

Reverse Engineering C++ & Java: Practical vs. Legal

Renfred Teo

Computer Science

Research Department of Superpc.org

renfred@superpc.org

Abstract

The purpose of my research is to compare and contrast the practical aspects of software engineering with the legal aspects of software engineering. Reverse engineering has very practical uses and is very commonly practiced. This is very useful when it comes to creating computer software. There are many tools available to aid in Reverse Engineering C++ and Java, and tools for either platform can be quite easily tweaked to work for the other. Issues of legality and fair use cloud the field of reverse engineering. The laws concerning reverse engineering in the United States are more relaxed, but in Europe they are much stricter and much less is permitted. My research led me to the conclusion that Reverse Engineering can be used to create compatible or even competing software and hardware. However, because of legal issues it is necessary to use a “Chinese Wall” or clean room to do so

1. Introduction

The original purpose of my research was to evaluate different forms of reverse engineering. Specifically, the similarities and differences of reverse engineering between C++ and Java. However, I discovered that Reverse Engineering for both C++ and Java are very similar, due to similar syntax. This is true for C# as well. Usually, when a tool has been made to reverse engineer either Java or C++, it can be tweaked slightly to be used for the other platform. One of the current trends of research in Reverse Engineering is to generate diagrams based on code. Various tools have been made to generate UML diagrams from source code or even a running application. Primarily the current focus is on Java code, due to the current popularity of Java.

Sometimes a manufacturer may want to develop software or hardware that will be compatible and work with another company’s software or hardware. Reverse Engineering can be used to make compatible software or hardware, and it has found to be very effective in doing so. Sometimes, even competitive software or hardware can be manufactured using reverse engineering. Products may be made to offer the same functionality of a competitor’s product with knowledge gained by the use of reverse engineering the product. As I continued to do more and more research, I discovered legal issues concerning the use of reverse engineering. Considering how the legality

of reverse engineering is seldom discussed I decided that it would be refreshing to focus my research on the legal aspects of reverse engineering.

2. Reverse Engineering Principles

The practical uses of reverse engineering involve using a clean room or “Chinese Wall.” Sometimes, the opposite is done, but that can lead to problems. There all also many C++ and Java tools available for reverse engineering, and they have very practical uses. The purpose of reverse engineering is to figure out how a program works. Source code is not always available.

Often times a clean room, or “Chinese Wall,” is used in reverse engineering. However, this is not always the case. When dealing with Reverse Engineering, most people probably think about having direct access to the hardware or source code. This can aid some aspects of reverse engineering, but may cause other problems— especially concerning matters of legality. I discovered that [5] and [7] are good references for the basics of reverse engineering and [7] is especially good at summarizing matters of legality.

2.1 Clean Room or “Chinese Wall”

When a clean room or “Chinese Wall” is used, first one team is sent in to observe and describe what a product or software does. The team is supposed to

just describe what the product or software does without using any references to source code or hardware designs. The descriptions and only the descriptions are then transmitted through a “Chinese Wall” and given to another team in a clean room. In other words, the second team has no knowledge of the software code and no knowledge of the hardware design. This is to prevent any copying of code, which could constitute copyright infringement. Any code that ends up being identical would be very limited and merely coincidental due to the nature of the functionality. However, even this may not be enough when it comes to issues of patent rights. Also, the new Digital Millennium Copyright Act has added new restrictions.

2.2 Hardware Design and Source Code

The clean room is not always the case. Sometimes in reverse engineering the source is directly analyzed and used to make the new product. In the case of hardware, the chips may be stripped down and the internal circuits observed to see what the wires are and where they are connected. This would be more difficult in cases where the wires are laid vertically. However, even being able to see the layout of the chip makes reverse engineering of a chip more difficult even with full schematics of the chip, because seeing the wires and seeing where they are connected does not show voltages at each point. Every wire may have different voltages and it is not apparent just in seeing the wires themselves. Sometimes this issue is addressed by keeping the chip powered up and taking measurements of voltages for the wires as the chip is being stripped down. Now, all of this may seem very easy, but it is easy only in the case of software.

