LEGAL ISSUES IN GLOBAL CONTEXTS:
Perspectives on Technical Communication in an International Age

Edited by

Kirk St.Amant
East Carolina University

and

Martine Courant Rife
Lansing Community College

Baywood's Technical Communications Series
Series Editor: Charles H. Sides

Baywood Publishing Company, Inc.
AMITYVILLE, NEW YORK

Baywood Publishing Company, Inc.

26 Austin Avenue
P.O. Box 337
Amityville, NY 11701
(800) 638-7819

E-mail: baywood@baywood.com
Web site: baywood.com

Library of Congress Catalog Number: 2013036928
ISBN: 978-0-89503-835-7 (cloth : alk. paper)
ISBN: 978-0-89503-836-4 (paper)
ISBN: 978-0-89503-837-1 (e-pub)
ISBN: 978-0-89503-838-8 (e-pdf)
http://dx.doi.org/10.2190/LII

Library of Congress Cataloging-in-Publication Data

Legal issues in global contexts : perspectives on technical communication in an international age / edited by Kirk St.Amant, East Carolina University and Martine Courant Rife, Lansing Community College.
pages cm. -- (Baywood's technical communications series)
Includes bibliographical references and index.
ISBN 978-0-89503-835-7 (cloth : alk. paper) -- ISBN 978-0-89503-836-4 (pbk. : alk. paper) -- ISBN 978-0-89503-837-1 (e-pub) -- ISBN 978-0-89503-838-8 (e-pdf) 1. Computers--Law and legislation--United States. 2. Intellectual property--United States. 3. Information technology--Law and legislation--United States. 4. Technological innovations--Law and legislation--United States. 5. Law and globalization. 6. Technology and law. I. St.Amant, Kirk, 1970- editor of compilation. II. Rife, Martine Courant, editor of compilation.
KF390. 5. C6L445 2013
343.09'99--dc23

2013036928

Table of Contents

FOREWORD

Considering Legal Issues in Global Contexts

TyAnna Herrington

It is not news that online media have shrunk the world and made it possible for technical communicators living in different nations and regions to collaborate on a range of activities. This kind of global reach allows projects to broaden in terms of who may participate in them and how. This reach has also allowed businesses to expand from local or regional operations to become participants in a growing global marketplace. Such opportunities for collaboration, however, require care in approaching workplace interactions, for with new contexts come new challenges, both anticipated and unexpected. In such situations, the more participants know about the variables affecting international interactions, the better they can anticipate certain problems or respond to others.

In expectation that cultural divergence could arise as a key factor affecting international exchanges, technical communicators need to prepare for potential conflict resulting from such differences. They must also prepare themselves to expect such situations to arise and consider strategies for negotiating the developments that could result from cultural difference. Fortunately, much has been made of this kind of preparation in the published technical communication research over the last decade. And these works have done an effective job of examining how issues of linguistic, operational, and cultural difference can create a variety of challenges that technical communicators might face when working in international venues. Yet within these discussions, the range of valuable publications on global interaction has not been augmented by an extensive treatment of the legal issues that can shape international interactions until now. Such legal issues, however, could be seen as one of the remaining barriers to affect the sharing of ideas and information in the modern global village.

http://dx.doi.org/10.2190/LIIC9

CHAPTER 9

Software Patent Law in Global Contexts: A Primer for Technical Writing Specialists

Annette Vee

The work and personal lives of technical communicators are often conducted through the medium of software, and so the intellectual property (IP) laws that concern software's composition and international distribution affect the way we write and live. Copyright is the IP regime most often discussed in the field, but patent law has a potentially greater effect on composition in, of, and with software because the monopoly rights granted through patents can be more powerful than the distribution rights afforded by copyright. Patent protection for software has become more readily available through recent court decisions, international treaties, and changes to guidelines for patent examiners in the United States, European Union, and Japan. Software patents, when coupled with our accelerating reliance on both desktop and web-based software, are impacting our lives as writers and teachers.

Several cases illustrate the way that increased patent protection for software has threatened our unfettered access to digital writing and teaching environments. In the high-profile patent case over online content management systems (CMS), *Blackboard v. Desire2Learn* (2009), Blackboard was poised to have monopolistic rights over basic interface features of CMSs before the settlement for the suit created oligarchic rights instead. In another sphere of digital composition—mobile phones—nearly every hardware and software producer is suing every other one (Bilton, 2010). This legal thicket precariously positions what is now both a primary composition space for young people (Lenhart, Arafah, Smith, & Rankin Macgill, 2008) and an increasingly important one for higher education (Johnson, Levine, & Smith, 2009). Most dramatically, the very existence of the

World Wide Web was threatened by a suit from British Telecom, which claimed the company owned all rights to the hyperlink (Jaffe & Lerner, 2004).

Global debates on software patent law are fierce and commensurate with their significant implications for software's composition and distribution. However, the fact that international implementations of software patent law can be complex and arcane may be why it is so seldom addressed in the literature on technical communication or in writing studies more generally. Here, I hope to translate some of the current and global issues in software patents into contexts relevant to writing specialists in order to help us have a better understanding of the ways patent law, especially for software, affects our lives. I begin by outlining some specific reasons patent law pertains to writing specialists, including a more detailed explanation of several patent law cases and an exploration of the blurring boundaries between written text and computer code. I then give an overview of patent law: its relationship to copyright, the ostensible objective of patent protection for intellectual property, what is generally required to obtain a patent, and variations in international patent law regarding software. Finally, I review some of the controversies surrounding patent protection for software, namely, its *de facto* favoring of corporations over independent software developers. Software patents can affect our teaching and composition processes, but they also give us a way of reflecting on our profession as writers and understanding the increasing proximity of our work to the work of computer programming.

HOW SOFTWARE PATENT LAW AFFECTS WRITING SPECIALISTS

The patenting of software directly affects the choices of digital tools we have at our disposal as teachers and writers. Moreover, the disappearing boundaries between writing and coding that patent law implies can lend us insight into our evolving writing practices. Below, I explain in more detail why writing studies must include software patents in our discussions of IP protection.

Direct Effects of Software Patents on Writing and Teaching

Patents directly impact the software environments in which we compose. In a 1991 article arguing against software patents, Garfinkel, Stallman, and Kapor note that XYWrite, a popular word processor at the time, had to eliminate certain features in order to appease another company that had patented them. More recently, OpenOffice has been threatened by Microsoft for infringement of multiple patents (Parloff, 2007). The thicket of patent suits that surround mobile devices (wherein a large part of everyday writing now occurs) is enough to make anyone's head spin (e.g., Microsoft vs. Motorola; Research in Motion vs. Motorola; Apple vs. HTC). Some patents lie dormant (so-called submarine

patents), ready to be used against competitors should they be needed to protect the company's business plan. In short, the spaces in which we write for work and play are constrained by software patents and their corresponding lawsuits.

