**Handouts of Lecture 16 Professional Practices (IT)**

**Lecture Title: Intellectual Property Rights (Cont.)**

**Protections for Software**

**Software Copyrights**

The first software copyrights were applied for in 1964. The Copyright Office allowed the submitted computer programs to be registered, reasoning that a computer program is like a “how-to” book. The Copyright Act of 1976 explicitly recognizes that software can be copyrighted. When a piece of software gets copyright protection, what exactly is copyrighted? First, copyright protects the expression of an idea, not the idea itself. For example, suppose you develop a program for a relational database management system. You may be able to copyright your implementation of a relational database management system, but you cannot copyright the concept of using relational databases to store information. Second, copyright usually protects the object (executable) program, not the source program. Typically, the source code to a program is confidential; in other words, a trade secret of the enterprise that developed it. The company only distributes the object program to its customers. The copyright also protects the screen displays produced by the program as it executes. This is particularly valuable for the developers of video games.

**Violations of Software Copyrights**

The holder of a copyright has a right to control the distribution of the copyrighted material. Obviously, this includes making copies of the program. The definition of what it means to make a copy of a program is broad. Suppose you purchase a program stored on a CD. If you transfer a copy of the program from the CD to a hard disk, you are making a copy of it. If you execute the program, it is copied from the hard disk of the computer into its random access memory (RAM). This, too, is considered making a copy of the program. The standard licensing agreement that comes with a piece of commercial software allows the purchaser of the product to do both of the above-mentioned copying operations. However, doing any of the following actions without authorization of the copyright holder is a violation of copyright law:

1. Copying a program onto a CD to give or sell to someone else

2. Preloading a program onto the hard disk of a computer being sold

3. Distributing a program over the Internet Another kind of copyright violation can occur when a company attempts to create software that competes with an existing product. Two court cases illustrate a copyright infringement and fair use of another company’s product.

APPLE COMPUTER v. FRANKLIN COMPUTER

In the early 1980s, Franklin Computer Corp. manufactured the Franklin ACE to compete with the Apple II. The Franklin ACE was Apple II compatible, meaning that programs sold for the Apple II would run on the Franklin ACE without modification. In order to ensure compatibility, the Franklin ACE contained operating systems functions directly copied from a ROM on the Apple II. Apple Computer sued Franklin for infringing on its copyright. The US Court of Appeals for the Third Circuit ruled in favor of Apple Computer, establishing that object programs are copyrightable.

SEGA v. ACCOLADE

Video-game maker Accolade wanted to port some of its games to the Sega Genesis console. Sega did not make available a technical specification for the Genesis console, so Accolade disassembled the object code of a Sega game in order to determine how to interface a video game with the game console. Sega sued Accolade for infringing on its copyright. In 1992 the US Court of Appeals for the Ninth Circuit ruled in favor of Accolade, judging that Accolade’s actions constituted fair use of the software. It noted that Accolade had no other way of discerning the hardware interface and that the public would benefit from additional video games being available on the Genesis console.

**Safe Software Development**

Suppose a company needs to develop a software product that duplicates the functionality of a competitor’s product without violating the competitor’s copyright. For example, in the 1980s companies developing IBM-compatible computers needed to develop their own implementations of the BIOS (basic input/output system). A “clean room” software development strategy helps ensure a company’s software program does not duplicate any code in another company’s product. In this strategy two independent teams work on the project. The first team is responsible for determining how the competitor’s program works. It may access the program’s source code, if it is available. If it cannot get access to the source, it may disassemble the object code of the competitor’s product. It also reads the product’s user manuals and technical documentation. The first team produces a technical specification for the software product. The specification simply states how the product is supposed to function. It says nothing about how to implement the functionality. The second team is isolated from the first team. Members of this team have never seen any code or documentation from the competitor’s product. They rely solely on the technical specification to develop, code, and debug the software meeting the specification. By isolating the code developers from the competitor’s product, the company developing the competing product can demonstrate that its employees have not copied code, even unconsciously.

**Software Patents**

Since then, hundreds of thousands of software patents have been granted. Microsoft alone files about 3,000 patent applications every year. Companies generate revenue by licensing their software patents to other companies. It’s also common for several technology companies to hold patents that cover different but essential components of a commercial product. By signing an agreement to cross-license each other’s patents, all of the companies are free to bring their own versions of the product to market. Given the value of software patents, it’s not surprising that a secondary market for them has arisen. For example, when a company holding patents goes bankrupt, its patents are sold to another company. Some companies specialize in holding patents and licensing the rights to use these patents. Patent-holding companies aggressively use the courts to enforce their patent rights; these companies are sometimes referred to as patent trolls. Because defending against a patent infringement lawsuit can easily exceed a million dollars, companies that get sued have a strong motivation to simply settle out of court, putting patent trolls “in a position to negotiate licensing fees that are grossly out of alignment with their contribution to the alleged infringer’s product or services”. In 1992 inventor Thomas Campana and lawyer Donald Stout formed New Technologies Products (NTP), a patent-holding company. The purpose of the company was never to make anything but to protect valuable intellectual property. About half of the company’s 50 patents were originally held by Telefind Corporation, which went out of business. In 2000 NTP sent letters to several companies, warning them that they were infringing on NTP wireless email patents and inviting them to negotiate licensing rights. One of these letters went to Research In Motion (RIM), maker of the BlackBerry, but RIM did not respond to the letter. The next year NTP sued RIM for patent infringement. Instead of settling out of court for a few million dollars, RIM took the case to trial and lost. After more unsuccessful legal maneuvering, RIM in 2006 agreed to pay NTP $612.5 million to settle the patent infringement dispute.

