the software world at large seems to be slowly moving towards distributed version control systems like Git and Mercurial. I've become somewhat familiar with Mercurial and I hope to find more excuses to work with it in the future. I also looked a bit at software licenses. In particular, I looked at the differences between restrictively open "free" licenses like GPL and LGPL and much simpler ones like the MIT license. I didn't spend much time on this subject as it doesn't directly relate to programming and would probably be more interesting to a law student but it was good to get a sense of what exactly is meant by terms like open and free when they are applied to software.

I also spent some time this semester looking at subjects tangential to open source but directly related to becoming a well rounded programmer. For example, I began formally acquainting myself with Perl. Perl seems to be quite powerful though intimidatingly open-

---

[3] http://wordpress.org/

[4] http://codex.wordpress.org/Version_3.3

[5] http://core.trac.wordpress.org/ticket/19475

ended. I've also begun learning Ruby, which I feel will soon replace Python as my go-to scripting language. Finally I dove into Emacs which is increasingly my default text-editor. I've spent the semester continuing to improve my general coding skills and using open source as focal point. I set out to get exposure to a larger world of software technology specifically in the open source community and I feel that I have done this. This is of course not to say that I'm done; the software world is enormous and there will always be more to learn.

Contributing

I did make some initial attempts to contribute to Mozilla Firefox. I found the process inviting, with many helpful tutorials on how to get started. The main barrier to entry for me was that most of the tickets marked as "Good first bugs" in Bugzilla[6] were like Greek to me. They often referred to aspects of C++ or the Mozilla core that I wasn't familiar with and assumed that I was. There were a couple of bugs that I was able to understand well enough to reproduce. The first was one involving Firefox's handling of <wbr /> tags, which was inconsistent with other browsers. There was some disagreement as to whether the behavior of Firefox was in fact incorrect. Given the clear lack of agreement this seemed like a bad place to start. The other bug involved a button in the bookmark interface that needed to be disabled in a certain circumstance. This seemed like something that I might be able to fix but before I got the chance to a patch was submitted by someone else. This was about the time I decided to switch to looking at Wordpress.

Wordpress was slightly less initially inviting. Unlike with Firefox, there was no "so you want to contribute to Wordpress" page with helpful step by step instructions. There was a general description of how to get involved, which I followed. The main difference was my

---

[6] Mozilla's bug tracking system, https://bugzilla.mozilla.org/buglist.cgi?quicksearch=sw:[good%20first%20bug]

familiarity with the underlying technologies. About half of the tickets in trac[7] were clear enough that I could begin looking into them without much trouble at all and the other half would become more straight forward if I looked at them more. My first contribution was to ticket #18314[8] which was about cleaning up duplicate entries in the admin style-sheet. I took a look a the offending CSS file and quickly found a section where a large block of identical rules was being applied separately to three different selectors. I merged the identical rules into one block, made a patch using svn diff and submitted it to Trac. My patch was soon committed and so I decided to do a once over of the CSS file and and I made many similar fixes.

Another ticket I contributed to was ticket #18866[9]. It referred to a bug in the error page displayed if the wp-config.php file is missing. Another bug having to do with certain situations where the URL of the appropriate style-sheet for the error page might not be accessible had necessitated putting the CSS rules for the page in-line in the HTML head. A couple of rules had been forgotten so I tracked down the file that was generating the page and added the necessary rules. I found it necessary to include a reference to a background gradient image, causing the same problem as including a style-sheet would have, except that the difference between the page with and without this gradient was barely noticeable so this seemed like the solution most likely to work most of the time. Another contributor suggested using CSS gradients but, as they are not supported by Microsoft Internet Explorer prior to version 10, my patch was used.

I also fixed a bug in the “Press This” interface (ticket #18989[10]). Press This is a bookmarklet that opens a pop-up window allowing bloggers to easily post about content they find while browsing other sites. In addition to automatically pasting a link to the site into the post the bookmarklet grabs any selected text as well as providing an interface to include in-line images from the page. A change from another ticket had caused a bug in the image including

---

[7] Wordpress’ bug tracking system, http://core.trac.wordpress.org/

[8] http://core.trac.wordpress.org/ticket/18314

[9] http://core.trac.wordpress.org/ticket/18866

[10] http://core.trac.wordpress.org/ticket/18989

functionality. Everything seemed to work fine but the image was not pasted into the post. Another contributor had pointed to the offending bit of code. It seems that an HTML input tag with the same id had been added to the interface and the Jquery selector that got the image URL to insert it into the post was now no longer specific enough. I added a more specific selector but this didn't fix the problem. By adding console.log() statements, I was able to follow a relatively complex chain of seemingly unnecessary functions and discover a place where the HTML to be inserted into the post was being passed as the only argument to function that expected it as the second argument. Not wanting to break another aspect of the application, I passed the parameter as both the first and second arguments and the bug was fixed. If I had been more comfortable with general workings of the editing interface I'd have looked further to see why this was so excessively complex but this felt like a much larger project so I just made it work in a way that was unlikely to cause conflicts. It strikes me that this sort of apathy is probably what allows this code to remain as complex as it is. Perhaps I will look into cleaning it up for the next release.