For teachers, this impact is doubled, since software patents affect digital environments for teaching as well as composition. For example, Washington, DC-based Blackboard, the company that makes a popular CMS used at many schools and universities throughout the world, has been involved in at least two major software patent lawsuits, one as plaintiff and one as defendant. In 2006, Blackboard sued the Canadian company Desire2Learn in Texas for infringement of a patent entitled "Internet-based education support system and methods" (USPTO #6,988,138). The patent claimed an educational content management system that would allow multiple users access to multiple courses under different identities (e.g., as instructor, student, teaching assistant).

Despite widespread criticism of the breadth of the patent and educators' attempts to collect information that might lead the patent to be found invalid (e.g., Wikipedia, n.d.), the Eastern Texas district court decided in favor of Blackboard, awarding it $3.1 million in damages (*Blackboard v. Desire2Learn*, 2007). In December 2009, during Desire2Learn's appeal of the verdict, the companies decided to settle their dispute by cross-licensing their patent portfolios (Baker, 2009). The terms of the settlement were undisclosed, as they generally are in cases such as this one, which means that Desire2Learn ceased to challenge the patent and Blackboard's patent still stands.

Continuing the CMS patent wars, TechRadium sued Blackboard in 2008 for infringing a patent that covered a software-implemented system allowing a message to be delivered to various devices when sent from a single, centralized source. This suit also settled privately out of court. As with Blackboard's CMS patent, TechRadium's patent seems quite broad. Moreover, it was originally implemented in the context of emergency notification, yet it potentially extended to *all* software-implemented messaging services that work across different electronic devices. TechRadium also sued the popular online messaging service Twitter for infringement of the same patent in 2009 (Schonfeld, 2009).

These legal battles are fought at great cost to the companies involved. But whether or not they are named in the suit, universities, users of programs, and open-source implementations of software are also caught in the fray. Blackboard's patent suit against Desire2Learn threatened diversity in the CMS market, but it also endangers open-source CMSs such as Sakai and Moodle, as well as any CMS developed locally by universities. To attempt to prevent suits from Blackboard against open-source and university-developed CMS technology, EDUCAUSE banded together with universities. In response to this pressure, Blackboard pledged to not use their patent against noncommercial CMSs, thereby assuaging a rapidly escalating public relations nightmare with their main clients (Carnevale, 2007). While this pledge offers some comfort to those who wish to use a non-Blackboard CMS, the pledge is nonbinding—it does not negate

the patent itself nor Blackboard's right to sue anyone implementing a system that might infringe it. The mere existence of a broad-sweeping patent so central to online learning is likely to discourage innovation in this educational sector.

If a multiple-login CMS system such as Blackboard's can be patented, what other methods and programs that we use to teach might be patentable? How might this affect the ways that we use technology to augment education? The vast number of software and methods patents, the high variability in how they are treated across the globe, and the increasing popularity of distance education makes the answers to these questions impossible to determine but important to explore.

Disappearing Boundaries between Writing and Coding

Writing online or with software documentation is common in professional writing; we write with and for websites, APIs (application programming interfaces), and online spaces such as Twitter and Facebook. Many technical writers are also scripters or programmers for these interfaces. As the line between composing software and composing more traditional texts is growing fainter, the legal fate of software may hold keys to the fate of writing as well.

In 2000, the use of the hyperlink—perhaps the clearest example of hybrid text/code common in technical writing—was threatened when British Telecom sued Prodigy for infringement of the patent they claimed to hold on it (Jaffe & Lerner, 2004). Since the hyperlink is a basic building block for the World Wide Web, British Telecom's patent threatened all uses of writing on the web. The suit was finally thrown out, but as Jaffe and Lerner (2004) argue, "What is so scary about this case is that British Telecom's assessment of the current state of the patent landscape gave it sufficient hope of prevailing so as to make this case worth bringing" (p. 58).

Even when professional writers are not coding alongside writing, we often work closely with programmers and in similar compositional contexts as they do. Oracle's 2010 purchase of Sun, the originator of the open-source language Java, alarmed many employed and freelance programmers because of Oracle's far more aggressive reputation for prosecuting their IP (e.g., Kuhn, 2010). Sun had pledged not to enforce their patents on Java, allowing programmers to improve the language through open-source methods; however, Oracle is not bound by Sun's promises to the open-source Java community. Software (much of it written in Java) defines the delivery of our work and shapes its composition; the fate of our profession may be mirrored in the fate of programmers alongside whom we work.

The blurred line between software and technical writing extends to teaching as well. Alfred Essa (2007) warns that software patents will become increasingly relevant as well as restricting as higher education shifts teaching toward online spaces, for

> As the world becomes increasingly digital, more educational technology tools become commonplace, as more faculty utilize cyberspace for some part of their teaching, and as the boundary between teaching in the traditional classroom and teaching online begins to blur, our intuitions about property, collaboration, and academic freedom will be put to the test. (p. 72)

From his perspective as Associate Vice Chancellor and Deputy CIO for the Minnesota State Colleges and Universities, Essa sees higher education under attack from politics, for-profit institutions, and decreased public funding. "Information technology is one of the few strategic levers we possess for controlling costs and improving quality," he argues; yet "the specter of software patents has now placed that strategic lever at risk." The fact that software—processes embodied in texts—can be patented suggests that even teaching methods could be subject to patents.

OVERVIEW OF PATENT LAW

Now that I have outlined specific reasons for professional writers to take note of trends in patent protection for software, I will explain a bit of the history and nature of patent law to protect intellectual property, especially as it pertains to software. I focus primarily on the United States and the European Union; along with Japan, these jurisdictions account for the vast majority of software patents issued worldwide (Park, 2005). This explication is not designed to be comprehensive, but it provides sufficient background to understand various issues in patent law that might affect technical communication and writing studies more generally.

Patent Law vs. Copyright Law

Since copyright law is likely to be more familiar to professional writers, we can look at patent law's relationship to copyright to understand how it functions as intellectual property protection. Copyright protects original expressions and is an especially important form of protection for writers, musicians, and artists. Patent rights protect "inventions" that are functional and are more central to the work of engineers, architects, and chemists.

Distinctions between function (generally protected by patents) and expression (generally protected by copyrights) have proven very difficult for courts to determine. For example, in *Brandir International, Inc. v. Cascade Pacific Lumber Co.* (1987), at issue was whether an elegantly designed yet functional bicycle rack could be afforded copyright protection. The Court decided that because its function could not be fully separated from its aesthetic form, the "Ribbon Rack" was not afforded copyright protection. For writing, however, copyright law has been able to draw a distinction between expression and function. In the landmark case *Feist Publications v. Rural Telephone Service* (1991), the U.S. Supreme

Court asserted that the mere collection of facts was below the bar of "original expression" in U.S. copyright law. In contrast to the phone book under discussion in *Feist*, instruction manuals, photo captions, cooking directions, and business information on websites are all subject to copyright because they involve a kind of creative expression.

