INCORPORATING THE COMMONS: A POLITICAL ECONOMIC ANALYSIS OF CORPORATE INVOLVEMENT IN FREE AND OPEN SOURCE SOFTWARE

by

BENJAMIN J. BIRKINBINE

A DISSERTATION

Presented to the School of Journalism and Communication
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

September 2014

DISSERTATION APPROVAL PAGE

Student: Benjamin J. Birkinbine

Title: Incorporating the Commons: A Political Economic Analysis of Corporate Involvement in Free and Open Source Software

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the School of Journalism and Communication by:

Dr. Janet Wasko	Chairperson
Dr. Biswarup Sen	Core Member
Dr. Gabriela Martinez	Core Member
Eric Priest, J.D.	Institutional Representative

and

J. Andrew Berglund	Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded September 2014

This document was produced entirely with free and open source software.

DISSERTATION ABSTRACT

Benjamin J. Birkinbine

Doctor of Philosophy

School of Journalism and Communication

September 2014

Title: Incorporating the Commons: A Political Economic Analysis of Corporate Involvement in Free and Open Source Software

Free (libre) and open source software (FLOSS) emerged in the 1980s as a radical alternative to proprietary software. Fighting back against what FLOSS enthusiasts viewed as overly restrictive intellectual property protections placed on proprietary software, FLOSS was designed with the intent of granting users the right to study, modify, adapt, or otherwise tinker with the source code of software. As such, FLOSS users were able to collaborate in producing software that could be distributed freely and widely to others, who could, in turn, make changes to the software. As FLOSS projects grew in popularity, the productive process was spread throughout a broad network of distributed users, all of whom could work on the code. The result of this process was the creation of robust, effective, and efficient forms of software that could compete with those offered by large software companies.

Increasingly, however, some of those large software companies became involved in the development of FLOSS projects. On its face, this may seem to be a contradiction of interests. Why would a for-profit company invest in the development of software that is made freely available for others to use? This is the contradiction that lies at the heart of this research project. More specifically, this project looks at the dynamics that exist

between communities of FLOSS developers and the corporations that are involved in or make use of their projects. Working from a critical political economy perspective, this study complicates theories of the commons and commons-based peer production by illustrating how FLOSS processes and products are being incorporated into broader corporate structures and strategies.

The three case studies presented – Red Hat, Microsoft, and Oracle's acquisition of Sun Microsystems – exemplify different elements of this dynamic. Red Hat provides an example of how a company that relies exclusively on free software can be turned into a profitable business. The Microsoft case demonstrates why the company has undergone a transition from vehement opposition to FLOSS toward a more supportive position. Finally, Oracle's acquisition of Sun Microsystems demonstrates how FLOSS communities cope with changing ownership structures and unwanted corporate interference into their projects.

CURRICULUM VITAE

NAME OF AUTHOR: Benjamin J. Birkinbine

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR
Southern Illinois University Carbondale, Carbondale, IL
University of Wisconsin-Green Bay, Green Bay, WI

DEGREES AWARDED:

Doctor of Philosophy, Media Studies, 2014, University of Oregon
Master of Arts, Media Theory and Research, 2010, Southern Illinois University Carbondale
Bachelor of Arts, Communication, 2005, University of Wisconsin-Green Bay

AREAS OF INTEREST:

Political Economy of Communication
Open Source Technology
Communication Theory
Media Studies

PROFESSIONAL EXPERIENCE

Graduate Teaching Fellow, School of Journalism and Communication, University of Oregon, 2010-2014

Teaching/Research Assistant, College of Mass Communication and Media Arts, Southern Illinois University Carbondale, 2008-2010

Instructor, Department of Communication, University of Wisconsin-Green Bay, 2006-2008

GRANTS, AWARDS, AND HONORS

The Dallas Smythe Award, Awarded by the International Association of Media and Communication Researchers for the paper, “Incorporating the commons: Toward a political economy of corporate involvement in free and open source software,” 2014.

Digital Scholarship Center Graduate Affiliate, 2014

Rapid Response Grant, Co-sponsored by the Open Society Foundation and the International Association for Media and Communication Researchers, 2014

The Columbia Scholarship, School of Journalism and Communication, University of Oregon, 2013

Outstanding Teaching by a Doctoral Student, School of Journalism and Communication, University of Oregon, 2013

Glenn Starlin Fellowship, School of Journalism and Communication, University of Oregon, 2013

Nominee for Doctoral Research Fellowship, School of Journalism and Communication, University of Oregon, 2013

The Columbia Scholarship, School of Journalism and Communication, University of Oregon, 2011

Lucien P. Arant Scholarship, School of Journalism and Communication, University of Oregon, 2010

ACKNOWLEDGMENTS

No project as formidable as a dissertation can ever be undertaken in complete isolation. Along the way, I've had inspiration, assistance, advice, critiques, clarifications, and collaboration with numerous people who made this possible. I cannot do justice to the entire community who helped, but I would like to thank a few in particular.

I thank my dissertation committee for their patience, comments, and critiques. Janet Wasko was an incredibly gracious, accommodating, and fun adviser throughout the process. Bish Sen provided critical feedback and always pushed me to think about the broader implications of my work. Gabriela Martinez was equally supportive, provided great feedback, and was always available for conversations. Finally, Eric Priest became one of the few law professors to serve on a dissertation committee at the University of Oregon. He deserves additional credit for coordinating his involvement while the law school followed a different academic calendar than the rest of the university.

I thank the members of my cohort – Toby Hopp, Erica Ciszek, Francesco Somaini, Brant Burkey, and Fatoumata Sow – who were sources of inspiration, support, and friendship. In addition, I would like to thank Jeremy Swartz for the introduction to open source and the guided tour of my first Open Source Convention (OSCON) in Portland in the summer of 2011. That experience, along with the ensuing discussions about the broader implications of open source, led to a change in my academic focus that ultimately provided the impetus for this study. I'm also thankful for the support and friendship of Brenna Wolf-Monteiro, Tewodros Workneh, Randall Livingstone, Jacob Dittmer, Lauren Bratslavsky, Glenn Morris, Karen Estlund, Andre Sirois, Jolene Fisher, and Geoff Ostrove.

Outside the School of Journalism and Communication, Kat at the ResNet Help Desk helped me with my first Linux install. Without her initial help, this journey may have not been possible. I am now happy to say that I've passed the gift along to others. John Russell and everyone at the Digital Scholarship Center provided support and were always willing to assist in whatever way they could. In addition, the members of the Eugene Unix Gnu Linux Users Group (EUGLUG) were an incredibly welcoming group of people who were kind and patient enough to help a total noob like myself with various Linux questions. I would especially like to thank Larry Price and Jacob Riddle for setting aside extended periods of time to help me with this project and related topics.

Last, but certainly not least, I would like to thank my family. I thank my mother and father for their years of support and patience as I've pursued graduate studies. Undoubtedly, without you, none of this would be possible, and I am forever indebted. My sister, Ann, inspires a wealth of adjectives: congenial, affable, good-humored, gregarious, etc., and I'm thankful for the years of fun we've shared. My son, Caden, was only two years old when this journey began, and he is now well on his way to becoming a young man. I've enjoyed watching you grow over these years, and I look forward to many more. You are a blessing. Finally, my love and inspiration, Roberta, *meu coração.* You have truly been wonderful throughout all these years. In the past, I have seen other writers thank their partners, they allude to the fact that a great deal of patience is necessary to cope with someone who is going through the process of writing a dissertation. I now know what they mean, and I am truly fortunate to have had you as a companion throughout this process. You are truly a gift.

For the community, and *para o meu coração*.

TABLE OF CONTENTS

Chapter Page

Chapter Page

Chapter Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

In March of 2012, The Linux Foundation released a report entitled, “Linux Kernel Development: How Fast it is Going, Who is Doing It, What They are Doing, and Who is Sponsoring It.” The kernel is an essential part of an operating system that facilitates communication between computer hardware and software, and the Linux kernel development project is considered to be “one of the largest cooperative software projects ever attempted” (The Linux Foundation, 2012, 1). Aside from a technical overview of how kernel development has changed over time, the authors included a curious note in the report's highlights: Microsoft was one of the top 20 contributors to the kernel. This marks the first time that Microsoft appeared as a top contributor, but was not the only corporation in the top 20. Other corporate contributors included Intel, IBM, Google, Texas Instruments, Cisco, Hewlett-Packard, and Samsung, as well as others. The Linux operating system is a form of Free (Libre) and Open Source Software, or FLOSS, which allows users to freely study, use, copy, modify, adapt, or distribute the software. Why, then, would major corporations contribute directly to a FLOSS project, especially when that project seemingly does not directly contribute to corporate profits? This question becomes even more curious when one considers that many of the companies contributing to the kernel not only compete with one another in the market for information technology, but companies like Microsoft and Google are direct competitors with Linux in the market for operating systems.

