Managing Open Source Projects – A Case Study

James W Miller

millerjw@world.oberlin.edu

A paper submitted in partial fulfillment of the requirements of
TS5130 System Development Theory and Practice
Capella University

Instructor: Dr. Nancy Johnson

September 21, 2004

Abstract

Traditional proprietary software is delivered as object code only. The producer of the software keeps the source code secret to protect the investment in the development of the code. Open Source software is delivered in source code format and sometimes in object code format as well. The source code is open for all to see and even to modify and redistribute under certain conditions. Open Source software development and distribution is now established practice. This paper analyzes three Open Source projects in the Center for Realtor Technology at the National Association of Realtors and measures these projects against creiteria for successful Open Source projects found in the literature.

Table of Contents

Table of Figures

Introduction

Open Source Software - Description

Traditional proprietary software is delivered as object code only. The producer of the software keeps the source code secret to protect the investment in the development of the code. Open Source Software (OSS) is delivered in source code format and sometimes in object code as well. The source code is open for all to see and even to modify and redistribute under certain conditions. Customers, users and anyone else that is interested can see the code and may even submit bug reports and enhancement requests that refer to specific modules and line numbers. In fact, they might even submit modified code.

Open Source Software – Current State

This new approach to source code availability and distribution can no longer be considered highly unusual or experimental. The web site www.sourceforge.net (SourceForge.net, 2004) shows that there are now over 85,000 Open Source projects registered on this site alone. There are web sites such as http://freshmeat.net/ that provide information on the latest Open Source software releases and assist users in searching for relevant Open Source software.

Mainstream Information Technology resources such as the Gartner Group track Open Source Software developments and treat them seriously. Gartner's research note on team collaboration software treats Open Source software and proprietary commercial software equally. The note speculates that Open Source software could provide very significant competition traditional commercial software in the area of team collaboration starting in 2006. (Drakos, 2004). Gartner also documents the benefits of Open Source software.

> "The potential benefits of open-source efforts such as Eclipse include reducing technology acquisition costs, increasing the pace of innovation resulting from multivendor competition, improving quality from the networking effects of participating developers worldwide and from formalized review process, and lowering total costs from greater interoperability and reuse." (Blechar, 2004)

There are benefits to using Open Source software even if you do not view or modify the source code that you receive with the software. Michael Asay documents benefits such as lower prices for the customer, higher quality of source code and the security of knowing that the source code is available (Asay, 2004).

Major software such as Linux, Apache, PHP, and MySQL are all Open Source Software. Linux is an Operating System that competes successfully with proprietary operating systems such as Microsoft Windows and Sun Solaris. Apache (Apache, 2004) is a Web Server that delivers HTML pages to client Web Browsers. Apache has a very large market share of the UNIX Web

Server market and it competes successfully with Microsoft IIS (Internet Information Server) in the Microsoft Windows Server market. The PHP Hypertext Processor (PHP, 2004) generates HTML from scripts and competes successfully with Microsoft ASP (Active Server Pages) and with JSP (Java Server Pages). MySQL (MySQL, 2004) is a relational database system that competes successfully with other Open Source databases as well as with Microsoft SQL2000 and ORACLE in many situations.

Scope of Paper

This paper summarizes the world of Open Source and then analyzes three Open Source projects in the Center for Realtor Technology at the National Association of Realtors. The paper measures the projects against criteria for successful Open Source projects found in the literature. The results of this work show what has been successful and also what might be done differently in future projects.

The World of Open Source

Comparison to the World of Publishing

In commercial publishing, successful authors are paid for their work. The most successful authors may receive payments in advance. The author, or possibly the publisher, retains all rights to the work and it may not be reproduced or distributed by others without permission. The author may receive a royalty for each copy of the work that is sold. The reputation of the author and the publisher's view of how well the work will sell will determine where (if) the work will be published. Where the work is published and how much effort is spent marketing the work will determine how many people see the work. An author "self publish" on the Web and attempt to convince Web surfers to view the publication.

In commercial software, software companies that publish and market software, hire programmers to write the works. The programmers are paid a salary but might receive other compensation based on the success of the company. The software company bears the reproduction, marketing expense and distribution expense. The content of the software are the sales and marketing effort and effectiveness of the software company determine how many people will use the software. Customers pay for the software and generally do not receive the rights to copy or redistribute the software. Customers in many cases do not receive a copy of the source code for the software.