Both copyrights and patent rights are routinely assigned to people or institutions that are not the creators; pharmaceutical companies often own the patents on the work of their chemists, and publishers often own the copyright on the writing they publish. While patents generally bar anyone from manufacturing or re-creating an invention without the patent holder's permission, copyrights prevent others from copying the copyright holder's work. To infringe a copyright, then, a person must have had access to the original work to copy it. Since proof of patent infringement does not require that the alleged infringer had access to the original invention, however, a person can violate a patent of which she is unaware.

Like the Ribbon Rack, software is a difficult nexus of function and expression because it is both (aesthetic) text and (functional) virtual machine. Software is composed of source code, a specialized kind of text written by programmers to direct the computer's processes. Source code is writing that turns a computer into a virtual, specialized machine, and so it can be thought of as a text, a machine, or both. When framed as a text, source code is subject to copyright protection. When framed as a machine, it is subject to patent protection. Whether it is in the form of object code (the code that the machine can read) or compiled code (the human-readable language in which it is written) or whether it is in an interpreted programming language (i.e., one that does not require compiling), source code is assumed to be subject to copyright. This presumption of copyright protection for source code is global, as various international treaties such as the Berne Convention harmonize copyright protection. The presumption of patent protection for source code, or the software that is constructed from it, is more controversial and is not uniformly implemented through international IP law.

What is Patentable?

A patent is a government-sanctioned, limited monopoly on the manufacture or practice of a particular invention. Specific requirements that an invention must meet to be patentable are up to individual nations to determine, subject to some international treaties and organizations, but common requirements include specific subject matter, novelty, usefulness, and disclosure. To obtain a patent, an inventor must apply to a national authority that grants patents. If it meets a nation's particular requirements, she can obtain a patent and exercise her limited monopoly for the time specified by that nation.

The monopoly right of a patent is limited by time: at least 20 years for all signatories to TRIPS (Agreement on Trade-Related Aspects of Intellectual

Property Rights), an international treaty governing patents and that includes the United States, Europe, Japan, and all other World Trade Organization (WTO) countries. Theoretically, patents encourage innovation by allowing the inventor to recoup her investment in researching and developing the invention. Yet the disclosure of the invention (i.e., the full description of it) allows others to learn from its development and eventually to practice or make the invention once the patent expires. This exchange of information is central to the economic theories of patent law.

Requirements for Patent Claims

Subject Matter

Subject matter available for patenting differs across nations, especially for software. "Inventions patentable" is defined by 35 U.S.C. §101 as "Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefore, subject to the conditions and requirements" specified in subsequent Sections defining the conditions of novelty, nonobviousness, and disclosure. The European Patent Organization (EPO), which provides guidelines for EU patents, outlines what is and is not subject to patent protection: "European patents shall be granted for any inventions, in all fields of technology, provided that they are new, involve an inventive step and are susceptible of industrial application." Explicitly excluded as inventions are "rules and methods for performing mental acts, playing games or doing business, and programs for computers" (European Patent Convention, 2000). Japanese patent law states that an invention is patentable if it is "a highly advanced creation of technical ideas utilizing natural laws" (as cited in Park, 2005). Game rules, economic principles, and human mental activities are not patentable in Japan because they rely on human situations rather than natural laws (Park, 2005).

As suggested in the terms *technology* (Europe), *invents or discovers* (U.S.), and *creation* (Japan), patentable subject matter generally excludes anything already found in nature or anything not invented by humans. What is *natural,* however, can be difficult to define. Medicines derived from plants, genes (when created or processed by humans), and applied mathematical algorithms have been eligible for patenting worldwide, although they are all arguably found in nature.

Software is explicitly excluded from patentable subject matter in Article 52, 2(c) of the European Patent Commission; however, the EPO does issue patents on "software-related inventions" (Nard, Barnes, & Madison, 2005, p. 759). Jinseok Park (2005) provided cases that illustrate the distinctions the EPO makes between "software as such" and "software-related inventions." For example, an EU court asserted, "If the program controls the operation of a conventional general-purpose machine so as technically to alter its functioning, the unit consisting of program

and computer combined may be a patentable invention" (*Koch & Sterzel* T 0026/86-3.4.1, as cited in Park, p. 338). Additionally, in *Vicom*, the Boards declared that a mathematical method, when directed at a technical process, was patentable (EPO Decision T 0208/84-3.5.1, as cited in Park, p. 338). The Japanese Patent Office (JPO) treats software similarly to the EPO (Park, 2005).

Compared to the JPO and EPO, the USPTO has been more liberal in issuing patents on software. This is especially true since *State Street Bank and Trust v. Signature Financial Group* (1998), when the U.S. Court of Appeals for the Federal Circuit (CAFC) lifted the case law exemption for the patentability of business methods. Prior to that, patents on software were issued by the USPTO, but they had to be embodied in a machine and their status was more ambiguous. *State Street* indicated unambiguously that computer programs *per se* were patentable as long as they were useful. As a 2000 EU study on the patentability of computer programs states,

> The real difference between the USA and Europe is that in Europe the invention has to be of a technical character whilst in the USA the mere fact that the invention uses a computer/software makes it of the technological arts, if also useful, concrete and tangible results are provided. (Hart, Holmes, & Reid, 2000, p. 2)

However, following the *State Street* decision in the United States, the EPO began to issue many more patents on business practices as well as data processing and equipment (Blind, Edler, & Friedewald, 2005). These disparities in patentable subject matter mean that computer-related inventions are protected differently across nations despite the fact that many are globally distributed through the Internet.

Novelty and Nonobviousness

To meet the novelty and nonobvious requirements for patenting, inventions generally must demonstrate some kind of innovation; one cannot patent a wheel, for instance, since it is technology widely known and used. However, a particular type of wheel, constructed in a way to serve a unique purpose, may be eligible for a patent if it involves "an inventive step" (see TRIPS, outlined below). If the step is not "inventive," then the invention might be considered obvious in light of available technology and therefore ineligible for patenting.

The novelty requirement for U.S. patents is specified in 35 U.S.C. §102(a): if, previous to the patent-applicant's invention, "the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country" then it is not novel. In other words, the United States does not consider foreign inventions to be prior art if they are not patented or described in print. In contrast, public foreign knowledge *is* treated as prior art by the EPC (Article 54(2)) and Japan Patent Law (Section 29(1)) (Nard et al., 2006, p. 700). This treatment of foreign prior art means that U.S. law allows for U.S. corporations to patent such things as foreign indigenous plant remedies and profit from them in the United States, sometimes called "biopiracy."[1]

Utility

The utility requirement of patents is expressed by 35 U.S.C. §101 ("useful") and Article 54 EPC ("industrial applicability"). Seldom does the lack of an invention's utility bar its patenting because patent offices and courts often assume that utility can be more clearly determined by market forces than by law. When utility does emerge as a challenge to a patent, it tends to be in the field of biotechnology, when a company attempts to patent something that appears to be a preliminary stage of a research trajectory (Nard et al., 2006).