Indeed, Steve Ballmer, the Chief Operating Officer of Microsoft, once referred to Linux as “a cancer that attaches itself in an intellectual property sense to everything it touches” (Greene, 2001). Ballmer was referencing the GNU General Public License, or

GNU GPL, which is the most commonly used free software license. The GPL grants users of GPL-protected software the right to study, use, copy, modify, or adapt the software as he or she wishes. In addition, users are granted the right to redistribute the software, as well as a modified version, and the user may even charge a fee for the modified version, provided that the distributor does not place greater restrictions on the rights granted by the GPL. By granting such rights, the GPL does not preclude corporations from modifying free software or charging a fee for their modified versions, but the corporation must still grant free software rights to end users. Ballmer's quote implies that free software is antithetical to commercial software companies. If this were the case, then Microsoft or any other commercial software firm would have no incentive to contribute directly to one of the largest open source projects. This seemingly contradictory stance lies at the heart of this dissertation project. To further exacerbate this contradiction, consider the fact that Ballmer made his denunciation of Linux on June 1, 2001. Merely 27 days later, on June 28, 2001, the United States Department of Justice found Microsoft guilty of monopolistic business practices in violation of the Sherman Antitrust Act primarily for bundling its Internet Explorer web browser with its Microsoft Windows operating system as a way to rapidly increase its share of the market for web browsers. However, Microsoft has dramatically changed its position on Linux and open source since 2001, as signified by its inclusion in the top 20 contributors to the Linux kernel.

In 2012, Microsoft created Microsoft Open Technologies, Inc., a wholly owned subsidiary dedicated to facilitating interoperability between Microsoft and non-Microsoft technologies, while promoting open standards and open source. What changed during this twelve-year period that Microsoft would so dramatically reposition itself in relation

to FLOSS? Moreover, why are so many other corporations contributing to open source projects?

In this project, I am primarily concerned with the seemingly contradictory relationship between FLOSS communities and for-profit corporations. The dissertation explores the nature of this relationship by focusing on three case studies that illustrate different ways that corporations have been involved in FLOSS projects. However, I am also interested in whether corporate involvement in FLOSS projects will change the dynamics of the broader FLOSS community over time. In other words, this project investigates the extent to which corporations like Red Hat, Microsoft, and Oracle wield power over or within FLOSS projects. If so, in what ways? In this sense, the issue of corporate power is the center of the analysis. Finally, one of the proposed outcomes for this project is to speculate as to whether increasing corporate involvement in FLOSS projects will have consequences for the future of FLOSS communities and what those consequences may be. To sum up, then, the current project is guided by the following research questions:

RQ1: What is the relationship between proprietary, for-profit corporations and free and open source software communities, and how has this relationship changed over time?

RQ1a: What are the power dynamics between corporations and the FLOSS community? In other words, does one party hold the ability to exert influence on the other and how?

RQ2: What constitutes value for each of these stakeholders? What value do corporations provide for the FLOSS community, and what value does the FLOSS community provide for corporations? Do any external factors or stakeholders exist that may profit from this relationship?

To address these questions, three case studies illustrate different types of relationships that FLOSS projects have with corporations. Specifically, I focus on Red

Hat, Microsoft, and the Oracle Corporation's acquisition of Sun Microsystems. These case studies were strategically chosen because they represent three very different examples of corporate involvement in FLOSS projects. Red Hat is the largest and only publicly traded company providing software and services that are completely based on free software. As such, Red Hat cannot rely on traditional copyright protections to exclude others from using the underlying source code included in its software. Thus, I explore *how* Red Hat has been able to create a profitable business based on free software.

Microsoft was chosen because it has been viewed as the antithesis to FLOSS throughout its corporate history. Now, however, Microsoft has signaled that it is committed to and supportive of FLOSS projects. Consequently, the chapter on Microsoft traces the company's long and winding history with FLOSS, but focuses specifically on key moments throughout the company's history that demonstrate contradictions between the public claims made by the company and its actions. Whereas the investigation of Red Hat was driven by an interest in *how* the company uses FLOSS, the investigation of Microsoft is interested in *why* the company has shifted its position to FLOSS.

Finally, the third case study focuses on *what* happens when a company that supports various FLOSS projects is acquired by a company that does not. Specifically, Sun Microsystems provided support for various FLOSS projects, but was later acquired by the Oracle Corporation, which had different plans for those projects. In that chapter, I focus on the diverse destinies of three such projects – the OpenSolaris operating system, the MySQL relational database management system, and the OpenOffice productivity software – and the ways that the communities involved in those projects resisted Oracle's encroachment into their projects.

When considered together, these three case studies are indicative of the general tendencies of corporate involvement in FLOSS projects. Moreover, all three companies are some of the largest software companies in the world. While Red Hat may not have the same level of revenue as Microsoft and Oracle, the company is the largest and only publicly traded company operating almost exclusively in FLOSS. As such, Red Hat was chosen because it illustrates how FLOSS can be transformed into a profitable business. Microsoft and Oracle were selected because they are the two largest software companies in the world when measured in total revenue. An explanation of how, why, and when these companies compete or cooperate with FLOSS communities offers a germane moment for understanding the dynamics existing between corporations and FLOSS communities.

Furthermore, an increasing amount of our lives spent on the Internet where we communicate with friends and colleagues, read news, watch movies and television, and listen to music, among other activities. When we connect to the Internet and visit web sites, our requests for information are relayed through a network of interconnected servers that facilitate communication between other clients on the network. The operating systems running those servers are increasingly FLOSS projects like Linux or FreeBSD, but Microsoft also designs server software. This provides another example of FLOSS projects competing with proprietary companies like Microsoft. Therefore, whether we realize it or not, our ability to connect to the Internet may depend, in part, on the ability of FLOSS projects to work together with proprietary software. Consequently, understanding the ways in which proprietary software and FLOSS projects work together, as well as what happens when these relationships break down, is an important step in

unpacking the relationships that enable and, at times, constrain our ability to connect with others online. This is precisely the purpose of this project.

To explain exactly how this project research was completed, CHAPTER III provides a more in-depth overview of methodology and method. The remainder of this chapter will focus on providing an introduction to FLOSS. Readers who are already familiar with the history of FLOSS and its defining characteristics may wish to skip directly to the next chapter, which more succinctly outlines the theoretical frameworks drawn upon for this study, as well as an overview of the relevant literature that contextualizes the study.

In the following sections, I situate FLOSS within the history of computing and provide some basic information about its size and scope. In addition, I draw distinctions between free software and open source by focusing on the foundational figures associated with each community. While there are differences between free software and open source, I will be using the combined term FLOSS throughout this dissertation unless a specific reference to one or the other is required. After clarifying the differences between free software and open source, some of the individual motivations for those contributing to FLOSS projects are addressed. After this introductory material, I discuss the relevance of this study and its contribution to a broader corpus of knowledge. Finally, the chapter concludes with an outline of the remainder of the dissertation.

Situating Free (Libre) and Open Source Software

Although free software and open source communities are related and, in some cases, not mutually exclusive, each of them have distinct characteristics that can best be described by reference to the ethos underlying each movement. To contextualize the

emergence of FLOSS within the evolution of the computing and software industries, a brief history of these industries is provided below. Following that discussion, I focus on situating two key figures associated with FLOSS within their historical context: Richard M. Stallman and Linus Torvalds. These two figures represent free software and open source, respectively.

History of Computing and Software

Prior to the use of machines for processing information or calculating differences in numbers, human beings performed such work. But human calculations were, at times, prone to errors. To reduce this uncertainty, Charles Babbage, a philosopher and mathematician working at the University of Cambridge in 1822, proposed that it was "only by the *mechanical fabrication of tables* that such errors can be rendered impossible" (Gleick, 2011, 95). Such was the proposition for Babbage's Difference Engine, which performed routinized calculations mechanically, and was arguably the genesis for modern computers as we know them today. Later, Babbage expanded on his idea planned a new type of machine that was capable of being controlled by instructions that could be encoded and stored to facilitate operation. The new iteration of the idea was called the Analytical Engine, but this still only provided the idea for the hardware or mechanisms necessary for such processes to occur. What was needed for this hardware was software.

The idea for software arguably originates with Augusta Ada Byron King, the Countess of Lovelace, or otherwise known simply as Ada Lovelace. She developed the idea that Babbage's Analytical Engine could perform a series of operations beyond the mere calculation of numbers. By abstracting from the differences between two things,

Lovelace posited that the Analytical Engine could be programmed to perform operations that relied on symbols and meaning, which, in turn, could be communicated to the machine. Although Lovelace's idea was never realized in her lifetime, she is credited with developing the idea for software and is known as the first programmer (Computer History Project, 2008).

While Babbage and Lovelace are credited as pioneers in developing the ideas for modern computers and software, the construction of such machines did not begin until World War II. Developments in the field of computer science and information theory – like Kurt Gödel's incompleteness theorem, Alan Turing's idea for a Universal Turing Machine, Claude Shannon's mathematical theory of communication, and Norbert Wiener's cybernetics – provided the intellectual inspiration for the development of such machines. Before, during, and after World War II, many of the developments leading to modern computers were used for military purposes. Most notably, perhaps, were the German Enigma machine that was used to encrypt secret messages and the electro-mechanical *bombes* used by the United Kingdom to decipher those messages (Smith, 2011). However, in 1941, Konrad Zuse, a German electrical engineer, built the Z3, which is regarded as the first electro-mechanical, programmable, fully automatic digital computer (Zuse, et al., 2010). The first comparable computer in the U.S. was developed by John Atanasoff at Iowa State University in 1942 (Copeland, 2006). Only one year later, the first fully functioning electronic digital computer was put to use by the cryptanalysts working at Bletchley Park in the U.K. as part of the Government Code and Cypher School. The Colossus, as the new machine was known, was programmed to decipher German communications during the war. By the end of the war, Bletchley Park had 10 Colossi working to decode German communications (Copeland, 2006).