In academic publishing, the author is many times not directly compensated for the work produced. The author is motivated by strong interest in the subject matter or by some indirect gain such as esteem in the academic community or the possibility of obtaining a better academic position. Anyone can freely make use of the work of others as long as the author is given proper credit. For example, this paper takes advantage of the work of many authors and has a References section to give proper credit. In general, the reputation of an author improves the more his or her work is referenced. As in commercial publishing, it is not easy to get work published especially in the most prestigious academic journals. The reputation of the author and the content of the work are important in determining where the work will be published. The author has the option to publish his or her work directly on the Web and attempt to convince others to view the work.

In Open Source, as in academic publishing, the author is many times not directly compensated for the work produced. Work can include documentation, program code, test cases, and other deliverables associated with system development. The author many not receive direct compensation for his or her work. The author may be motivated by strong interest in the technical work being performed. Indirect compensation can include increased reputation in the community and possible future consulting contracts. Anyone can freely make use of the work of others as long as credit is given. There are standard places in the project .documentation and in the source code to give proper credit. In general, the reputation of an author improves the more his or her work is downloaded and used. As in academic publishing, contributions of work are not always accepted where the author would like to have the work published. The reputation of the author and the content of the work will determine whether or not the work is accepted into a given Open Source project. If the author "self publishes" the work as a new project, the same factors will help determine whether or not the work will be used by others.

In both academic publishing and Open Source, it is a very serious offense to use the work of others without giving proper credit. Plagiarism leads to rapid loss of reputation and the ability to have future work accepted. It is acceptable to examine work critically and make critical comments on the work. In the Open Source community, personal attacks on the author of the work are considered very bad form (Raymond, 1999).

Review of Open Source Literature

There are three sources that fairly completely define the world of Open Source software from the technical, legal, cultural and economic points of view.

1. The Software Engineering Institute at Carnegie Mellon University has an excellent technical report that defines Open Source and analyzes a number of projects. (Hissam, 2001). Read this paper for a quick but thorough and objective analysis of Open Source.

2. The book *The Cathedral and the Bazaar* by Eric Raymond (Raymond, 1999) is a more passionate description of the way the world of Open Source operates and includes a chapter on approaches you could take to enter that world as a developer. This is a classic work that almost all Open Source developers have read. Consistent with Raymond's passionate commitment to open source, he has made the full text of the book freely available on the Web at http://www.catb.org/~esr/writings/cathedral-bazaar/. His work contains a list of rules for successful Open Source development that are reproduced in Appendix A.

3. The book *The Success of Open Source* (Weber, 2004) provides a good explanation of how the world of Open Source works economically.

Characteristics of Successful Open Source Projects

This list of characteristics of successful Open Source projects was generated from the sources above.

1. Successful projects start with a vision and an artifact (prototype) (Hissam, 2001). An individual or a small group can take the lead and generate a substantive core that promises to evolve into something truly useful. (Weber, 2004)

2. Successful project scratch the developer's itch to code (Raymond, 1999) Successful projects are technically cool. (Hissam, 2001) For example, this would imply that graphics editors are more apt to succeed in the world of Open Source than batch general ledger programs.

3. Successful projects use reasonable tool sets (version control, bug tracking, documentation …) (Hissam, 2001).

4. Successful projects provide a general community service (Hissam, 2001). The product is perceived as important and valuable to a critical mass of users (Weber 2004).

5. Developers also use the results of the project (Hissam, 2001)

6. The project has committed leaders (Hissam, 2001)

7. Release early, release often and listen to your customers. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource. (Raymond, 1999). The product benefits form widespread peer attention and review and can improve through creative challenge and error correction. (Weber, 2004)

8. Provided the development coordinator has a medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one (Raymond, 1999). A voluntary community of iterated interaction can develop around the process of building the product (Weber, 2004).

Description of Center for Realtor Technology Projects

The Center for Realtor Technology (CRT) is an advanced development organization in the National Association of Realtors (NAR). The NAR has one million members who are able to call themselves Realtors because they belong to the NAR. These Realtors may work through a Broker that provides office and computer services.