The court deciding the case *Brenner v. Manson* (1966) articulated one value of the utility requirement: the *quid pro quo* patents provide between inventors and society through the sharing of useful technology (Nard et al., 2006). As the *Brenner* court argued, technology should not be patented prior to its moment of utility because a patent on an unfinished process "may confer power to block off whole areas of scientific development, without compensating benefit to the public" (as cited in Nard et al., 2006, p. 737). The hamstringing of innovation that the *Brenner* court warns about may be especially pertinent to software composition because of its rapid iteration and development cycle.

Disclosure

The *quid pro quo* of patent law is most clearly enacted in the disclosure requirement. In exchange for a patent, an inventor must explain how to make or practice the invention in the patent's "specification." Companies naturally prefer to be as circumspect as possible when describing their inventions in order to keep their innovations from their competitors. Yet without the full description of the invention, the patent holder would receive a monopoly right without contributing to the public good. Consequently, nations generally specify the level of detail required to obtain a patent. For example, the United States requires enough detail for a "person having ordinary skill in the art" of the field (the PHOSITA) to reproduce the invention.

The disclosure requirement has been oddly applied in the case of software in the United States, and some critics suggest that too many junk patents on

[1] Nard et al. (2006) described a 1990s case in which a multinational company, W. R. Grace, patented products made from the Indian neem tree. These derivations of the neem tree were widely used by people in India, but they had not been patented. When Indian NGOs challenged Grace's patents, the EU patent was invalidated for lack of novelty, but the U.S. patent stood, allowing W. R. Grace to continue their government-granted monopoly on Indian folk uses of the neem tree within the United States (p. 700).

computer programs have been issued in part because the specification requirements are low. One of these critics, Ben Klemens (2006), describes several software patents in which the requisite "written description" includes only vague flow charts rather than the code that would pin down the methods more concretely. In *Northern Telecom vs. Datapoint Corp* (1990), Northern Telecom showed evidence that it was not necessary to reduce their software invention to code in their patent specification because their written description differed only slightly from the actual code that enacted it. The Federal Appeals Court agreed, suggesting that the translation of software design into code is merely a technicality (Burk & Lemley, 2005).

The U.S. Supreme Court says that software is "ready for patenting" once a commercial order is placed and the rough outlines of the software are drawn, even if the code is never written or does not work (Burk & Lemley, 2005). However, as technical communicators can attest, the difference between an abstract or general description of a proposed text and the actual text can be significant. Through this lens, U.S. patents on software come dangerously close to patents on *ideas*, which are not allowed in any nation.

Patent Law Jurisdiction

Jurisdiction for patent law is at the national level. For example, the United States stipulates its own requirements for issuing patents, the USPTO issues patents, and the U.S. court system handles disputes about U.S. patents. Since a patent is only enforceable within national boundaries, a person could not be barred from making a U.S.-patented invention in Japan and selling it there unless a Japanese patent is also obtained for the invention. In the EU, member states issue their own patents, but the EPO also offers patents and provides guidelines to member states. If an inventor obtains an EU patent, she must have that patent validated by each member state in which she would like protection.

Organizations such as the WTO (World Trade Organization) and WIPO (World Intellectual Property Organization) and treaties such as TRIPS (Agreement on Trade-Related Aspects of Intellectual Property Rights) stipulate how patents must be honored beyond national boundaries. Because TRIPS is attached to the WTO, enforcement of the agreement is possible through trade sanctions (Mueller, 2009). TRIPS codifies requirements for patenting: "Patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive step and are capable of industrial application" (as cited in Mueller, 2009, p. 544). Countries that wish to become WTO members and TRIPS signatories must have patent laws that reflect this requirement, although they have leeway in how they interpret that clause. TRIPS explicitly mentions that copyright law pertains to computer programs, and it calls the computer programs "literary works," to be treated as specified by the Berne Convention, which pertains to copyrights. Therefore,

computer code is almost universally covered by copyright law, but its coverage under patent law varies across nations.

Numerous minor agreements impact patent law as well. One example is the Paris Convention, which harmonizes the file date for patents; if an inventor files for a patent in Japan, then files in the United States, he gets the earlier Japanese filing date for review in terms of novelty and nonobviousness (Mueller, 2009). Language barriers, complex enforcement of international treaties, and differences in patentable subject matter and requirements across nations can make the process of enforcing patents internationally much more complex and expensive than enforcing them nationally, which is already very difficult without the help of patent attorneys.

Shifts in International Patent Law

Intellectual property laws have generally been stronger and more established in Europe and the United States than in other countries. Beginning in the 19th century, as trade and transportation accelerated communication across the globe, IP law radiated out from the centers of industry and began taking on an international scope. For example, the Berne Convention harmonized copyright for literary and artistic works among many nations in 1886 (although the United States did not sign on until 1988). In the last few decades, IP law harmonization through international treaties has tended to increase protection throughout the globe. TRIPS was negotiated through the WTO in 1994 and is largely the result of pressure from more industrialized, IP-producing nations like the United States, Japan, and members of the EU.

Further evidence of global trade and efforts to harmonize patent law to facilitate it can be found in the Five IP Offices working group. The Five IP Offices is a group organized to streamline processes of patent filing in the largest patent offices in the world: Japan, Korea, China, Europe, and the United States, which together account for 90% of patent applications filed worldwide (Five IP Offices, n.d.). The Five IP Offices, whose heads met in 2008, claim they want to streamline their applications because "as the world sees economic barriers between nations fade away, innovators want their intellectual creations protected concurrently in multiple major markets" (Five IP Offices, n.d.). The USPTO and the EPO are also currently working on a joint classification system that will make it easier for companies to file for patents in both the EU and the United States (USPTO and EPO work, 2010). This international focus on trade is corroborated by patent law scholar Janice Mueller (2009), who noted that international patent law legislation in the 1980s shifted from the WIPO, which concentrated on IP, to the WTO, which covers international trade more generally and seeks to lower trade barriers between nations.

Although the focus of this chapter is not on international trade relations, it should be noted that lowered trade barriers often imply imbalances in power

between industrialized and developing nations. To illustrate, language about what is patentable under TRIPS was aimed at less-developed countries who were refusing to grant patents on agricultural and pharmaceutical inventions because monopolies on those inventions increased their price, sometimes prohibitively (Mueller, 2009). Under TRIPS, some compulsory licensing is allowed for desperately needed pharmaceuticals and agricultural technologies, but only with certain conditions met: a national emergency or extreme situation must exist; reasonable terms for a consensual license are not available from the patent holder; and the patented invention must be for use only in the domestic market (Mueller, 2009). This ethical issue of technology transfer between industrialized and developing nations is also addressed indirectly in Article 52 EPC, which provides an exception to patenting to protect "*ordre public*" or "morality," suggesting that such technology as highly productive varieties of grain or AIDS drugs should be made widely available.[2]

In the United States, the general trend toward increased copyright protection has moved in parallel with increased patent protection. Although legislation has driven increased copyright protection (e.g., the Digital Millennium Copyright Act), most of the changes related to software patents in the United States have occurred through shifts in policies at the USPTO and major decisions in the courts. For example, the Court of Appeals for the Federal Circuit (CAFC) was established in the early 1980s to streamline the patent appeal process, and it has interpreted patent law to make it easier to get patents, easier to enforce patents, easier to get large financial awards from such enforcement, and harder for those accused of infringing patents to challenge a patent's validity (Jaffe & Lerner, 2004). Application fees for patents finance the USPTO and also provide a surplus of discretionary funds that Congress uses to fill budget shortfalls elsewhere. This budgetary reliance on patent fees translates to governmental and USPTO support of patent issuance. As Jaffe and Lerner (2004) explain, the concern about American competitiveness in the global marketplace in the 1980s led Congress to push policies that turned the USPTO and patent system into more of a business than a government organization.