I also found a bug and created ticket #19119[11]. A major feature had been added creating fly-out menus for the main admin navigation in ticket #18382[12]. I noticed relatively early that the changes being made had caused the collapsed menu fly-outs to be off by one pixel. Since there were a lot of developers working on this at the time I assumed that someone was on it and was worried that too many cooks might spoil the sauce. Apparently I was wrong because ticket #18382 was closed with the notice "If you encounter any bugs, please open a new ticket." The bug was still there so I opened a new ticket. I tried submitting a patch to it but a better one was used by someone with a better understanding of the CSS involved. I also made several smaller contributions in the form of testing patches and commenting on discussions.

---

[11] http://core.trac.wordpress.org/ticket/19119

[12] http://core.trac.wordpress.org/ticket/18382

The Open Source Community

My initial reaction to the communities I looked at was that they were very inviting. This was unsurprising as it seems like a necessity to an organization fueled by volunteers. It was interested to see how decision were made within these communities. The Mozilla approach is a complex hierarchical structure that rewards longstanding contributors with the ability to make more serious decisions. Wordpress being smaller, has a much simpler approach. There is a core Wordpress team that has commit access to the Subversion repository. Anyone else can check out the repository and submit changes to Trac in the form of patches (anyone can create an account in Trac). Because Wordpress is relatively small, it's rather trivial for the core team to keep track of tickets with patches in them and commit things that need to be committed.

It was interesting to see the varying quality of documentation in Wordpress code. Any function intended to be used by plug-in/theme developers is described in detail in the Wordpress Codex[13] in terms that can be understood by someone who has very little programming experience. Anything internal is lucky to have more than basic in-line comments next to it in the code. This also varies widely from file to file. Most of the files have a comment block at the top explaining the file's purpose but this is often all you get and you have to hope the code is well written (which it only sometimes is). It's not clear to me why there is such a discrepancy between end user documentation and developer documentation. It seems reasonable that user documentation would be prioritized as it is part of making the product usable but it seems like it would be just as important to provide support to developers so to help developers improve the core.

One thing that surprised me that seemed to be a major feature of the community was a

[13] http://codex.wordpress.org/

general sense of inertia. A debate came up in the wp-hackers email list[14] about a month ago about the differences between relative and absolute URLs. Whenever an internal image or link is added to a post or page on a Wordpress site, the editor stores an absolute URL in the database. The strongest argument for this seems to be that absolute URLs will fully specify the correct location in any context and since the pages therefore always want to be served that way, they shouldn't need to be processed for every page load. The strongest argument against this seemed to be the issue of portability. If development is at dev.example.com and a production is at example.com we shouldn't need to fix the database any time we push the database to production. Various other arguments were made and things got a bit out of hand at times but the end result was that since the core team was mostly in the former camp nothing was going to change in the Wordpress core but it was fine to implement relative URLs in a plug-in (this was not particularly satisfactory for the proponents of relative URLs since the plug-in solution turned out the be somewhat clunky). "Make it a plug-in" seems to be an unfortunately common answer to feature requests that don't seem necessary to the use case that the core team has in mind. For example re-arranging the order of your pages is an annoyingly complex task of meticulously changing a number in a field on each page. There is a plug-in that provides a nice drag-and-drop interface for this but no one has considered incorporating it into the core[15]. It strikes me as unfortunate that this sort of inertia exists and I hope that it isn't as prevalent in the open source community at large as it is in Wordpress as it seems like a key barrier to competition with commercial software.

Gaining Programming Skills

One of my main goals for the semester was to gain skills in group development. To this

[14] http://codex.wordpress.org/Mailing_Lists#Hackers

[15] http://wordpress.org/extend/plugins/my-page-order/

end, I spent a good deal of time acquainting myself with the relevant tools. One of the most important tools in group development is version control. The three version control systems I looked at were Subversion, Mercurial, and Git. Subversion was clearly a vast improvement on its predecessor CVS, but it seems to be somewhat outdated at this point because the advantages of distributed version control are significant. Unfortunately it is still quite prevalent. Mercurial seems to be a really nice solution as it is fast, powerful, distributed, and relatively easy to learn. Git seems to be the most powerful of the three but is harder to learn. My current preference is for Mercurial which I have started using for some of my own projects.

The experience of working with other people's code was helpful in general. My experience with ticket #18989 in particular drove home the importance of keeping code simple and easy to follow. It also re-affirmed my appreciation for the value of debug statements.

I also spent some time looking at Software licenses. Particularly I looked at the differences between GPL, LGPL, and more permissive licenses like the MIT license. GPL is restrictively free. This sounds like a contradiction but the idea is basically that if a piece of software is licensed under GPL then any software built on top of it must also be GPL. This ensures that the use of this code will never be subject to proprietary endeavors. LGPL is less restrictive. It allows proprietary software to be built on top of it so long as the licensed software is unmodified. LGPL is common for programming languages and other situations where GPL would severely limit the products usability

I gained a lot of general knowledge/skills this semester that will help me in my goal of becoming a great software developer. In addition to gaining familiarity to the Wordpress core, I got exposure to XUL, Version Control, software licenses, Perl, and Ruby. Although my semester was not as focused on contributing to open source as I originally intended it to be, I don't see this as having been a problem. The nature of software is that making it well is often as much about knowing how to use the tools as it is about understanding the concepts. I feel

like this semester gave me the opportunity to become acquainted with a number of powerful tools of the trade. Even though it didn't necessarily coalesce into a clearly structured curriculum, it gave me a chance to learn what I need to learn.