Critics of software patents argue that too many software patents have been granted. A problem faced by patent examiners in the Patent and Trademark Office is knowing what the existing technical knowledge (prior art) in computer programming is. Patent examiners typically look at patents already issued to determine prior art. This works fine for other kinds of inventions, but it doesn’t work well for software patents because a significant amount of software was written before software patents were first granted. The consequence is that patent examiners issue many “bad patents”—patents that would not have been issued if the examiner knew about all of the prior art. The Patent Office has also been criticized for granting patents for trivial inventions that would be obvious to any skilled computer programmer. As a consequence of the extremely large number of software patents, the large number of bad patents, and the number of obvious software inventions that are patented, any company releasing a new product that includes software runs a significant risk of being sued for infringing a software patent owned by someone else. Thousands of patent lawsuits are filed in the United States every year. Large corporations are resorting to building stockpiles of their own patents, so that if they are sued for infringing another company’s patent, they can retaliate with their own patent infringement countersuit. The use of software patents as legal weapons is a perversion of their original purpose. Some opponents of the current software patent system maintain that patent protection is inappropriate for software, which is less expensive to produce and has a much shorter useful life than other patentable properties, such as new pharmaceutical drugs. Jeff Bezos, CEO of Amazon.com, has suggested that software patents should have a life span of only three to five years.

**Key Differences between Software Copyrights and Software Patents**



**Open-Source Software**

In the 1970s the number of computer applications expanded, and organizations recognized the increasing value of software. To protect their investments in software development, most companies decided to make their programs proprietary (owned). Today companies developing proprietary software tightly control the distribution of their intellectual property. Typically they do this by treating source code as a trade secret and distributing only the object code, which is not in human-readable form. In addition, they do not sell the object code. Instead, when people “purchase” the program, what they are actually buying is a license allowing them to run the program. Their rights to do other things with the code, such as make backup copies, are limited.

**Consequences of Proprietary Software**

Governments have given ownership rights to those who produce computer software because of the perceived beneficial consequences. A key benefit is the ability to profit from the licensing of the software. The assumption is that people will work harder and be more creative if they must compete with others to produce the best product. Those who produce the best products will have the opportunity to make money from them. While most people point to the benefits of a system encouraging the development of proprietary software, some people have noted the harms caused by such a system. A well-known critic of proprietary software is Richard Stallman. According to Stallman, granting intellectual property rights to creators of computer software has numerous harmful consequences:

The copyright system was designed for an era in which it was difficult to create copies. Digital technology has made copying trivial. In order to enforce copyrights in the digital age, increasingly harsh measures are being taken. These measures infringe on our liberties.

The purpose of the copyright system is to promote progress, not to make authors wealthy. Copyrights are not promoting progress in the computer software field.

 It is wrong to allow someone to “own” a piece of intellectual property. Granting someone this ownership forces the users of a piece of intellectual property to choose between respecting ownership rights and helping their friends. When this happens, the correct action is clear. If a friend asks you for a copy of a proprietary program, you would be wrong to refuse your friend. “Cooperation is more important than copyright”. The open-source movement is the philosophical position that source code to software ought to be freely distributed and that people should be encouraged to examine and improve each other’s code. The open-source software movement promotes a cooperative model of software development.

**Open-Source Definition**

Open source is an alternative way of distributing software. Licenses for open-source programs have the following key characteristics (there are others):

1. There are no restrictions preventing others from selling or giving away the software.

2. The source code to the program must be included in the distribution or easily available by other means (such as downloadable from the Internet).

3. There are no restrictions preventing people from modifying the source code, and derived works can be distributed according to the same license terms as the original program.

4. There are no restrictions regarding how people can use the software.

5. These rights apply to everyone receiving redistributions of the software without the need for additional licensing agreements.

6. The license cannot put restrictions on other software that is part of the same distribution. For example, a program’s open-source license cannot require all of the other programs on the CD to be open source.

**Beneficial Consequences of Open-Source Software**

Advocates of open-source software describe five beneficial consequences of open-source licensing.

The first benefit of open source is that it gives everyone using a program the opportunity to improve it. People can fix bugs, add enhancements, or adapt the program for entirely new uses. Software evolves more quickly when more people are working on it.

Rapid evolution of open-source software leads to the second benefit: new versions of open-source programs appear much more frequently than new versions of commercial programs. Users of open-source programs do not have to wait as long for bug fixes and patches.

A third benefit of open source is that it eliminates the tension between obeying copyright law and helping others. Suppose you legally purchased a traditional license to use a program and your friend asks you for a copy. You must choose between helping your friend and conforming to the license agreement. If the program had an open-source license, you would be free to distribute copies of it to anyone who wanted it.

The fourth benefit is that open-source programs are the property of the entire user community, not just a single vendor. If a vendor selling a proprietary program decides not to invest in further improvements to it, the user community is stuck. In contrast, a user community with access to the source code to a program may continue its development indefinitely.

The fifth benefit of open source is that it shifts the focus from manufacturing to service, which can result in customers getting better support for their software. If source code were distributed freely, companies would make money by providing support, and the companies that provided the best support would be rewarded in the marketplace.

***Reference***

***Lecture 16 slides: Intellectual Property Rights(cont.)***

***Gao, Y. (2012). Ethics for the Information Age by Michael J. Quinn. World Libraries, 20(1).***