Software Patent Law

When software first became important to businesses in the 1960s, the prevailing opinion among intellectual property scholars in the United States was that

[2] However, Nard et al. (2006) noted that despite some member states' arguments along morality lines, the EU has allowed patents on human DNA (e.g., the hormone Relaxin, used during childbirth). Moreover, the United States, the origin of many of these technologies and patents, does not have a stipulation for "*ordre public*" or "morality" as part of their patent code. The work of Vandana Shiva (2001) further illustrates how patents held by large international companies such as Monsanto can cripple agriculture in developing countries.

software, algorithms, and business methods were too abstract or too much like pure math to be patentable (Merges, 1999). President Johnson assembled a special commission of academics, scientists, and industry representatives in order to review the USPTO's stance on patenting software; the commission ultimately recommended that software should not be patentable (Samuelson, 1990). Court cases in the 1970s affirmed the idea that computer programs were too abstract to be patentable (*Gottschalk v. Benson*, 1972; *Parker v. Flook*, 1978). The Freeman-Walter-Abele test, the case law standard for determining patentability of computer-related inventions at that time, indicated that an invention claiming a mathematical algorithm had to have that algorithm applied to a physical process in order to be patentable.

Patent cases in the 1980s began to chip away at the special exemptions for algorithms and business methods. *Diamond v. Diehr* (1981) indicated that although an algorithm *per se* was not subject to patent, a specific application of an algorithm in an industrial process was potentially patentable. After this case, patent claims emphasized the "industrial process;" they began to use the phrase, "a general-purpose computer on which is programmed a method to calculate" rather than "a method to calculate" (Klemens, 2006, p. 45). *In re Alappat* (1994) further undermined the software patent exemption. In that case, the CAFC wrote, "We have held that such programming creates a new machine, because a general purpose computer in effect becomes a special purpose computer once it is programmed to perform particular functions pursuant to instructions from the program software" (as cited in Klemens, 2006, p. 58). In other words, a new software program *legally created* a new physical machine.

The case that removed all doubt about the patentability of software in the United States was *State St. Bank and Trust Co. v. Signature Financial Group, Inc* (1998). In *State Street*, the court unambiguously declared that the "so-called 'business method exemption'" to patentability was invalid. In considering patentability, the emphasis should not be on subject matter (specified in 35 U.S.C. §101) but on the result: as long as a process has a "useful, concrete and tangible result," it is patentable subject matter (as cited in Nard et al., 2005, p. 757). Because software was considered a kind of business method, a flood of applications for software patents inundated the USPTO after *State Street*.

Software is still considered patentable subject matter in the United States, although more recently, the U.S. Supreme Court rejected the "useful, concrete and tangible result" test in favor of the "machine-or-transformation" test, which states that a process claim is eligible for patenting "if: (1) it is tied to a particular machine or apparatus, or (2) it transforms a particular article into a different state or thing" (*Bilski v. Kappos*, 2010, p. 1). Although critics of software patents had pinned hopes on the Supreme Court's review of *Bilski*, the Court's decision in *Bilski* was narrow, and they "decline[d] to adopt a broad exclusion over software or any other such category of subject matter beyond the exclusion of claims drawn to fundamental principles set forth by the Supreme Court" (p. 21 n. 3).

Other countries can resist the U.S. model of increased software patent protection because TRIPS does not force signatories to consider software patentable subject matter. For instance, New Zealand was considering a ban on software patents (Heras, 2010). Moreover, the EU policy is currently applied heterogeneously across member states, since each has the power to grant its own patents and implement its own interpretation of EU patents. Support for software patents is stronger in the United Kingdom than in other member states, for instance.

A commission was formed in 2002 to review the EU policy on software patents and consider whether the U.S. approach would be economically beneficial. This commission specifically considered the impact of patents on open-source software, which contributes substantially to the technological economy in the EU (Blind et al., 2005). The commission was also charged with the goal of homogenizing support for software patents across EU member-states in order to lower some of the financial and bureaucratic barriers to technological business in the EU. During the review process, software patents were hotly debated among technology experts, industry veterans, and intellectual property scholars. Across the globe, industries, lawyers, and programmers watched with interest as Europeans considered whether to adopt a U.S.-style system or to continue to exempt software from patentable subject matter. The final vote in the EU Parliament rejected the patentability of software programs *per se* (Patentability of computer-implemented inventions, 2005).

In summary, patents are generally awarded for human inventions that are useful, nonobvious, and genuinely new, although the specific treatment of software as patentable subject matter varies worldwide. Like copyright law, patent law is subject to interpretation by individual nations, but it is affected by numerous international treaties. Intellectual property law regarding software is particularly susceptible to international laws because its location is often hard to pin down to a particular national jurisdiction. And yet software patents affect all users of software worldwide. In the final part of this chapter, I consider what the controversies of software patent law have to do with writing studies specifically.

CONTROVERSIES SURROUNDING PATENT PROTECTION FOR SOFTWARE

Given the intimate connections between writing and software, we should be concerned with the balance of legal protection awarded to software. Legal protection for intellectual property is designed to stimulate new production of ideas and technology for the benefit of society. Yet when protection is too strong, it can work instead to keep ideas and technology out of the hands of the public. As Lawrence Lessig (2002) has argued, stronger intellectual property protection does not necessarily lead to greater opportunities for innovation. If patent protection encourages innovation and diversity in software, then we should

encourage it; conversely, if patent protection constrains our choices in teaching and composing in software, then we should be wary of that protection. Unfortunately, the benefits and drawbacks of patent protection of software are not straightforward or clear. With a goal of helping us to understand this ideal balance to encourage innovation, this section looks at the arguments and evidence surrounding patent protection for software.

Opposition to Patents by Software Writers

Patent protection was designed within a paradigm of industrial production—highly centralized development with significant investment in research and development. Because of software's unique development profile, many developers argue against any kind of patent protection for it. They argue that patents are not as good a fit for software as they are for industrial technology; they can tie up ideas or bracket off potential innovation more so than in other fields. In contrast to biotechnology, engineering, and industrial production, the field of software production often has low start-up costs and rapid development and iteration. An independent software developer can produce patentable technology using no resources other than a computer. This also means that the independent software developer working on her home computer can *violate* patents, most of which are held by companies with significant legal resources such as IBM and Microsoft.