Following these initial developments, the development of modern computers accelerated as many of the early pioneers began working for academic institutions and private companies after the war. In the U.S., Grace Hopper, who served in the United States Navy Reserves as a member of the Women Accepted for Voluntary Emergency Service (WAVES) during World War II, was assigned to the Bureau of Ships Computation Project at Harvard University. While there, she worked on the Mark I computer project, which was built by IBM. Later, after she began working for private companies, Hopper popularized the idea of machine-independent programming languages. This led to the development of the Common Business-Oriented Language (COBOL). Hopper is also credited with popularizing "debugging" as a term for removing defective material or code from a program. While Hopper may not have invented the term, she popularized it by literally removing a moth from a Mark II computer at Harvard University after it had caused the machine to short circuit (Deleris, 2006).[1]

During the 1960s, the creation of microprocessors drastically reduced the cost of computing. As such, communities of hobbyist programmers and computer enthusiasts began to experiment with the technology. For example, Gordon French and Fred Moore began the Homebrew Computer Club, which met at the Community Computer Center in Menlo Park, California, and provided a forum for hobbyists to trade parts and advice about the construction of personal computers. More will be said about this specific hobbyist community in CHAPTER V when the rise of Microsoft is discussed. However, aside from these hobbyist communities, the majority of computer development occurred within the military, academic institutions, and private companies.

1 Interestingly, a photo of the moth that was removed from the machine is available from the Naval Historical Center at http://www.history.navy.mil/photos/images/h96000/h96566kc.htm (last accessed August 2, 2014).

Most notable were the initial developments within the Defense Advanced Research Projects (DARPA), as well as the Artificial Intelligence Lab at the Massachusetts Institute of Technology (MIT). Programmers working at the time were using a proprietary programming language called Unix, the intellectual property rights for which were owned by AT&T. One of the programmers working at MIT at the time was Richard Stallman, who found that when he wanted to work with the Unix programming language outside of officially sanctioned spheres, he was denied access to the code by AT&T. In protest, he posted a message to a computer-based bulletin board saying that he was developing a Unix-based language that would be available for free so that others could use the language however s/he saw fit. The programming language was called "GNU," a recursive acronym standing for "Gnu's Not Unix." Along with the programming language, Stallman developed the GNU Public License (GPL), which stipulated that anyone could access the source code for free, and that anyone using the GPL agreed to make their contributions available under the same conditions. This would ensure that computer programmers could freely share their work with one another, thereby creating a common form of property that developed in opposition to its proprietary and closed counterparts.

Stallman thus became the impassioned leader of the crusade against proprietary software. He viewed access to source code as a fundamental right, which he wanted others to believe in as well. He summed up this view in his famous dictum, "Free as in freedom, not as in free beer," thus positioning free software as a moral right (Stallman, 2002). In addition, the free software definition stipulates that “users have the freedom to run, copy, distribute, study, change and improve the software” (Free Software Foundation, 2012). As the principles of free software grew beyond the borders of the

U.S., others have tried to reduce the confusion over the English term "free" by using the French term *libre* rather than *gratis*. Stallman developed the Free Software Foundation (FSF) as a way to promote his crusade against proprietary software, and he represents an impassioned counter-cultural figure who still continues to espouse his free software philosophy.

While Stallman is generally considered to be the leader of the free software movement, open source software is generally associated with Linus Torvalds. In many ways, Torvalds and Stallman have similar stories, but differ on philosophical terms. During the 1980s, free software projects were being developed but generally on a smaller scale. Free software had not yet found a way to coordinate efforts on a grand scale. Torvalds wanted to work on kernel development for an open-source operating system. Rather than relying on numerous programmers all working independently on such a task, Torvalds released the source code for his project, which he was calling "Linux," a portmanteau of his name, Linus, and the language he was working with, Minix (itself a simplified derivative of AT&T's Unix). Torvalds suggested that anyone who was interested in contributing to such a project was encouraged to do so, provided that they release their work back to the community so that others could progressively work toward completing the kernel. The project proved to be successful, and eventually led to the creation of the open source operating system, Linux. Coordinating such a large-scale programming project was accomplished by asking those working on the code to release their work, no matter how small the changes seemed. The rationale was that coordinated efforts reduce the amount of redundant work, which was summed up in the adage “with many eyes, all bugs are shallow,” which Eric Raymond refers to as “Linus's Law” (Raymond, 2000).

Stallman and Torvalds differ with respect to how free software projects ought to relate to proprietary software. Whereas Stallman tends to be somewhat more rigid in his opposition to proprietary software, Torvalds is less so. Williams (2002) describes the decisive moment at a conference in which Stallman and Torvalds appeared on a discussion panel together. Torvalds expressed admiration for the work that Microsoft was doing and suggested that free software advocates could even work together with companies. Such a suggestion was generally seen as taboo since Stallman was perceived with esteem by the programming community, and the Free Software Foundation generally took a very adamant stance against proprietary software companies. However, this was apparently a watershed moment in which the fervor of the free software movement thawed a bit and Torvalds came to represent a more liberal approach to free software.

In sum, then, we can understand the free software and open source movements within these differing philosophies. Stallman and free software advocates tend to make moral claims against supporting proprietary software, while Torvalds and open source are associated with a more liberal and inclusive stance. While Stallman and Torvalds have been used to illustrate the differences between free software communities and open source communities, they should not be viewed as mutually exclusive communities, nor should Stallman and Torvalds be seen as representative of the entire free software and open source communities. One of the peculiarities of the free and open source software community is that, although the overall community is united in their belief that software ought to be free for users to study, modify, adapt, or customize, its members will often vehemently defend their preferred free software project while deriding others. In a sense, this signals to others where their loyalties lie and engenders stronger ties within niche

communities that exist within the larger FLOSS community. The present project is less concerned with these intra-group fissures than the relationship of the community as a whole to the corporations that rely on their labor. To that end, the combined term “Free (Libre) and Open Source Software” or “FLOSS” is used to refer to the overall community.

The Size and Scope of FLOSS

Since its emergence in the 1980s and 1990s, FLOSS has proved to be a tremendously efficient and effective way of producing software. As an example of the size and scope of some FLOSS projects, consider the Linux kernel, which was discussed in the introduction to this chapter. When it was first released in 1991, the Linux kernel featured approximately 10,000 lines of code. Version 3.10 of the Linux kernel was released in June of 2013 and featured almost 17 million lines of code, which was produced by nearly 1,400 developers and 243 companies (The Linux Foundation, 2013). Aside from the sheer growth in its size and complexity, Linux as an operating system has become widely used. For example, Linux enjoys more than 96% market share in the market for supercomputer operating systems (Top500.org, 2014). While Linux does not yet have a significant share of the personal computing desktop market, the operating system has been customized and used within a variety of contexts.

Between 1999-2001, four cities and municipalities in Brazil – Amparo, Solonópole, Recife, and Ribeirão Pires – passed laws that required government agencies to use or give preference to Linux (Tramontano & Trevisan, 2003; Festa, 2001). The decision to switch to free software systems was mainly economic, as Brazil reported spending nearly $1 billion on software licensing fees to Microsoft between 1999-2004

(Kaste, 2004). By switching to free and open source software, Brazil estimated that they could save approximately $120 million per year (Kingstone, 2005). Similar measures were taken in Kerala, India, during 2008, as the state banished Microsoft and allowed only GNU/Linux free software to be used for the mandatory state information technology exams (Kochi, 2008). The German city of Munich developed its own version of Linux called LiMux (Linux in Munich), which it uses as an operating system for its 15,000 city council members instead of Microsoft Windows (Saunders & Morrison, 2014). The National University of Defense Technology in China has also developed its own Linux-based operating system called Kylin. In addition, the computers used for the One Laptop Per Child project, which was founded with the goal of bringing low-cost computers to developing countries for educational purposes, featured a free and open source operating system based on Fedora. Within the United States, Linux is used for high-level military operations. For example, the United States Navy announced that its new $3.5 billion warship, the USS Zumwalt, will effectively serve as an armed floating data center that features server hardware running various Linux distributions and more than 6 million lines of code (Gallagher, 2013). In addition, the Linux Foundation (2014) claims that the International Space Station will migrate to Linux to power the station (The Linux Foundation, 2014).

Beyond the increasing use of Linux, open-source principles have been used in areas outside of information technology. For example, open source hardware increases access to physical goods, including furniture, musical instruments, construction materials, and wind turbines for generating renewable energy. Such projects are particularly attractive to those living in developing countries, where access to information, goods, and services may be restricted or limited. One of the more ambitious projects in this area is

the Open Source Ecology project, which offers “open source blueprints for civilization,” and includes instructions for building industrial machines with recycled or low-cost materials (Open Source Ecology, 2014). While this is just one notable example, it demonstrates the optimism and creativity involved in applying open source principles to a whole way of living rather than simply information technology. However, the core values inherent in these projects do not necessarily originate in open source software. Rather, the cultural values of openness and sharing are what hold the most value. When applied throughout an entire community, these principles hold the promise of a more sustainable future, especially when such principles are linked with environmental and ecological preservation practices.[2] But these principles only become radical propositions in a system that discourages or provides little incentive for such behaviors.

What these examples should illustrate is that Linux in particular, but FLOSS more generally, has become more than just a tool used within the computer hobbyist community. Its widespread and increasing adoption across the globe within a variety of high-level contexts demonstrates the power of the FLOSS production model as well as the effectiveness of its products. As FLOSS continues to be used within an increasing variety of contexts, understanding the ways in which corporations, governments, non-profit organizations, and other types of institutions are involved in FLOSS projects will become increasingly important. Therefore, FLOSS provide an important area for research not just because of its increasing ubiquity, but also because of the claims that have been made about the democratic, egalitarian, and non-market characteristics of its

2 These practices and the potential of environmental media were explored in greater detail during the Inaugural History and Theory of New Media Unconference at the University of Oregon in 2012 (Jher & Birkinbine, 2012).

products and processes. This is precisely how this project seeks to contribute to such debates.