Mark Lesswing, the Vice President of the Center for Realtor Technology provided the information about the three projects that are described here. (Lesswing, 2004). Additional information was retrieved from the CRT web site. (CRT, 2004). In the project descriptions below, if a particular point relates to one of the eight success criteria mentioned in the previous section, a reference of the form (*SC n) will be used to indicate which success criteria applies. If a particular point strongly relates to one of Eric Raymond's rules in Appendix A, a reference of the form (*R nn) will be used to indicate which of Raymond's rules apply.

Retriever Project

The NAR defines rules and policies about how their members can communicate with Listing Services. The members wanted to be able to make a local copy of data from the Listing Services and then make that data available to home buyers who are the customers of the members. This provides the opportunity for the members or their employer to uniquely format the data as part of building their own unique brand. The Listing Services wanted the members to put frames around the Listing Services web pages instead of transferring data to a Broker's machine.

The NAR proposed a concept called Transient Download as shown in Figure 1. The NAR member is shown as "Broker" in this figure. In the Transient Download approach, software running on the member's machine is able to request specific listing data and can receive this data in a standard format. The member's software is then free to display the listing data in any format. Data is not permanently stored on the member's machine.

Members believed that this concept would be difficult to implement and did not want to implement it. Therefore, the CRT established a project to build generalized software that could retrieve the listing information from the Listing Services, parse it, and prepare it for display. Members could then continue to use HTML to display the retrieved information in any format desired.

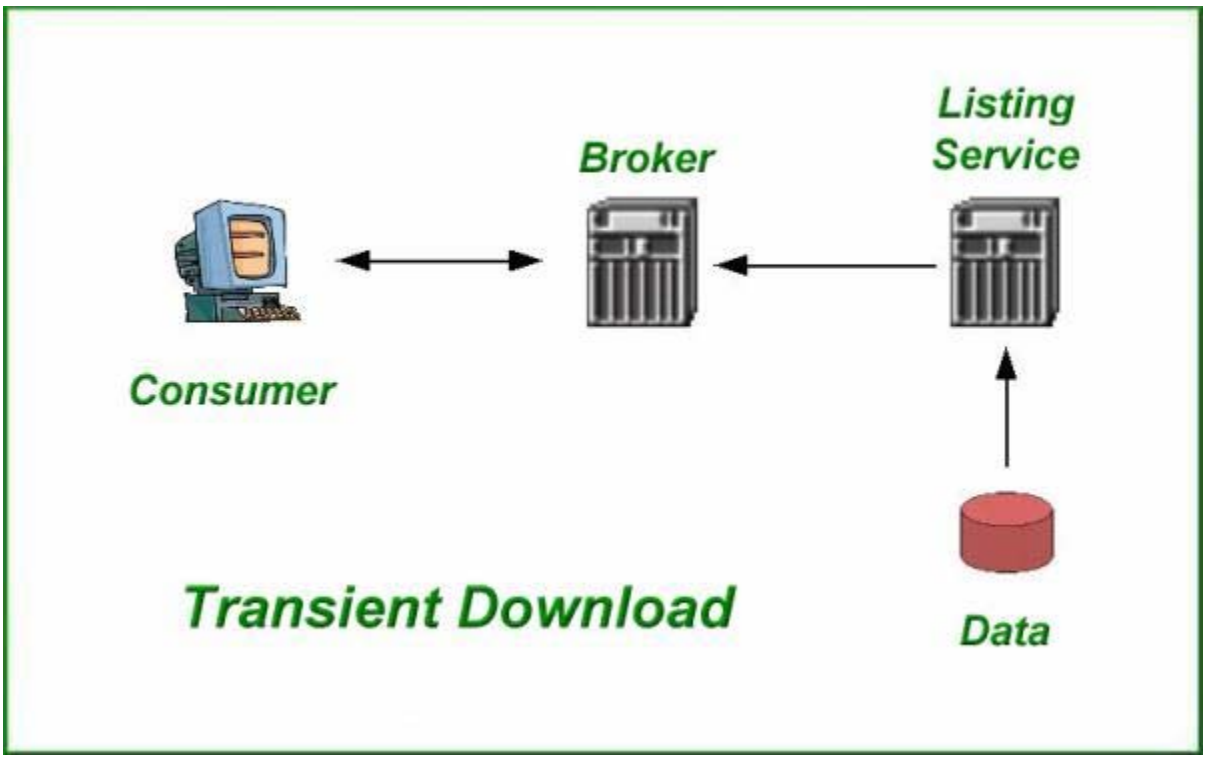

Figure 1 Retriever System Overview (CRT, 2004)