Blind et al. (2005) showed that although some companies had benefited from a software patent, this benefit was not seen across firms on average. Moreover, firms that did not use software patents expressed uneasiness that their actions were restricted by software patents. The authors indicate that the differences in software development—shorter development cycles, frequent new products, and need for interoperability—make patents less beneficial for that field. Other dangers of patent protection peculiar to software include the collaborative nature of software composition, the difficulty of searching for prior art (Blind et al., 2005), and the sheer volume of code written for software (Garfinkel et al., 1991).

Perhaps the most vocal and long-standing critic of software patents has been Richard Stallman, author of the popular General Public License (GPL), which is used prominently in open-source software development. In an article he coauthored with Simson Garfinkel and Mitchell Kapor (1991), Stallman argued that the USPTO's standards for awarding software patents was too low and that (even before *State Street*) the flood of patents on "ideas so elementary that they could have been answers to problems in a first-year programming course" (p. 51) did real harm to the innovation in software. The authors note that large companies such as IBM, Lotus, and Microsoft became successful without software patents. Lawrence Lessig (2002) points out the odd fact that patent protection for software in the United States came not from arguments by the software industry but from the courts; major software companies Adobe and Oracle both came out

against software patents in 1994 because they did not see them as useful incentives to software innovation.

Computer game pioneer and id Software founder John Carmack, who often releases game code under Stallman's GPL license, has repeatedly refused to patent his own programming techniques because he believes that patents on software slow innovation and creativity (Carmack, 1997). Other entities who release free software under Stallman's GPL license—many of them universities or individuals—cannot afford the risk of patent infringement. Software patents also do not reduce trade secrecy, which is one aim of the patent system, because the disclosure requirement for software is relatively lax (Garfinkel et al., 1991). Many developers argue that the intellectual property laws of copyright and trade secret protection fits the development profile much better than patent protection.

Opposition to software patents is particularly strong among independent software developers; in a survey of European software developers, independent developers were much less likely than larger software companies to support stronger and more globally binding patent law (Blind et al., 2005). They believe software should be excluded from patenting, and nearly all surveyed strongly disagree that Europe should move to a model such as the United States has, with patentability for software as such (Blind et al., 2005, p. 87). Tech-centered Internet forums such as Slashdot also demonstrate the popular sentiment against software patents among individual developers; see, for instance, the community's uproar over the patent-supporting brief that IBM submitted to the U.S. Supreme Court during the *Bilski* trial (IBM's Supreme Court Brief, 2009).

The Problems of Patent Protection for Independent Software Development

The composition and distribution practices of independent software developers are part of why they are more likely to oppose software patents. An extensive survey conducted in 2008 by prominent U.S. patent law scholars revealed that especially for software start-ups, "patents provide relatively weak incentives for core activities in the innovation process, such as invention and commercialization" (Graham, Sichelman, Samuelson, & Merges, 2009, p. 1261). As a result, small software companies are much less likely to patent their technology. On the other hand, companies that consider the core part of their business to be product innovation (those that mainly make hardware or physical technology) were twice as likely to indicate that patenting was "important in capturing competitive advantage" (p. 1293). Software companies, even those who make software "products," or packaged software for sale, are less likely to see patents as central to their competitive advantage; instead, "first-mover advantage" is much more important (Graham et al., 2009, p. 1293).

Blind et al. (2005) found similar tendencies against patenting for smaller software firms in Europe.[3] They showed that the tendency to patent increases as the size of the corporation increases. An increased research budget did not correlate with higher patenting for software, implying that the availability of patents for software does not promote research investment. Like Graham et al. (2009), Blind et al. (2005) also found that the tendency to patent was higher in hardware-centered corporations, showing that the patenting profile of hardware-plus-software firms followed the higher-patent model of hardware rather than the lower-patent model of pure software firms.

Noting that smaller firms lack expertise in patent law and have more limited budgets, Blind et al. (2005) indicate that these firms have "neither the awareness nor the resources to make effective use of patenting" (p. 19). Indeed, Graham et al. (2009) show that small software firms are more sensitive to the costs of patenting and prosecution. They cite that "the average out-of-pocket cost for a respondent firm to acquire its most recent patent was over $38,000," and smaller firms not only have fewer monetary resources, they also need to get out-of-house counsel, which is more expensive and harder for them to monitor (Graham et al., 2009, p. 1311). Lessig (2002) points out that the marginal costs for smaller firms to patent software is much greater than it is for firms with much larger budgets.

Compounding the problem that independent developers have lower *de facto* access to patents because of their lack of resources, independent developers tend to be disproportionally affected by software patents. Law regarding patent infringement does not discriminate between large corporations and individuals and so the lack of resources among independent developers can lead not only to their lack of patents but also to their greater vulnerability to patent infringement suits. As a U.S. court stated in *Lough v. Brunswick Corp.* (1996), "The law does not waive statutory requirements for inventors of lesser sophistication" (as cited in Nard et al., 2006, p. 685). In other words, everyone must play the game the patent lawyers play. In fact, independent developers are significantly more likely to have a project impeded, go up in price, or prolonged because of software patents, and the threat of these effects mean that they are less likely to start a project in the first place (Blind et al., 2005). They may also be more likely to hide their code or keep their business small to stay under the radar of patent-infringement suits (Vee, 2010).

Open-source software development is even more vulnerable to harm from software patents. Because other incentives exist for open-source software development, patents do not do much to encourage innovation in that sector, yet the development process is still subject to patent infringement (Blind et al., 2005; Vee, 2010). Even if an open-source software group wanted to patent their work,

[3] Although patents on software per se are not supported in Europe, this study focused on *software-related* patents.

their marginal cost of obtaining a patent would be magnified because budgets in open source are generally nonexistent. Plus, the organization of open-source development is so decentralized that a coordinated effort toward patenting is nearly impossible (Lessig, 2002). Many commercial developers and other industries rely on open-source software, so the negative effect of software patents on open-source development bleeds out into the commercial sector as well (Blind et al., 2005). A dearth of economic studies on the impact of open-source software means that we cannot know the full extent of the damage.

In a global context, the negative effect of software patents on small-scale, independent, and open-source development could be accentuated. The patchwork of regulations makes it difficult to navigate patents internationally; firms with less access to monetary and legal resources will find it much more difficult to patent their technology in international markets and yet will be more vulnerable to the threat of patent infringement lawsuits. International patent law allows for mandatory licensing on such technologies as pharmaceuticals for health, but this can reify the consumer position of developing countries. Although he does not address patent law directly, an extension of Arjun Sengupta's (2004) argument for a "human right to development," based on a 1986 United Nations Declaration, reveals a need for international support of technology in developing countries. One could argue that this support could encompass differential treatment of software composition in these countries to allow safe haven for native developers who accidentally infringe code on international software distributions. Otherwise, patent protection for software can discriminate against less powerful countries and contribute to the status quo in differential distribution of IP power.