Before concluding this brief introductory overview, however, I would like to clarify some additional terminology as well as situate the activities of the FLOSS community within a broader context. FLOSS communities comprise a socio-technical system insofar as their activities are made possible by and exist within a technologically mediated realm. However, in larger cities or in cities with a relatively large community of people working in the information technology industries, one can find local Linux Users Groups, or LUGs, where regular meetings are held to promote FLOSS, to assist new users with installing FLOSS, to troubleshoot any issues that may arise when using FLOSS, or to simply meet other people interested in FLOSS. In this sense, the social connections that exist within these groups are mediated by their mutual interest in technology. Because members of the FLOSS community are brought together by their mutual appreciation of technology, their cultural practices depend upon and are supported by interconnected network technologies. As more people become connected to the network, the opportunities for additional participants in these communities grow. This also means that those who lack a network connection will have a difficult time contributing directly to the cause of free software or open source software development.

Therefore, both free software and open source communities exist within a very particular and privileged technological realm that requires a certain level of intellectual and economic development. Furthermore, having a network connection is not necessarily enough to enable direct participation to FLOSS projects; Stallman and Torvalds are computer programmers who have the ability to read and write code. As such, they are not just users of software, but they have the ability to actively engage with the software, to

make changes to its code. In this sense, both Stallman and Torvalds can be called "hackers."

Hacking, Cracking, and Motivation

The term "hacker" has taken on negative connotations recently, but the term is generally used to describe anyone who "tinkers" with or makes changes to technology to create something new. In this sense, the practice of hacking could be seen as a form of innovation, although profitability is not always a prerequisite motivation for hacking. Steven Levy (1984) outlined the principles of the hacker ethic. Among other elements, Levy claimed that computers can be used for creative purposes, hackers ought to be judged by the quality of their work rather than any other characteristic (gender, race, ethnicity, etc.), and that having the ability to hack is a prerequisite for hacking. This last caveat may seem obvious but, in order to perform a hack, a hacker must have access to the technology (in this case, the source code). For hackers, closed, proprietary technologies that do not allow for tinkering are unjust.

The practice of removing proprietary restrictions on closed technologies is known as "cracking," which can be performed on a CD or DVD that does not allow copying, a video game console that requires users to only purchase games and software from a company (Microsoft Xbox, Nintendo Wii, Sony Playstation), or on proprietary software or operating systems. An important distinction to make here is that crackers and/or hackers may not necessarily be interested in the consequences of their crack/hack. Rather, they are motivated by the desire to signal to other crackers/hackers that they deserve credit for the sophistication of their crack/hack. This signaling motivation is also recognized within open source software communities (Lakhani & Wolf, 2005), but

whereas crackers are interested in freeing technology from its restrictive measures, hackers are interested in remixing, modifying, adapting, or creating something new from a given product.

The same signaling motivation that is used to explain why hackers do what they do has been used to understand why programmers contribute to FLOSS projects. Lakhani & Wolf (2005) explain that signaling can take place within at least a couple levels. At the level of the individual, a single hacker may perform a hack in order to signal his or her skills to others. Hackers might also use this type of signaling as a way to communicate their skills to potential employers to secure paid employment. Gaining recognition within the broader community for performing certain programming tasks effectively can translate into increased job opportunities with companies looking for specific skills. However, a different type of signaling takes place between groups of hackers. Groups or collectives may signal their prowess to others by shutting down a web site or otherwise disrupting services. Often, this is done in the spirit of competition, but can also be explicitly driven by a particular ideology. For example, nationally based hacker groups can be found in Syria where a pro-Syrian government hacking group called the Syrian Electronic Army has waged hacking battles against the pro-rebel hackers associated with the Free Syrian Army (Fitzpatrick 2012). In these situations, hacker groups strategically target the web sites of their opponents to signal the strength of their movement.

Although the signaling motivation appears to be the most prevalent motivation, Weber (2004) identifies other motivations as well. In a survey of self-identified hackers, respondents reported their primary motivation for contributing to FLOSS development was a desire to challenge oneself and perform creative work. This seems to support what

Levy (1984) identified as a primary tenet of the hacker ethic: creativity and aesthetics. Weber (2004) also found additional motivations reported in the survey, including the belief that all software should be free, which echoes the philosophy of Richard Stallman and the Free Software Foundation. Weber concludes that motivations are diverse and that the results from these surveys need to be properly contextualized. For instance, many contributors to FLOSS development do not disclose their identity or any institutional affiliation. Indeed, a look at the credits file for users contributing to the development of the Linux kernel shows that most contributors are listed in the "unknown" category. This means that a large portion of the FLOSS community simply chooses not to self-identify. Therefore, the results of any survey that claims to represent the entire FLOSS community must be approached somewhat skeptically.

While motivations represent one category of questions about the FLOSS community, the more robust questions about FLOSS community are related to the economics and governance of FLOSS as both a process and the products created by the community. These topics will be covered in more detail in CHAPTER II because they are reflective of certain theories about the commons and commons-based peer production. In what follows, however, the scope and focus of the study are discussed, including how the study will contribute to a broader body of literature.

Justification for the Study and Its Contribution to Scholarship

FLOSS products and the productive process that make those products possible have been widely lauded as revolutionary changes that enable greater degrees of freedom and autonomy on behalf of users and contributors (Benkler, 2006; Raymond, 2000; Stallman, 2002). This project intervenes in these debates by tempering these claims with

a critical approach to understanding technological change and systems of production within a broader capitalist system. Although some of the more celebratory arguments about FLOSS are notable for explaining the internal dynamics of FLOSS production and the unique social, technical, and legal characteristics of FLOSS products that make peer production possible, these analyses have not placed commons-based peer production within a broader social context to illustrate how such production intersects with capitalist production. The purportedly revolutionary changes brought about by FLOSS and commons-based peer production are now becoming incorporated into corporate strategies and corporate structures. As such, FLOSS projects constitute a contested terrain, whereby these projects are faced with a number of organizational difficulties. These difficulties are primarily associated with the benefits and detriments of finding corporate sponsors to support those projects. In these cases, a community of developers must cope with varying degrees of corporate influence in defining the direction of the project. In many cases, employees of the corporation and members of the community are not mutually exclusive. More specific cases will be discussed in the chapters that follow, but the primary intention of this project is to highlight the diverse ways that both corporations and the broader FLOSS community cope with co-presence.

Aside from tempering the claims of novelty by those who use FLOSS as a primary example of commons-based peer production, this project also contributes directly to our understanding of commonly held resources under capitalism. As will be discussed in greater depth in the following chapter, the commons are often held in contradistinction to capitalism. The rise of capitalism saw the enclosure of the commons and the end of common right. The work of Elinor Ostrom (2005; 1990) has helped to broaden our understanding of the diverse array of commonly held resources as well as

institutional diversity designed to protect them, particularly in the face of capitalism. That said, the knowledge commons are also subjected to types of enclosure. James Boyle (2008) referred to this as the Second Enclosure Movement, whereby knowledge and information are becoming enclosed by restrictive intellectual property protections. Copyleft, most notably in the form of the GNU General Public License, seeks to maintain common rights for knowledge or informational resources. This project demonstrates how FLOSS as a knowledge commons is not becoming *enclosed* in the absolute sense of total exclusion, but how knowledge commons and their attendant cultural practices are becoming *incorporated* into corporate strategies and corporate structures.

By looking at three different case studies that illustrate the different ways that FLOSS is being incorporated into corporate structures, this dissertation also engages with broader debates about the political economy of communication, digital labor, participatory culture, and information politics. The project contributes most immediately to the field of political economy of communication in that FLOSS remains a relatively understudied phenomenon within the approach. Those working within the political economy of communication approach are broadly interested in working toward more just and democratic communication systems that truly serve the needs of local communities. A large part of this work has been to critique both corporate and state power, particularly in terms of the way it operates within media systems for the transmission of ideological messages. While this project does not specifically focus on message transmission, I am interested in analyzing the products and processes of FLOSS development by placing issues of corporate power at the center of the analysis. By doing so, this project highlights the dual challenge of the FLOSS community's need to ensure the long-term

survival of their projects and the corporations' desire to harness the collective labor power of the FLOSS community.

By focusing on this dynamic, the current project also engages directly with debates about digital labor (Lazzarato, 1996; Terranova, 2004; 2000; Scholz, 2013). Debates about digital labor have focused on fan cultures and other forms participatory cultures whereby individuals voluntarily contribute to the creation of novel cultural artifacts (De Kosnik, 2013). But these debates have also focused on the unpaid free labor performed by individuals online as value is extracted from data about their browsing habits (Andrejevic, 2012; Fuchs, 2012). In addition, Fuchs (2013) has focused on issues of class and exploitation, while Bauwens (2013) has focused on the possibilities of peer-to-peer organizing. This project contributes directly to these debates by focusing on the dynamics between laborers in the FLOSS community and the corporations that profit from their labor. This project is unique in that it focuses on the intersection between the digital labor of FLOSS programmers and the corporations using FLOSS as a part of their business operations. As will be made more clear in the literature review, previous studies have focused either solely on the ways businesses can use FLOSS to their advantage, or on how FLOSS enables greater degrees of freedom and democracy. By focusing on the intersection of labor and corporations, this project is also unique within the broader body of literature on free and open source software.