The Retriever software was built by a small team within CRT and was then placed on the CRT web site in January of 2003 (*SC 1). It contained a proxy server for accessing Listing Services and a parser for processing the data that is transmitted by the Listing Service (*SC 2). This technical approach makes it appear that a copy data is stored on the Broker's site even though this is not the case.

Nothing happened for a year. Apparently, the project was not important and valuable to a critical mass of users (*SC 4). In January of 2004, bug reports began arriving from unexpected sources. Two Application Service Providers (ASPs) had downloaded the software and were attempting to use it to provide services to multiple brokers. One of these providers was located in Ohio and the other was located in Arizona. Some bug reports even contained the source line number where the ASP thought the bug was located.

CRT immediately moved resources from other projects to work with the two ASPs (*SC 7, *RR 10). Bug reports and requests for new features arrived steadily. In early May, the CRT decided to implement the requested features (*RR 11). CRT further decided that the most effective way to implement the new features was to refactor the code. Refactoring involves moving logic from one object to another and may even involve changing data structures. In short, refactoring is very close to rewriting (*RR 3). For further information on refactoring, see *Refactoring at Berkeley: Improving code while containing costs* (Berkeley, 2004).

The new version with the improved code was released in July 2004. As a result of the new clarity in the code, the existing users began to implement new features themselves in order to speed up development. CRT's role changed from being the sole developer to being the integration manager. CRT attributes this success to providing very rapid bug fixes and to the willingness to refactor the code (Lesswing, 2004) (*SC 7).

Noscrape Project

Listing Services and Brokers (Realtors) have web sites that contain information in both image and text format about properties being sold. This information was copied from these sites by unauthorized third parties and stored on third party servers which the Realtors called "basement servers" to indicate that these machines were mostly run by individuals. The 1909 Realtor Code of Ethics requires that compensation be given for referrals and therefore, the operators of the "basement servers" demanded payment when customers located property using the "basement server".

Legally, if the data is marked proprietary, the operator of the "basement server" can be sued. However, when individual number 1 scraped the data from a Listing Service web page, removed the proprietary markings, and sold it to individual number 2, it proved difficult to make a case against individual number 2.

An outside consultant was hired by the National Association of Realtors to propose a solution to this problem. The solution was to use a digital rights management approach. This required that consumers download proprietary browser plugin software in order to access information about properties being sold. This plugin would have the side effect of preventing the user from saving to disk or copying data from the browser screen. The vendor of the plugin would also require a payment for each installation of the plugin.

CRT decided to develop an Open Source solution that would not involve installing any software on the customer's machine. This solution is shown in Figure 2. CRT built software written in Java to run on the Realtor's machine that would read data and images from the Realtor's relational database and image files and produce a single image. This image contains all the information that the Realtor would normally display on the web page including the Realtor's branding data and the Realtor's notice that the data is proprietary. The Realtor's web site then displays only a single image. If the image is scraped for use on a "basement server", there is no practical way to remove the branding information and the copyright notice. This solution was quickly developed, shown at a conference in March of 2004 and released in April.