With software, the binary code can always be quickly converted to assembly language and decompiled from assembly language into C++ or Java code. All of this direct access to the design of the hardware or the source code can make it possible to end up with a finished product much faster and easier. However, there are matters of legality and intellectual property rights when such is done. Software and hardware pirates will often break the rules and use direct access to designs or software code. Their main purpose may just be to bypass security features and enable copying the software and hardware in a usable form without paying any rightful licensing fees.

2.3 Tools for C++, Java, and other languages

There are many tools that have been developed to aid in reverse engineering of Java and C++ code. Often

times they were developed to generate the UML diagrams from source code. A. Merdes and Dorsch implemented an interesting tool which analyses a Java program as it is running. By analyzing source code, you may notice when one class calls another class, but by analyzing the source code as it is running, you can see when the class that is called returns a message. This will help in creating sequence diagrams.

There is also research underway which is trying to overhaul the sequence diagrams into animated diagrams. This would make the sequence of events more clear. More information can be found from reference [1]. B. Tonella and Potrich’s research was on reverse engineering to derive UML Diagrams. However, instead of being from Java, it was from C++. Their research was done much earlier than those I had reviewed on Java. It was still one of the newer papers on reverse engineering in C++ I had found. This made it apparent that the current trend in reverse engineering is towards Java as opposed to the older C and C++ technologies.

I noticed an example output of a diagram looks like a UML 1 style class diagram that was generated from a tool named CERN. Only the public data items and methods were shown. For those interested in seeing more historic approaches in reverse engineering, they can refer to reference [2]. C. PINOT was a reverse engineering tool noted by Shi and Olsson as executing faster because its “recognition algorithms are hard-coded and compute common sub-patterns once” [9]. D. Cohen, Gil & Maman noted something very interesting in terms of portability between languages. Most of JTL could be tweaked to work with C# [6]. This helps to make it apparent that it is fairly easy to manipulate Tools designed for Java to work with other similar languages like C++ and C#. E. One of the things I found quite shocking was that there has been active research in COBOL. Cleve, Herard, and Hainaut discuss a tool used to reverse engineer system dependencies graphs from COBOL [10].

3. Reverse Engineering Legal Issues

Sometimes Reverse Engineering is used to detect the possibility of copyright or patent infringement. The laws concerning reverse engineering vary depending on what country you are in. At the heart of the laws are copyright and patent legislation. There are, however, limited cases where the use of reverse engineering is permitted. There have also been many legal disputes involving the use of reverse engineering.

3.1 Legislation

There are many laws concerning reverse engineering and they vary from country to country. Fair use doctrines and violations of copyrights and patents are the primary concerns. The Digital Millennium Copyright Act adds new restrictions on reverse engineering. For one, it disallows reverse engineering of security measures that are in place to prevent copyright infringement. Vicarious infringement is another concern because it means that parent companies and managers are responsible to make sure reverse engineering is neither used improperly nor done improperly. Root word of vicarious is from Latin. Vicarious infringement suggests “let the superior answer” [8].

- A. In the United States, it is legitimate to use reverse engineering to create compatible hardware or software. However, it must be limited to what is necessary just to make the hardware or software compatible— nothing more, nothing less. Reverse engineering may also be legitimately used to create competing software with similar functionality. However, code may not be copied from the original software, and the original circuit designs may not be copied from hardware. The use of the clean room or “Chinese Wall” environment that I previously described must be used for such cases.
- B. In Europe the laws are much stricter. Even simple things like font designs cannot be copied and can be considered copyright infringement. Usually reverse engineering is allowed only when it is licensed by the original company that produced the hardware or software. Fair use of reverse engineering may be considered only in cases where there is a monopoly. Just making compatible hardware or software is not sufficient reason to allow reverse engineering in Europe.

3.2 Legal Disputes

There have been many legal cases of reverse engineering. In some cases companies using reverse engineering have been found to have properly used the process and in other cases they have found to be in violation of the local legislation.