Insofar as this project focuses on forms of collective labor that produce digital texts, it is also broadly concerned with the changing institutional context of participatory media. Jenkins (2006) used the term “participatory culture” to describe the ways in which networked communications technologies enable novel forms of meaning-making to arise, whereby consumers can remix cultural artifacts in new and creative ways. In

this sense, consumers are transformed into “prosumers” because their relationship with a particular commodity is no longer solely based on consumption, but they have the opportunity to produce new and unintended meanings. While Jenkins primarily focused on audience members' relationships with commodities, free and open source software projects might also be framed as a form of participatory culture. However, the differences between these types of participatory culture need to be distinguished carefully. While the *processes* of production may be similar in the sense that they rely on a variety of inputs from geographically dispersed populations who contribute to an overall project, the end *product* in each case is quite different.

Analyses of participatory culture tend to focus on cultural artifacts that are often held under strict copyright protections by their ultimate owners. FLOSS, on the other hand, relies on an increasing repository of code that is protected by copyleft and other alternative intellectual property licenses that encourage and allow other users to build upon the work that has been performed previously. Another key difference in this respect is the form of the end product itself: in most instances of participatory culture, the end products come in the form of content designed for literary, artistic, political, or entertainment purposes (fan fiction, remixes, mash-ups, culture jamming, etc.), while the end product in FLOSS is source code, with which others can study, modify, adapt, or build upon. In this sense, the end product of FLOSS tends to be more technical, in the literal sense of the word (of or relating to the applied and industrial sciences), than the end products of participatory culture. This distinction is, perhaps, a crude one. I am not trying to imply that FLOSS projects cannot be artistic, political, or even literary, nor am I trying to imply that a mash-up or remix cannot be technical. Both of these creations

involve a certain level of craft, but it seems to me that there is still a fundamental difference in the end products even if the productive processes are similar.

Finally, in an increasingly networked world that is mediated by the use of information and communication technologies, the struggle for ownership and control of information has risen to the forefront of many national and international debates. In this sense, the current project can be contextualized within broader conversations and debates about informational politics. The revelations of Edward Snowden in 2013 (and, at the time of writing, still ongoing) that exposed the massive and widespread collection of personal communications data by the National Security Agency (NSA) in the United States revealed how the NSA systematically collected personal communications data from citizens both within the United States and around the world. These revelations have opened up a new space for debate about the right to privacy in the digital age.

The majority of these debates have focused on curbing the power of the state to collect massive amounts of data on its citizens. However, these criticisms can also be directed at the corporations who were either coerced into cooperating with the NSA or complicit in such collection. While this project does not specifically focus on the individual's right of privacy, the project does focus on the ways in which information – in the form of source code – can provide a contested terrain in which struggles over intellectual property and informational resources take place. The FLOSS community often uses alternative intellectual property licenses to ensure that their creations remain freely available as commons-based resources rather than becoming enclosed by restrictive or exclusionary intellectual properties for use as a corporate commodity. Throughout this study, I focus on the ways in which this is happening, why it is

happening, and how the FLOSS community responds to corporate encroachment into its communities.

Overview

The remainder of the dissertation is structured as follows: CHAPTER II begins with an overview of two main areas of theory used to provide a framework for the study. Specifically, I draw from a critical political economy of communications approach to study corporate involvement in FLOSS projects, and discuss why such an approach can be advantageous for understanding corporate power. In addition, theories of the commons are discussed, including different types of commons and how those commons can be subject to various forms of enclosure within a capitalist system. CHAPTER II concludes with some of the previous literature used to understand corporate involvement in FLOSS projects. Having provided a conceptual framework for the current study, CHAPTER III explains the specific methodology and research methods used. That chapter also revisits the research questions that framed this investigation, as well as a discussion of the potential shortcomings of the research because of the chosen methodology and methods.

CHAPTERS IV through VI present the main findings of the study. Each chapter provides the results of the three case studies. CHAPTER IV focuses on *how* Red Hat, Inc. has become the largest and only publicly traded corporation with a business model that relies almost entirely on free software, for which the company is unable to rely on traditional copyright protections to exclude others from using the underlying source code. CHAPTER V charts the history of the Microsoft Corporation and its relationship to FLOSS. Specifically, the chapter focuses on *why* Microsoft has undergone a transformation from total opposition to FLOSS toward embracing FLOSS through the

creation of a new division of its company entirely focused on supporting FLOSS projects. Throughout that chapter, key moments in the history of Microsoft illustrate how the company relies on strong intellectual property protections to exclude others from using its software, but also how this strategy is somewhat contradictory to its own strategy of negotiating partnerships and licensing agreements that either enable widespread adoption of its software or provide the foundation for its development. Finally, whereas CHAPTERS IV and V focus on how and why two corporations are involved in FLOSS projects, CHAPTER VI illustrates what happens when a company that was supportive of FLOSS projects is acquired by another company that does not support such projects in the same way. CHAPTER VI focuses on the Oracle Corporation's acquisition of Sun Microsystems and the effect that acquisition had on three separate FLOSS projects: the OpenSolaris operating system, the MySQL relational database management system, and the OpenOffice office productivity suite of software. The goal of this chapter is to demonstrate the ways in which the FLOSS community copes with undue corporate encroachment into its projects by focusing on its ability to leverage its collective labor power to resist such influence.

Finally, CHAPTER VII summarizes the major findings and presents conclusions from the study. The intent of the chapter is to illustrate what increasing corporate involvement in FLOSS projects means for the FLOSS community and the corporations involved in FLOSS projects, and what this dynamic can tell us about commons-based peer production under capitalism. In addition, the chapter acknowledges the limitations of the study, and suggests germane areas for future research.

CHAPTER II

THEORETICAL FRAMEWORKS AND LITERATURE REVIEW

To properly ground the current study within established theories and extant literature, this chapter provides an overview of the primary areas of theory that are drawn upon to understand corporate involvement in FLOSS projects. To that end, the chapter begins with a discussion of the political economy of communication and how a critical economic perspective in particular offers a useful lens for investigating corporate involvement in FLOSS projects. Then, I discuss the ways that FLOSS has been understood theoretically by focusing on theories of the commons and the ways that information and knowledge have been understood as a type of commons. After establishing this basic understanding of the types of commons that FLOSS represents as a resource (or product), I focus more specifically on how the *processes* that enable FLOSS have been understood as a form of commons-based peer production or non-market production (Benkler, 2006). After theoretically situating FLOSS as both a product and process, I consider the ways that the commons – both common land as well as the knowledge commons – become enclosed under capitalism.

The argument presented in the theoretical overview is that even though we have a relatively robust understanding of FLOSS as both a product and process, there is still a gap in our understanding of how commons-based peer production and non-market production are enmeshed in processes of capitalist production. This project is specifically aimed at filling this theoretical gap by focusing on the different ways that capitalist firms make use of FLOSS products and processes. To that end, the chapter concludes with a review of relevant literature that has sought to provide a typology of FLOSS business models. While this typology is useful for understanding various business strategies, we

have yet to link this typology with theorizations of commons-based peer production and non-market production in a way that complicates our understanding of capitalist firms that make use of FLOSS products and processes.

Political Economy of Communication

This research project has been informed by the political economy of communication. At the heart of this approach is a concern for the "social relations, particularly the power relations, that mutually constitute the production, distribution, and consumption of media resources" (Mosco, 2009, 24). By investigating power relations, those working from a political economic perspective are concerned with the ways in which power manifests itself not just as a resource to achieve particular goals, but also as a form of control that is embedded within a broader set of social relations. As such, power itself is omnipresent throughout the social system and structures the way that certain relationships exist and tends to reproduce those structures over time.

To that end, those working within within political economy or, more specifically, a *critical* political economy of communication (CPEC), are interested in "uncover[ing] connections between ownership, corporate structure, finance capital, and market structures to show how economics affects technologies, politics, cultures, and information" (Meehan, Mosco, and Wasko, 1993, 347). However, the concerns of those working within the CPEC tradition are not only scholarly; rather, they are often concerned with *praxis* or theoretically informed practice, in which scholarly activity is pursued with the goal of achieving more just and democratic forms of communication (Mosco, 2009). Most often, this is done by exposing the ways in which power is manifested within communications industries, whereby the control of informational

production, distribution, and access or exhibition is concentrated within only a handful of corporations. These large, often multinational and trans-industrial conglomerates often hold oligopolistic power within media markets, which limits the possibility for alternative or counter-hegemonic forms of communication to take place (Bagdikian, 2004; Meehan, 2005). By limiting the extent of available alternatives, these corporations reinforce systems of ideology that, in turn, tend to reinforce institutions of cultural hegemony (Gramsci, 1971). The CPEC approach is therefore rooted in a tradition of critical inquiry, which has roots in the work of Karl Marx and his critique of classical political economy.

Of Marx and Machines...

By understanding FLOSS production from a critical and materialist perspective, which derives its force from the work of Marx, we can debunk some of the claims that digital technologies by themselves have the power to change the course of human history. The unique technological features of FLOSS – mainly, the availability of the source code and the ability to study, modify, adapt, or change the program for one's needs – is only one part of the equation and does not, in itself, constitute the core value of FLOSS. Rather, the collective labor power of the broader FLOSS community is what constitutes the true value of FLOSS. Because FLOSS production as a *process* allows for highly efficient, collaborative, and speedy development, the end *products* of FLOSS production tend to be more secure, adaptable, and progressive because they are under constant revision and improvement by members of the FLOSS community. From the standpoint of corporations like Microsoft or Oracle, which rely on the sale of proprietary software or services, FLOSS production offers an attractive option for investment because it decreases in-house labor costs while effectively outsourcing development of core

components that can be integrated within their proprietary services. The exact details of how this is done will be the focus of the following chapters.