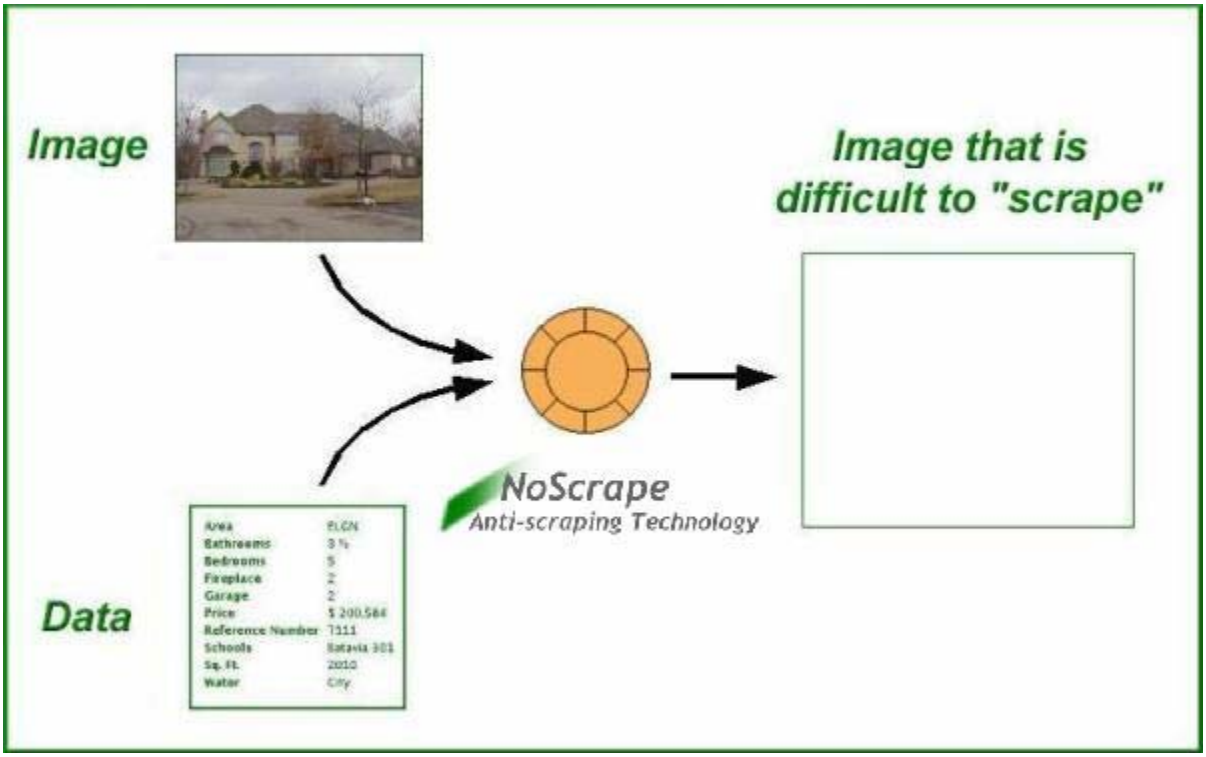

Figure 2 - Noscrape System Overview (CRT, 2004)

The Noscrape project is technically advanced. It uses the picture-in-picture facilities of Java. In fact, Java was selected because C# does not have comparable functions (*SC 2) (*RR 1). It was delivered quickly by CRT (*SC 1) (*SC 7). As in commercial software development, CRT obtained marketing input to establish the project name of Noscrape.

Unlike Retriever, this software was downloaded and tested immediately. Users in Michigan, Vermont and Utah discovered that links that were available on the web site before use of Noscrape were no longer active in the image produced by Noscrape. This was a serious problem because customers could see text inviting them to link to a Realtor's email but clicking on the text did not produce results. This problem was fixed in May 2004 and users are continuing to test the software and request new features. As of August 2004, the software is being piloted at the sites that reported the initial problems.

The documentation for Noscrape is of high quality and it was posted to the CRT web site almost a month ahead of the software itself. The visuals that were part of the documentation package were copied by consultants for use in their presentations to clients about the problem of appropriation of data from web sites. These same consultants learned to use the CRT Noscrape software and have been an important force in causing the software to be used.

RETS Server Project

In 1997, the Real Estate Industry began defining a standard to facilitate movement of data between Brokers and Listing Services. This standard was published in 2000 as the Real Estate Transaction Standard (RETS). The vendors who supplied software to the Listing Services declared that the standard was hard to implement and did not adopt it.

CRT suggested that a reference implementation of the standard be created and made available as Open Source software. This would prove that the standard was possible to implement. Vendors could then use code from the reference implementation in their products. Use of the standard would benefit the members of the National Association of Realtors and the project was approved.