The Balancing Act for Software Patents

Software patents may not support innovation. Yet as Samuelson (2007) argued, the dam has already burst; too much software located in powerful companies is now protected by patents for the courts or Congress to rescind patent protection for software. Burk and Lemley (2005) also see software patents as a now-permanent fixture in IP protection in the United States, and patent authority Robert Merges, author of an oft-cited 1999 critique of software patents, has more recently admitted that we are in a period of "normalization" of software patents (Merges, 2007)—a period in which they are simply part of the industry and not as destructive to its development as he had once argued. Allison and Hunter (2006) point out that even before the *State Street* decision in the United States, patents were issued to software simply because the lawyers had tied the software to a specific "machine" in their written description. Europe, despite a specific statement against the patenting of software *per se*, still issues patents for software (Blind et al., 2005; Nard et al., 2006). Therefore, patents on software are probably here to stay.

Given the inevitability of software patents, Burk and Lemley (2005) suggested that we should have a different approach to patent protection for software than we do for high-investment industries. They argue that a greater availability to more narrowly written patents might protect smaller firms from infringement and allow them to more easily acquire patents. Merges (1999) noted that we do not know for certain if software patents are bad or good for innovation, but we can all agree that bad patents are clearly bad; therefore, we should have higher scrutiny in business method patents, the category in which most software patents fall.

However, differential scrutiny of patents is simpler in theory than in practice. Allison and Hunter (2006) indicated that any special scrutiny of software patents will likely result in more circuitous written descriptions. They based this assertion on a review of a 2000 USPTO policy that implemented a "second pair of eyes review" (SPER) for business methods in an attempt to stanch the flood of these patents after *State Street*—a policy very similar to the one that Merges suggested. The USPTO reviewed only patents with business methods as a *primary* category, however, and so patent applications then spilled into related categories without a more thorough review (Allison & Hunter, 2006). As Allison and Hunter (2006) argued, definitional issues hamper special review; firms will circumvent the special regulations for software and simply file in a different category for a patent.

After the U.S. Supreme Court *Bilski* decision, it appears that any reform in U.S. patent law will need to come from Congress rather than the courts. Here is the key "passing the buck" passage in the U.S. Supreme Court's opinion:

> It is important to emphasize that the Court today is not commenting on the patentability of any particular invention, let alone holding that any of the above-mentioned technologies from the Information Age should or should not receive patent protection. This Age puts the possibility of innovation in the hands of more people and raises new difficulties for the patent law. With ever more people trying to innovate and thus seeking patent protections for their inventions, the patent law faces a great challenge in striking the balance between protecting inventors and not granting monopolies over procedures that others would discover by independent, creative application of general principles. Nothing in this opinion should be read to take a position on where that balance ought to be struck. (*Bilski v. Kappos*, 2010, pp. 9–10)

As the above passage suggests, patent law has become more difficult to understand at the same time that innovation is possible by a wider population. This theoretical balance in patent law to which the U.S. Supreme Court refers is struck differently across the globe, and it is perpetually inflected by international treaties, national legal codes, guidelines for patent examiners, and case law.

CONCLUSION

Patent protection for software can be a problem for writing researchers interested in the advancement of technology to support writing practices. Open-source alternatives to patented programs like Blackboard can be hamstrung by the intellectual property fences erected by companies flush with lawyers and other resources. Patent-supported monopolies can limit our freedom of expression and the means through which we distribute or teach writing. What are writing specialists to do? I argue that we should at least be aware of the relevance of patent law on the work we do.

Patent law is now part of the system of legal regulations for writing—the ways that writing is industrialized, constrained, and protected by law. Although patents were tested in the industrial economy and are often thought to be restricted to that economy, this is no longer true. Patents have come to affect the composition environments of software writers and technical writers for software—workers in the modern knowledge economy. As writers and writing researchers, our professional connection with computers draws us into the realm of patent law through the controversial and various patent protections for software across the globe. Writing studies can no longer bracket off patent law as irrelevant to our interests.

On a conceptual level, the application of patent law to software code—a kind of writing—should serve as a sign for us to reconceive our relationship to computers and the power of writing. Current applications of patent law can demonstrate just how fully integrated writing and coding are now; software can be thought of as a machine with physical actions in the world, but this specialized "machine" is a product of specialized writing. Patent law demonstrates how tricky it is to theorize software as a kind of writing *or* as a kind of technology; as a kind of expression *or* as a kind of action. It is all of these simultaneously, which opens up a deeper debate not only about how software and writing should be protected by IP law but also about how we should conceptualize the writing work we do.

REFERENCES

Allison, J. R., & Hunter, S. D. (2006). On the feasibility of improving patent quality one technology at a time: The case of business methods. *Berkeley Technology Law Journal, 21,* 729–794. Retrieved from http://www.btlj.org/data/articles/21_02_02.pdf

Baker, J. (2009, December 15). Blackboard, Desire2Learn announce patent cross license agreement and settlement of litigation. *Desire2Learn.* Retrieved from http://www.desire2learn.com/patent/

Bilski et al. v. Kappos. 561 U.S. (2010). *Slip opinion.* Retrieved from http://www.supremecourt.gov/opinions/09pdf/08-964.pdf

Bilton, N. (2010). An explosion of mobile patent lawsuits. *New York Times.* Retrieved from http://bits.blogs.nytimes.com/2010/03/04/an-explosion-of-mobile-patent-lawsuits/

Blackboard, Inc. v. Desire2Learn, Inc. 521 F.Supp.2d 575 (2007).

Blackboard, Inc. v. Desire2Learn, Inc. 574 F.3d 1371 (2009).

Blind, K., Edler, J., & Friedewald, M. (2005). *Software patents: Economic impacts and policy implications.* Cheltenham, UK: Edward Elgar.

Burk, D. L., & Lemley, M. A. (2005). Designing optimal software patents. In R. Hahn (Ed.), *Intellectual property rights in frontier industries* (pp. 81–108). Washington, DC: AEI-Brookings Joint Center for Regulatory Studies.

Brandir International, Inc. v. Cascade Pacific Lumber Co. 834 F.2nd 1142 (2d Cir 1987).

Brenner v. Manson. 383 U.S. 519 (1966).

Carmack, J. (1997, November 25). John Carmack—The Boot interview (with Alex St. John). *The John Carmack Archive.* Retrieved from http://www.team5150.com/~andrew/carmack/johnc_intervie_1997_John_Carmack_The_Boot Interview.html

Carnevale, D. (2007). Blackboard pledges to ease enforcement of controversial patent. *The Chronicle of Higher Education.* Retrieved from http://chronicle.com/article/Blackboard-Pledges-to-Ease/28490/

Diamond v. Diehr. 450 U.S. 175 (1981).