Marx (1867) was not the first to investigate the inner workings of capitalism and the processes by which wealth is created. However, he does represent a shift in the study of political economy due to his criticism of previously existing political economic thought. His three volumes of *Capital* offer some of his most thoroughly developed arguments about political economy, and some of the key arguments made can provide a framework for understanding technology and technological change within a broader set of social relations. This background will prove useful as we consider the ways in which digital technologies have either continued or extended such relations or whether they mark a radical shift.

Marx (1906) begins his analysis of capitalism with a discussion of the commodity. He explains how life appears to be an endless procession of commodities. The commodity form, however, contains two different values: use value and exchange value. To use a simple example, an apple has a use value if I eat the apple and receive its nutrients, but it can also have an exchange value if I decide to trade the apple for some other commodity. Although a commodity may contain two values simultaneously, the commodity form is still a *product* of human labor. That is, the *process* of human labor creates *products* in the form of commodities. Although different types of commodities require different types of labor, what is common to all commodities is human labor. The value of commodities, then, is determined by the socially necessary labor time required to produce the commodities. This is the *labor theory of value*.

In early economic configurations, the trading of goods for other goods could be expressed in the simple formula: C -- C (commodity for commodity trading), which

characterizes economies based on barter and trade. In order for such a trade to take place, however, the producers of such goods need to agree on an equivalence in trade (i.e., ten apples equate to one chair). This form of trading relies on the availability of equivalent goods in order for such a market to operate effectively. In such a system, an apple farmer who wanted to trade apples for a chair needs certain conditions to be met in order to obtain the chair. First, a chair needs to be produced. Second, the chair needs to be available for trade. Third, the person who produced the chair would have a need for apples. If these criteria are met, then an exchange can occur. To reduce the uncertainty of supply and demand in such a situation, the money form (M) was introduced as a *universal equivalent* to which the value of all other commodities can be equated. So instead of trading ten apples for a chair, the apple farmer can sell the apples for $5. The money can then be used to buy a chair when one becomes available. The introduction of the money form, then, introduces a new type of market exchange, expressed as C -- M -- C (commodity for money for another commodity).

Capitalism, however, relies on larger scale production and a reinvestment in the productive process. In such a system, we can invert the C -- M -- C circuit to be expressed as M -- C -- M', whereby money is invested in the production of a commodity with the intention of re-selling it for profit (M' or, simply, more money). This is possible in a system in which certain individuals do not have any commodity to sell other than their labor power. In such a system, a division exists between those who own the means of production and those who do not. As such, the owners of the means of production will employ others who do not own the means of production. Importantly, however, the owners of the means of production will only pay laborers enough to satisfy their demand, for the ultimate goal is to increase profits. By doing so, those who own the means of

production can continuously reinvest their money into the means of production (buying more land, developing technology, etc.). Consequently, those who own the means of production extract a certain amount of *surplus value* from the productive process. Thus, society is divided into classes based on ownership of the means of production (capital vs. labor). In order to see the ways in which this form of exploitation continues, Marx suggests that we delve into "the hidden abode of production" (Marx, 1906, 195).

In perhaps the most important section of *Capital*, Marx discusses surplus value in depth, including the ways in which capital continues to realize surplus value, while labor is subjected to various forms of exploitation. Particularly relevant for the current study, however, are Marx's discussion of co-operative labor and the use of machinery. Before proceeding, it is important to note that machinery is not given a determining role, *per se*. Rather, machinery is just one way in which capital constantly reinvents itself to further exploit labor. The focus on machinery is therefore simply to frame the discussion of new digital technologies and the ways that they have been used by capital and labor alike. Although technological change constantly ensures that labor is always at the mercy of capital because labor does not own the means of production, the argument presented here is that it is entirely possible for technologies to be used as tools of resistance against unwanted encroachments by capital. When put into the service of capital, technology is constantly used to increase the efficiency of production and thereby increase corporate profits while further alienating labor from the production process. However, technology may be used by labor as a broader part of social resistance and social struggle.

Capital is constantly looking for ways to increase surplus value, which requires more productivity by labor. This can be accomplished in at least two ways: absolute surplus labor and relative surplus labor. *Absolute surplus labor* is used to describe a

condition in which labor is asked to work beyond the normally required working time in order to increase productivity. For example, workers could be asked to work through the weekend as one way of increasing productivity. On the other hand, *relative surplus labor* is realized when machinery supplements or supplants the time normally spent working by labor. In this sense, workers can still work the same amount of time, thereby keeping the wages owed to them constant, while human labor costs can be supplemented or supplanted by investment in a technology that performs the same function as human labor. With only limited exceptions, such a machine can be worked without the fear of fatigue or the need for sleep. Therefore, production increases without the need to pay additional wages to workers. This, then, is the key for understanding machinery (i.e., technological change) within the operation of capitalism: technology, when put in the service of capital, increases productivity, exploits labor, and is used for the realization of greater surplus value.

Continuing this line of argument, Braverman (1974) specifically provided an extended discussion of machinery. Braverman's task was to begin a critical history of technology, which would account for the specific ways that technology has been put in the service of capital as a way to further exploit labor. Braverman demonstrated how technological change has constantly forced labor to learn new skills in order to operate machinery. Furthermore, machinery has been used to supplement and supplant human labor, which drove members of the working class out of work and into unemployment. Anyone wishing to become employed again was forced to learn how to operate new machinery, which furthered the cycle of exploitation. Thus a vicious cycle of technology development, unemployment, and re-education was implemented as a way to constantly

reinvigorate the productive process while demanding that labor constantly acquire new skills.

Marx's analysis offers a useful framework for understanding the relationship between capital, labor, value and machinery. These four factors are all intertwined in the relationships that exist between FLOSS programmers, their collective labor power, the software they create, and the corporations that make use of their software. The labor theory of value can be used to understand why FLOSS as a *process* is so valuable. The model of FLOSS production expands the possible labor force working on a piece of software exponentially beyond those projects that are centralized within one firm. With more programmers contributing changes to the core software project, it can grow more efficiently and rapidly. The contributions to the core software take the form of fixing bugs, developing new features, or increasing functionality in some other way. In addition, and because the code is made open for anyone to view, FLOSS projects can be more secure than proprietary software as well, although this is not an absolute certainty.[3]

While FLOSS as a process has been lauded as a highly efficient, effective, and innovative production model, these treatments of FLOSS often focus on how the Internet has made such production possible or how this model of production can change the nature of commercial firms. However, the true value underlying this form of production is the cooperative labor of the software developers and programmers who contribute their labor time to the development of a FLOSS project, whether this labor is paid labor time or voluntary labor time. By understanding FLOSS from the perspective of the labor theory of value, we shift the focus away from organizational models, the nature of the

3 For example, in April 2014, a major security flaw was found in the open-source cryptography project, OpenSSL. At the time of its discovery, the flaw, known as Heartbleed, was estimated to have affected nearly 66% of all Internet users.

firm, or the technology that makes such coordinated labor possible. Instead, we can focus on the people who actually perform the work and recognize that they constitute a larger labor force than any one firm could possibly employ. As long as firms can continue to attract development from the FLOSS community for their projects, they will continue to enjoy the benefits of this collective labor power.

As we will see, however, corporations make use of FLOSS projects in a variety of ways. Some firms view FLOSS as an existential threat to their business model and use specific strategies to combat FLOSS production. Others have begun to embrace FLOSS because of the efficiency of its productive process and the effectiveness of its products even if they were once vehemently opposed to FLOSS. Finally, firms like Red Hat have found a way to turn completely free software into a successful business model.

Communication Labor, Free Labor, Digital Labor

A critical understanding of capitalist production, and particularly its consequences for labor, is useful for understanding the ways that information and communication technologies (ICTs) operate today. Political economists of communication have called for increased attention to be paid to communication laborers (McKercher and Mosco, 2007; Mosco, 2006). Communication labor encompasses a wide variety of labor, including those who work directly in various media industries (i.e., television, film, music, video game, and software industries, etc.), but it also includes various types of knowledge work, digital labor, and types of free labor (McKercher & Mosco, 2007; Scholz, 2013; Lazzarato, 1996; Terranova, 2004).

The terms “immaterial labor” and “digital labor” have found increased currency in contemporary debates about online life. FLOSS labor can be viewed as a form of

"immaterial labor" insofar as the final products of work are "immaterial products such as knowledge, information, communication, [or] a relationship" (Hardt & Negri, 2005, 108). The term "immaterial labor" was first introduced by Lazzarato (1996) and has since been debated by critical scholars.[4] Similar debates have occurred within critical scholarship circles about the nature of "digital labor" (see Scholz, 2013). The primary concern with these debates has been the nature of work and labor within the information, knowledge, and communication industries with a particular focus on forms of unpaid labor occurring online (see Andrejevic, 2007, 2012; Fuchs 2012). In these cases, users' online behaviors are tracked and can be transformed into an audience commodity in the same way that Dallas Smythe (1981) identified with broadcasting. Whereas Smythe argued that media programs constitute a "free lunch" used for producing audiences for advertisers, the same occurs online where companies and others seek the attention of users while data is collected about users' browsing habits. As most of us spend an increasing amount of time online during both work and non-work time, our digital labor – socially necessary time spent online – offers a more sophisticated form of the audience commodity as browsing data is extracted and transformed into value by service providers and other third-party elements (Fuchs, 2011a).