CRT released the reference implementation of their RETS server in 2002. During the course of development, it was discovered that existing RETS client software on the market did not conform to the Real Estate Transaction Standard. Therefore, CRT built a new RETS client in order to test the RETS server. In the seven month period following the first release in May of 2002, a large team of contributors assisted in testing and debugging (*SC 7) (*SC 8). The code was refactored twice before the December 2002 release (*RR 3).

A Broker in Naples Florida put the CRT RETS server and client into production instead of buying software from a vendor. This saved a $150,000 software license fee. The fact that anyone would attempt to use a reference implementation as production software was a surprise (*RR 14). This lead to requests for production-oriented features such as support for relational databases. CRT implemented such support using Open Source Software from the Hibernate Organization (Hibernate, 2004) to connect the RETS Java code to different flavors of relational databases. These enhancements were released in late 2003.

In late 2003, CRT decided that they could get more input from users and vendors if the project were to be rearchitected to include a larger number of smaller components. Each component could then be managed separately. There was a desire on the part of some users to maintain the original architecture, so CRT arranged for another organization to take over the software as it existed at the end of 2003 (*RR 5).

CRT has built the new, smaller components and has released them throughout 2004.

CRT noticed that they had spent much time explaining the scope of RETS and how it worked. The felt that this was a partially a result of the low investment made in documentation before the software was released.

Conclusions

1. The three CRT projects discussed in this paper measure up well against the Open Source project success criteria.
2. The initial release of all three projects was used or marketed differently than CRT expected. CRT reacted to this effectively by quickly adding features to support the unexpected uses.
3. The level of initial documentation has greatly influenced how much time the developers need to spend explaining the scope and operation of the initial release to users and potential users.

The remainder of this section evaluates the three projects against the eight project success criteria.

Criteria 1 - Vision and Prototype

All three projects provided a strong initial vision and design and quickly delivered a prototype release.

Criteria 2 - Technically Cool (scratch the developer's itch to code)

Project Retriever used advanced PHP coding techniques, had a parser to process the data, and used low level HTTP code to implement a proxy server. Project Noscrape used complex Java library classes to create a picture within a picture. Project RETS has used state-of-the art techniques for making Java object persistent.

Criteria 3 - Use Tools

All of the CRT projects use source code control, bug tracking, and debugging tools.

Criteria 4 - General Community Service (seen as valuable)

All of the CRT projects were eventually seen as valuable by the community. The RETS project had a side effect of driving down the prices for RETS servers from about $150,000 to $20,000. This represents a significant cost savings for members of the National Association of Realtors. The Retriever project was not used for over one year. This would have been a problem for a commercial software company. However, because Retriever was Open Source and because CRT reallocated resources to support it when it started to be used, Retriever did provide a community service.

Criteria 5 - Developers Use the Product

CRT does not meet this success criteria because its internal developers are not Realtors and therefore do not use the CRT software directly. CRT does, however, keep demonstration versions of all its software available and running on CRT servers.

Criteria 6 - Have Committed Leaders

CRT has provided a committed leader for each project. The leader is committed to responding to questions and bug reports in a very timely manner.

Criteria 7 - Listen, Use Customers to Test, Release Often.

CRT excels in this area. In all three projects, they have quickly reacted to the initial users by responding to bug reports and adding features to cover uses that were not anticipated in the original project. In the RETS project, CRT listened to users who wanted to continue using the old architecture and arranged for another group to take over the support and enhancement of the software that was based on the old architecture. CRT then moved on to develop a new series of RETS components using a new architecture.

Criteria 8 - Lead without Coercion

CRT has modified features and designs to accommodate the desires of project contributors. This is one of the reasons that the early RETS and Retriever releases moved so quickly.

Recommendations

As can be seen from how CRT measures up to the project success criteria above, there are no major changes required in the way that CRT interacts with the Open Source community. Therefore, the recommendation is to continue to do those things that have made past projects successful.

1. Continue to produce good detailed documentation and presentations before the first software release in each project and keep the documentation up-to-date.
2. Continue to respond quickly to bug reports and enhancement requests even as the number of projects and number of supported components grows.

References

1. Apache Organization Site (2004). Apache Web Server - Windows binary download retrieved January 19, 2004 from http://httpd.apache.org/download.cgi.