- A. *Blizzard v. Combs & Crittenden*— Ross Combs and Rob Crittenden reverse engineered a number of Blizzard Games. The new games connected to unofficial servers. This resulted in faster response times for the games. This project later came to be known as bnetd. However, this also meant that the reverse engineered games bypassed security-checking measures in the software that was used to make sure that every copy was properly licensed. This was a violation of the hotly disputed Digital Millennium Copyright Act. Blizzard was using its Battle.net site to detect copies of the software that were pirated. Games affected were along the lines of Diablo, Starcraft, and Warcraft. “The 8th Circuit Court of Appeals in St. Louis ruled Thursday that federal law— specifically, the Digital Millennium Copyright Act— disallows players from altering Blizzard games to link with servers other than the company’s official Battle.net site” [3].
- B. *ACLU v. N2H2* – The American Civil Liberties Union filed a lawsuit because they wanted the right to create circumvention software and publish circumvention techniques. The ACLU wanted Ben Edelman to be exempt from future lawsuits and be allowed to reverse engineer N2H2’s Internet filtering software. The ACLU feels that the Internet filtering software violates free speech. Richards Sterns, U.S. District Judge, wrote “There is no plausibly protected constitutional interest that... outweighs N2H2’s right to protect its copyrighted property from an invasive and destructive trespass” [4]. ie. The right to free speech is not enough to justify the violation of N2H2’s intellectual property rights.
- C. *NINTENDO v. ATARI* – Nintendo had implemented a security feature in its game to check for a valid game cartridge as well as for a valid console. ATARI supposedly used reverse engineering to duplicate both the parts of the security checks in the Nintendo game consoles. ATARI implemented the check for a valid game cartridge, which was determined by the courts as “fair use.” They also implemented the console-checking feature, which checked for a valid console. However, Nintendo had never implemented that feature in their

product. It was determined that ATARI had infringed upon Nintendo's copyright by going beyond what was necessary to make their cartridges compatible. ATARI also ended being found guilty of violating the patent rights of Nintendo, because Nintendo's interface was patented technology [5].

4. Conclusion

Reverse Engineering has been simplified by the availability of many tools to aid in the process. There is much that can be done with reverse engineering, and it can be very useful. Reverse engineering can be used to help make compatible software and hardware, or even competitive software. However, there is much that can be done which could very likely be illegal and should not be done. There is a necessity to use a clean room environment or "Chinese Wall" to insure that you are not violating another company's copyright. Also, there are issues of products protected by patents. Even if your software is developed using a clean room environment, it may still be in violation of patent rights. You must be licensed to use patented technology. Furthermore, the Digital Millennium Copyright Act prohibits the use of Reverse Engineering for the purpose of circumventing security features.

5. References

[1] M. Merdes, and D. Dorsch; "*Experiences with the development of a reverse engineering tool for UML sequence diagrams: a case study in modern Java development*;" Proceedings of the 4th international symposium on Principles and practice of programming in Java; 125-134, 2006, ACM Press.

[2] P. Tonella, and A. Potrich; "*Reverse Engineering of the UML Class Diagram from C++ Code in Presence of Weakly Typed Containers*;" 17th IEEE International Conference on Software Maintenance (ICSM'01), 376-385, 2001, IEEE Computer Society.

[3] D. Mcullagh; "*Blizzard wins lawsuit on video game hacking*;" CNET News.com; September 2, 2005; http://news.com.com/2100-1047_3-5845905.html

[4] D. Mcullagh; "*ACLU loses digital copyright battle*;" CNET News.com; April 9, 2003; <http://news.com.com/2100-1025-996245.html>

[5] D. Musker; "*Reverse Engineering*;" Protecting & Exploiting Intellectual Property in Electronics, IBC Conferences; 10 June 1998; http://www.jenkins-ip.com/serv/serv_6.htm

[6] T. Cohen, J. Gil, and I. Marman; "*JTL: the Java tools language*;" Proceedings of the 21st annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications; 89 - 108, 2006, ACM Press.

[7] M. Schwartz; "*Reverse-Engineering*;" Computerworld; November 12, 2001; <http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,65532,00.html>

[8] J. Tsai, L. Cranor, and S. Carver; "*Vicarious infringement creates a privacy ceiling*;" ACM Workshop On Digital Rights Management; 9-18, 2006, ACM Press

[9] N. Shi, and R. Olsson; "*Reverse Engineering of Design Patterns from Java Source Code*;" Proceedings of the 21st IEEE International Conference on Automated Software Engineering (ASE'06); 123-134, 2006, IEEE Computer Society.

[10] A. Cleve, J. Henrard, and J. Hainaut; "*Data Reverse Engineering using System Dependency Graphs*;" Proceedings of the 13th Working Conference on Reverse Engineering (WCRE'06); IEEE Computer Society.