Similarly, Schiller's (1999) "digital capitalism" approach demonstrates how the growth of digital networks originated within the context of neoliberal policy in order to expand marketing opportunities across the globe. In this sense, digitally networked technologies function merely as another way to expand capital's reach across time and space, while decreasing the amount of time necessary to send and receive information

4 A fully developed account of the digital labor debates is not offered here, especially because it has been applied to many different types of activities online. However, for a critique of "immaterial labor" as an analytical concept, see Sayers, 2007.

about markets. Furthermore, “digital capitalism” enables other types of market manipulation, especially when internetworked digital devices facilitate access to and greater control of information (Schiller, 1999; 2007). In a free market, perfect information would be a fundamental component of buyers and sellers' abilities to make rational decisions. When information can be controlled and manipulated, it completely undercuts the ability of markets to function equitably let alone perfectly as free markets (see Taibbi, 2013; Salmon & Stokes, 2010).

Similarly, Streeter (2011) has argued that the Internet and attendant romanticization of individualist entrepreneurs like Bill Gates arose within the neoliberal period. The growth of large tech firms like Microsoft contributed further to the neoliberal ethos that the romantic individualist was to be glorified along with the growth of his company. While this type of subjectivity continues today in so-called “creative industries” or in the romanticizing of start-up businesses and culture, the fragility and precariousness of these industries were laid bare when the dot-com bubble burst in the early 2000s. The optimism about the revolutionary potential of digital devices around the turn of the 21st century drove massive investment capital into dot-com companies, which created a speculative investment bubble (Cassidy, 2002). The companies emerging from that crash are now some of the most recognizable and dominant Internet-based companies today: Google, Amazon, eBay, Microsoft, and Yahoo!

That said, however, digital technologies have made it possible for diverse groups of people across vast distances to connect with one another in new ways and to produce, or remix, cultural artifacts. For this reason, others have celebrated rather than critiqued the Internet and digital technologies. Within this optimistic camp, scholars like Henry Jenkins (2006) celebrate “media convergence” and “participatory culture.” For Jenkins,

media convergence is the process by which previous media formats (i.e., print, film, television, music, etc.) converge onto a digitally networked environment. The fact that a computer allows these previously diverse forms of media to be reduced to digital information, which can be distributed freely across a network, provides a watershed moment in media history. The argument is that users have the freedom to remix cultural artifacts in ways that allow them to participate in diverse meaning-making practices because convergence enables such interconnection. To illustrate this, Jenkins provides examples of fan communities who choose to write original stories using characters from media franchises like *Harry Potter*. This type of fan fiction can feature Harry Potter engaging in any number of different scenarios created by “fan-fic” authors. Thus, media convergence, enabled by interconnected digital technologies, engenders a participatory culture in which meaning-making resides within online communities.

While Jenkins celebrates the freedom for creative cultural expression, others have focused on what digital technologies mean for our understanding of economics. Tapscott and Williams (2006) argue that the lessons to be learned from Wikipedia mean that projects of mass collaboration have literally "changed everything" (which is included in the title of their book). The lesson is that businesses can learn from these changes in order to position themselves for the future. In addition to these works, Yochai Benkler (2006) offers perhaps the most sophisticated exposition of what digitally networked technologies mean for economics, politics, and culture. Specifically, he focuses on the greater degrees of freedom, autonomy, and creativity that are made possible by such technologies. In response to these celebratory approaches, however, critical political economists have offered more sobering accounts of how the Internet and digital technologies have been used in the service of global capitalism.

In sum, we can see two distinct perspectives on digital technologies. On the one hand, there are those scholars who celebrate the cultural practices made possible by digital technologies. On the other, there are those scholars who claim that digital technologies originated with the intent to more fully network the global capitalist system and have only exacerbated previously existing inequalities. In the interest of reconciling these two approaches, at least in some way, we can accept some of the claims of the digital celebrants. Mainly, we can acknowledge that the Internet and digital technologies have made it possible to connect and collaborate with others in novel ways. Interconnection can make possible mass collaboration, commons-based peer production, as well as both celebratory and critical forms of meaning-making. In addition, these changes have caused us to rethink some of our assumptions about economic behavior and the motivation for contributing to collaborative projects. However, although technology has changed, the underlying class distinctions and social antagonisms that lie at the heart of capitalist development have not changed. In other words, no matter how purportedly revolutionary or novel the technology, the technology exists within a capitalist system that has certain well-documented tendencies that cause it to remain relatively constant.

In this sense, Raymond Williams (1975) asserted that technology itself does not constitute a determining factor. Rather, technologies are situated within a social system that has the ability to shape how a technology is used. What matters is not the technology itself, but how it is *used*. Technology, for Williams, is just one part of a broader social struggle that may be put to alternative uses that were previously unforeseen by its inventors. While the Internet may have originated with the intention of facilitating greater interconnection within the global capitalist market, digital technologies may also be used to disrupt the flow of global capital in other ways (see, for example, "Operation

Payback").[5] These alternative uses of technology become important particularly when a commons-based resource is threatened. To understand how FLOSS has been understood as a commons-based resource, the following section provides an overview of the commons as well as commons-based peer production and the consequences such practices have for the nature of capitalist firms.

The Commons

The concept of the commons has been used to describe FLOSS projects (Benkler, 2006; Lessig, 2006, 2001). However, FLOSS represents a particular type of commons: knowledge or digital commons, which have different characteristics from the more traditional meanings ascribed to the term. In medieval England, the commons referred to a portion of land owned by the lord of the manor, in which certain tenants had the right to use the land for their needs. This included cultivating the soil, producing crops, allowing livestock to feed, and other activities. The concept has since been expanded from this very specific meaning to encompass any resource that is owned by a community or to a resource that may be accessed by a broader community of people. The concept of the commons was critiqued most famously by Garret Hardin (1968), who developed the "tragedy of the commons" argument. In Hardin's critique, he argued that the commons were ultimately unsustainable. By using the example of sheep herders allowing their sheep to graze on a commonly owned pasture, he argued that each sheep herder, acting in his or her own self-interest, would want to increase their flock. As more sheep are added to the pasture, it would eventually become depleted of its natural resources.

5 "Operation Payback" was a series of coordinated attacks against opponents of Internet piracy carried out by Internet activists operating collectively under the name, Anonymous. For more information about the operation, see the Wikipedia article: http://en.wikipedia.org/wiki/Operation_Payback (last accessed July 30, 2014).

One of the more robust contributions to theorizing the commons comes from Elinor Ostrom (2004, 1990). Ostrom (1990) provided some nuance to the way that we understand commons, especially because they were often placed in a binary opposition: either state provision of common property (socialism) or private property ownership (capitalism). Ostrom focused on the diverse ways that different commons – fisheries, waterways, tribal lands – are managed. In addition, Hess and Ostrom (2007) critiqued Hardin's argument against the commons on two points: first, Hardin assumes that the sheep herders are acting according to the principles of neoclassical economics and are individually acting in their self-interest rather than allowing for forms of common governance, whereby concessions are made to the other sheep herders. Second, Hardin frames the issue within the binary choice between socialism and capitalism described above. However, the framing is fallacious for a couple reasons. The commons under feudalism were owned by a private individual and not the state. Furthermore, Ostrom (1990) demonstrates how different types of commons can be governed collectively so that the individual short-term gains can be compromised for the long-term survival of the common resource.

Table 2.1 illustrates the possibilities for commons ownership by using a simple matrix of two factors: *rivalry* and *excludability*. *Rivalry* refers to the extent to which a resource is finite or requires reproduction. Highly rivalrous goods tend to be finite objects like apples, which need to be planted again in order to reproduce the crop, while low rivalry goods tend to be intangible goods that can be reproduced without much additional cost, like ideas, information, or knowledge. *Excludability* refers to the extent to which an owner of such goods can exclude others from accessing or using that good. Highly excludable goods are protected by private property rights, whereas goods with

low excludability may be used by anyone. Following from these terms, the matrix for rivalry and excludability would look something like this:

Table 2.1. Possibilities for Common Ownership

		Excludability	
		High	Low
Rivalry	High	Individual property (finite resource)	Common property (Infrastructure)
	Low	Intellectual property (books, music, consulting)	Knowledge commons (FLOSS)

Source: Table adapted from Hess & Ostrom (2007) and Frischmann (2012).

FLOSS represents a knowledge commons in which knowledge – in the form of source code, README files, software packages, and the shared documentation required in collaborative production – is freely available for anyone to use and at no additional cost for reproduction. One of the unique characteristics of free software as a knowledge commons is that it avoids the free-rider problem, whereby someone who consumes or uses a resource does not give back to the community. An example is the Linux-based operating system, Ubuntu. I currently use Ubuntu, but did not need to pay for it, nor any of the software included on my computer. While I may not yet have the skills to make full use of the options available to me (i.e., writing or adapting code, tinkering with software packages, etc.), I can still use programs and report any flaws or "bugs," I encounter while using the software. I can report these bugs directly to the development community when I encounter them, or I can choose to share certain data about my operating system's performance with the community upon installing it. When reported, someone within the community can work on fixing the issue and ultimately submit his or her fix to the project manager for inclusion in a subsequent release of the software, or the fix may be distributed as an update to all users. This process is reflective of the adage

"with many eyes, all bugs are shallow," (Raymond, 2000) which makes it possible for the programs and operating system to maintain a high quality over time. In effect, my use of free software serves as a form of quality control.