Essa, A. (2007). Software patents: Why should we care? *EDUCAUSE Review, 42*(2), 72–73. Retrieved from http://www.educause.edu/EDUCAUSE+Review/EDUCAUSEReviewMagazineVolume42/SoftwarePatentsWhyShouldWeCare/158127

European Patent Convention. (2000). *Article 52. Patentable inventions.* Retrieved from http://www.epo.org/law-practice/legal-texts/html/epc/2010/e/ar52.html

Feist Publications v. Rural Telephone Service. 499 U.S. 340 (1991).

Five IP Offices. (n.d.). *Five IP offices.* Retrieved from http://www.fiveipoffices.org/

Garfinkel, S., Stallman, R., & Kapor, M. (1991). Why patents are bad for software. *Issues in Science and Technology,* 50–55.

Gottschalk v. Benson. 409 U.S. 63 (1972).

Graham, S., Sichelman, T., Samuelson, P., & Merges, R. P. (2009). High technology entrepreneurs and the patent system: Results of the 2008 Berkeley Patent Survey. *Berkeley Technology Law Journal, 24,* 1255–1328. Retrieved from http://www.btlj.org/data/articles/24_feature.pdf

Hart, R., Holmes, P., & Reid, J. (2000). Study contract ETD/99/B5-3000/E/106: The economic impact of patentability of computer programs. *Intellectual Property Institute.* Retrieved from http://ec.europa.eu/internal_market/indprop/docs/comp/study_en.pdf

Heras, K. (2010). New Zealand to ban software patents. *The Next Web.* Retrieved from http://thenextweb.com/au/2010/07/17/new-zealand-to-ban-software-patents/

IBM's Supreme Court brief says that patents drive free software. (2009). *Slashdot.* Retrieved from http://yro.slashdot.org/story/09/09/03/1616220/IBMs-Supreme-Court-Brief-Says-That-Patents-Drive-Free-Software

In re Alappat. 33 F.3d 1526 (1994).

Jaffe, A. B., & Lerner, J. (2004). *Innovation and its discontents.* Princeton, NJ: Princeton University Press.

Johnson, L., Levine, A., & Smith, R. (2009). *The 2009 Horizon report.* Austin, TX: New Media Consortium.

Klemens, B. (2006). *Math you can't use.* Washington, DC: Brookings Institution.

Kuhn, B. (2010, August 16). *Considerations for FLOSS hackers about Oracle vs. Google* [Blog post]. Retrieved from http://ebb.org/bkuhn/blog/2010/08/16/oracle-google.html

Lenhart, A., Arafah, S., Smith, A., & Rankin Macgill, A. (2008). *Writing, technology and teens*. Washington, DC: Pew Internet and American Life Project.

Lessig, L. (2002). *The future of ideas: The fate of the commons in a connected world*. New York, NY: Vintage.

Lough v. Brunswick Corp. 86 F.3rd 1113 (1996).

Merges, R. P. (1999). As many as six impossible patents before breakfast: Property rights for business concepts and patent system reform. *Berkeley Technology Law Journal, 14*. Retrieved from http://www.law.berkeley.edu/journals/btlj/articles/vol14/Merges/html/reader.html

Merges, R. P. (2007). Software and patent scope: A report from the middle innings. *Texas Law Journal, 85,* 1627–1676.

Mueller, J. M. (2009). *Patent law* (3rd ed.). New York, NY: Aspen.

Nard, C. A., Barnes, D. W., & Madison, M. J. (2006). *The law of intellectual property*. New York, NY: Aspen.

Northern Telecom vs. Datapoint Corp. 908 F.2nd 931 (1990).

Park, J. (2005). Has patentable subject matter been expanded? A comparative study on software patent practices in the European Patent Office, the United States Patent and Trademark Office and the Japanese Patent Office. *International Journal of Law and Information Technology, 13*(3), 336–377.

Parker v. Flook. 437 U.S. 584 (1978).

Parloff, R. (2007). Microsoft takes on the free world: Microsoft claims software like Linux violates its patents. *CNNMoney.com/Fortune Magazine.* Retrieved from http://ec.europa.eu/internal_market/indprop/comp/index_en.htm

Patentability of computer-implemented inventions. (2005, July 6). *European Commission.* Retrieved from http://ec.europa.eu/internal_market/indprop/comp/index_en.htm

Samuelson, P. (1990). Benson revisited: The case against patent protection for algorithms and other computer program-related inventions. *Emory Law Journal, 39,* 1025–1154.

Samuelson, P. (2007). Software patents and the metaphysics of Section 271(f). *Communications of the ACM, 50*(6), 15–19.

Schonfeld, E. (2009). Here come the Twitter patent lawsuits. *TechCrunch.* Retrieved from http://www.techcrunch.com/2009/08/05/here-come-the-twitter-patent-lawsuits-techradium-?les-the-?rst-one/

Sengupta, A. (2004). The human right to development. *Oxford Development Studies, 32*(2), 179–203.

Shiva, V. (2001). *Patents: Myths and reality*. New Delhi, India: Penguin.

State Street Bank and Trust v. Signature Financial Group. 149 F.3d 1368, 1375 (1998).

USPTO and EPO work toward joint classification system. (2010, October 25). *European Patent Office.* Retrieved from http://www.epo.org/topics/news/2010/20101025.html

Vee, A. (2010). Carving up the commons: How software patents are impacting our digital environments. *Computers and Composition, 27*(3), 179–192.

Wikipedia. (n.d.). *History of virtual learning environments.* Retrieved from http://en.wikipedia.org/wiki/History_of_virtual_learning_environments

http://dx.doi.org/10.2190/LIIC10

CHAPTER 10

Orphan Works and the Global Interplay of Democracy, Copyright, and Access

Martin S. Copenhaver

Egalitarian access to information is a core ideal of a democratic society. Recognizing the fundamental right to participate in the ongoing cultural, social, and political dialogue, the framers of the U.S. Constitution ensured this access by providing that "Congress shall have the Power . . . To promote the Progress of Science and useful Arts, by securing for limited Times to authors and Inventors the exclusive Right to their respective Writings and Discoveries" (Art. 1, Sec. 8, Cl. 8, hereinafter the "copyright clause"). This notion of egalitarian access to information is crucially important because access facilitates the exchange of ideas through an ongoing dialogue that enables democracy itself (Herrington, 2011).

As both users and creators of copyrighted materials, technical communicators experience this importance of access to information, copyrighted or otherwise, in their day-to-day activities. On the one hand, technical communicators in academia use this access to copyrighted works to comment on and critique various communicative structures. On the other hand, technical communicators working in industry access these materials in order to create new communication products (e.g., user manuals and online help systems) used by a range of clients or consumers. In both cases, technical communicators must obtain the appropriate permissions to use copyrighted works to create new materials should the intended uses exceed the allowable exceptions to or limitations of copyright. The rise of digital media, however, has created new contexts that can hinder these processes, and one such situation involves orphan works, or works for which the copyright holder cannot be identified and located.