2. Asay, Michael (2004), *What is the true Value of Source Code*, retrieved from the IT Manager's Journal Web Site on July 21, 2004 at http://management.itmanagersjournal.com/management/04/07/13/0639244.shtml?tid=85&tid=72

3. Berkeley Web Site. (2004). *Refactoring at Berkeley: Improving code while containing costs.* Retrieved September 19, 2004 from http://istpub.berkeley.edu:4201/bcc/Fall2004/refactoring.html.

4. Blechar, Michael J and Driver, Mark (2004, February 11). *Eclipse Is Targeting Growth Through Reorganization*. Retrieved August 27, 2004 from the Gartner Research Database through the Capella University Library at https://www.capella.edu/Gartner/GartnerIntraweb/research/119700/119746/119746.pdf

5. Brooks Jr., Frederick P. (1986). *The Mythical Man Month 25th Anniversary Edition*. Boston, MA: Addison-Wesley Professional.

6. CRT – Center for Realtor Technology. (2004). Information for Projects Retriever, Noscrape, and RETS Server retrieved on September 11 http://www.crt.realtors.org/projects/rets/retriever/index.html, http://www.crt.realtors.org/projects/noScrape/index.html, and http://www.crt.realtors.org/projects/rets/rex/ respectively.

7. Drakos, Nikos (2004, March 30). *Open-Source Team Collaboration Tools Not Up to Big Tasks Yet*. Retrieved August 27, 2004 from the Gartner Research Database through the Capella University Library at https://www.capella.edu/Gartner/GartnerIntraweb/research/120300/120323/120323.pdf

8. Freshmeat.net. List of recent Open Source releases retrieved from http://www.freshmeat.net on August 4, 2004.

9. Hibernate. (2004). Information on the Hibernate Open Source software retrieved on September 19 from http://hibernate.bluemars.net/.

10. Hissam, Scott; Weinstock, Charles B.; Plakosh, Daniel; and Asundi, Jayatirtha. (November, 2001). *Perspectives on Open Source Software* Retrieved on August 28, 2004 from the Software Engineering Institute Web Site at http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr019.pdf.

11. Lesswing, Mark. Personal Communications, July 20, August 5, and August 13, 2004.

12. MySQL Web Site (2004). MySQL Database Management Software – Windows binary retrieved January 19, 2004 from http://www.mysql.com/downloads/index.html.

13. PHP Network Site (2004). PHP Software – Windows binary download retrieved January 19, 2004 from http://www.php.net/downloads.php.

14. SourceForge.net. Count of projects retrieved from http://sourceforge.net/ on August 12, 2004.

15. Weber, Steven. (2004). *The Success of Open Source*. Cambridge, MA: Harvard University Press.

Appendix A – Raymond's Rules

These rules are taken from (Raymond, 1999, pp. 32-66.)

1. Every good work of software starts by scratching a developer's personal itch.
2. Good programmers know what to write. Great programmers know what to rewrite (and reuse).
3. "Plan to throw one away; you will anyhow" (Brooks, 1986).
4. If you have the right attitude, the right problems will find you.
5. When you lose interest in a program, your last duty is to hand it off to a competent successor.
6. Treating users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
7. Release early, release often and listen to your customers.
8. Given a large enough beta-tester and co-developer base, almost every problem will be characterized quickly and the fix obvious to someone.
9. Smart data structures and dumb code works a lot better than the other way around.
10. If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource.
11. The next best thing to having good ideas is recognizing good ideas from your users. Sometimes the latter is better.
12. Often the most striking and innovative solutions come from realizing that your concept of the problem was wrong.
13. Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away.
14. Any tool should be useful in the expected way, but a truly great tool lends itself to uses you never expected.
15. When writing gateway software of any kind, take pains to disturb the data stream as little as possible – and **NEVER** throw away information unless the recipient forces you to.

16. When your language is nowhere near Turing-Complete, Syntactic Sugar can be your friend.

17. A security system is only as secure as its secret. Beware of pseudo-secrets.

18. To solve an interesting problem, start by finding a problem that is interesting to you.

19. Provided the development coordinator has a medium at least as good as the Internet, and knows how to lead without coercion, many heads are inevitably better than one.