Commons-Based Peer Production

Because FLOSS exhibits the unique characteristics of knowledge commons and because FLOSS production takes place within a digitally networked environment, some scholars have focused specifically on the production process enabled by digitally networked technologies (Benkler, 2006). Specifically, Benkler (2006) highlights the ways in which *commons-based peer production* constitutes a new form of organization that is "radically decentralized, collaborative, and nonproprietary; based on sharing resources and outputs among widely distributed, loosely connected individuals who cooperate with each other without relying on either market signals or managerial commands" (60). Benkler positions social production in general and peer production in particular in contradistinction to market-based production, arguing that these forms of production constitute a form of non-market production. While these spheres are not mutually exclusive, Benkler argues that diverse forms of non-market production, like FLOSS, have the capability to influence market production.

Peer production can challenge market-based production in at least a couple of ways. First, peer production can develop goods that will compete directly with those produced by commercial firms. In this case, the commercial firm has a few different options: compete, do nothing, or adopt and adapt. If the firm chooses to compete, it will simply be required to somehow create a better product than that offered by the non-market rival, although this may come at considerable cost to the firm. On the other hand,

the firm can choose to do nothing. In this case, the firm is basically relying on the belief that its products are superior to the non-market option and that the non-market option will not gain additional market share. This is a risky strategy for the commercial firm. In the event that the non-market option does gain an increasing share of the market, the commercial firm, or at least its product that directly competes with the peer-produced option, runs the risk of becoming obsolete. The third option is to adapt to the changing forces in the market by adopting some of the strategies of the non-market forces. This type of strategic reorientation to non-market forces can have the consequence of altering the basic structure of an organization. As Benkler (2006) notes,

> As the companies that adopt this strategic reorientation become more integrated into the peer-production process itself, the boundary of the firm becomes more porous. Participation in the discussions and governance of open source development projects creates new ambiguity as to where, in relation to what is 'inside' and 'outside' of the firm boundary, the social process is (125).

Altering the firm's position in relation to peer production, which exists outside the firm, arguably offers a higher form of risk for the firm. The firm gives up a certain level of control over the production process. The traditional view of a firm's control over its informational resources or, more specifically, knowledge, is that knowledge can be viewed as an asset to be managed as an investment (Machlup, 1962). However, the peer-production process in general is seen as far more innovative and efficient than centralized production, including outside the realm of software production (Von Hippel, 2005). As a knowledge commons, FLOSS advocates encourage users to tinker, adapt, improve upon, or otherwise create something new. In this sense, FLOSS projects rely on intellectual property rights that allow users to make changes to a project. Proprietary and closed forms of production rely on strong intellectual property protection and the ability to

exploit those property rights across a variety of platforms. For example, before contributing to Linux kernel development, Microsoft was (and, in many ways, still remains) notorious for particularly strong protection of its intellectual property. Microsoft's model of production demands that only employees of the company have access to and work with their proprietary code. This differs from the open source model, whereby small, incremental changes to the open code are released early and often so that many eyeballs can study the code and make improvements. Von Hippel (2005) argues that innovation is much more effective in this latter model. Although he does not limit his analysis specifically to software development, Von Hippel argues that users of products like bicycles and surfboards routinely customize or adapt such products to their particular needs. When someone buys a bicycle, he or she is free to add or remove other parts or components of the bicycle to fit his or her particular need. Indeed, it would be absurd to think of a bicycle that was protected in such a way that did not allow users to change a tire. Keeping products like bicycles or software open enough for users to customize, adapt, modify, or improve upon fosters a system of innovation that is much more connected with users' unique needs. This type of open innovation is opposed to closed and proprietary innovation, which is driven by a single corporation's in-house capability, potential profitability, and perceived market need rather than real user demand.

Technologies that are sufficiently “open” enough to allow for this type of tinkering and adaptation are known as *generative* technologies (Zittrain, 2008). Zittrain (2008) identifies five principle factors in measuring the generativity of a technology. First, how extensively does the technology leverage a set of possible tasks? In other words, the more functions that a particular technology can serve is directly related to the extent to which a technology can produce change. More possibilities equate to greater

opportunities for change. Second, how well can the technology be adapted to a range of tasks? In other words, how easily can the technology be built upon or modified? Third, how easily can new contributors master the technology? Fourth, how easily accessible is the technology to those ready and able to build on it? Finally, how easily can the changes be transferred to other users and, especially, non-experts? By applying these five questions, we can measure the extent of a technology's generativity.

Importantly, Zittrain's (2008) argument is based on emerging trends that he sees as threatening the generativity of the Internet. Specifically, he identifies three specific ways that the generativity of the Internet is being attenuated. The first trend is *tethered appliances*, which refers to the centrally controlled information devices we use to access the Internet (i.e., mobile phones, gaming consoles, and tablets). While tethered appliances make it very easy to assure functionality and distribute updates as they become available, users are generally not allowed to make changes to these devices or the software running on them. Control of the device is centralized by the vendor. Zittrain argues that this is, on balance, a problematic trend for two reasons. This increases the possibility for regulating both the Internet and the devices used to access the Internet, while also decreasing the possibility for disruptive innovation to occur. In effect, tethered devices enable a system of more complete surveillance and control so that unintended uses or modifications of the technology become criminal activities.

The second trend that Zittrain sees as threatening the generativity of the Internet is through *software as a service* (SaaS). Whereas his argument against tethered appliances focused on hardware, Zittrain's argument against SaaS focuses on software. In SaaS, the storage and maintenance of software becomes centralized by a vendor. While this can ensure functionality and ease the distribution of security updates, the end users do not

have control over their software. They cannot study, modify, adjust, or make changes to the software for their own purposes. In effect, the user's software is stored in “the cloud,” which places it out of the control of the user. What makes SaaS even more problematic is that user data is sent to the software vendor, effectively serving as a form of spyware or surveillance of users' activities. Zittrain uses the metaphor of a walled garden to illustrate how SaaS functions. In effect, the carrier or service provider has control over applications, content, or media, while restricting convenient access to non-approved applications or content. This is in contrast to an open platform, where users are granted unrestricted access to applications or content.

Finally, the third trend threatening the generativity of the Internet is *perfect enforcement*. This trend synthesizes the concerns of the previous two – tethered appliances and SaaS – to identify the broader concerns of operating under such a system and the ways that user behavior can be controlled. Vendors may preempt unforeseen or unintended uses of technology by placing greater protections on the technology. For example, stronger intellectual property protection can restrict user behavior by criminalizing certain uses. In addition, vendors may issue specific injunctions against certain types of behavior. These would take the form of tailored remedies to any issues after they arise, such as security updates, fixes, or retroactive edits of software. Finally, the last way that perfect enforcement is made possible is simply through surveillance. Vendors can gather data about user practices and, perhaps, adjust future designs to cater more directly to user preferences. For example, Apple gathers data on which applications are downloaded onto a user's iPhone.

Zittrain's argument about the future of the Internet is important, especially if we consider the ways in which generativity is important for innovation. But, even on a more

general level, Zittrain's arguments can be contextualized within broader considerations of the way we frame information as a resource. Fritz Machlup (1962) was one of the first scholars to propose that knowledge could serve as an economic resource, and Machlup's work was one of the first to popularize the idea of the information society. However, knowledge and information are typically viewed from a supply-side perspective, especially in economics literature that treats these factors as investment costs for the firm. Arguing from an alternative perspective, Frischmann (2012) suggests that we can view knowledge, information, and cultural resources as a form of intellectual infrastructure. Doing so will position these resources as "basic inputs into a wide variety of productive activities," which "often produce public and social goods that generate spillovers that benefit society as a whole" (Frischmann, 2012, *xii*). Such an argument resonates nicely with the arguments in favor of promoting commons-based peer production for the purpose of enabling greater innovation (Benkler, 2006; Von Hippel, 2005). By framing knowledge and information as an infrastructural component of social development, protecting the knowledge commons becomes crucially important to the survival of commons-based peer production.

The concept of the commons is useful for thinking about informational resources. Given the increasing interconnectivity between people across vast spatial boundaries with the ability to communicate and collaborate in online environments, maintaining a base of commonly held resources that can be used for peer-production remains a central concern for facilitating more open and democratic forms of communication. This is particularly the case because the commons are subjected to the threat of enclosure, whereby the commonly-held resource is privatized in a such a way that the right of access to the commons is stripped away. Exactly how this occurs, however, differs depending on the

type of commons under consideration. To explain how enclosure threatens different types of commons, the following section focuses on enclosure.

The Threat of Enclosure

The commons are generally held in contradistinction to private property. In other words, once the commons become commodified or privatized, they cease to be commons and are in the service of capital. The process by which commons become transformed into private property is known as *enclosure*. Historically, the enclosure of common land in England took place in varying degrees between the 15th century to the 19th century.[6] Enclosure took various forms throughout this period, including voluntary enclosures, forced enclosure, parliamentary legislation, and others. Throughout this process, ownership of common land was transferred to private owners, who then had the right to restrict access to the land. This effectively ended the open field system, whereby commoners had a traditional right to use open fields for feeding livestock, farming, or harvesting from the land. While historians still debate the extent to which enclosure exacerbated class divisions and played an integral role in the development of capitalism in general, the process nonetheless drastically affected the relationship between commoners, capitalists, and the commonly held resources that once provided a means of subsistence for commoners. Moreover, the state played a crucial role in facilitating enclosure through the Enclosure Acts, which were passed between the 18th and 19th centuries in England and Wales (Polanyi, 2001).

6 A fuller historical account of English enclosures is not possible here, especially because of the diverse ways that common lands were enclosed. For some interpretations of this process, see Neeson, 1993; Thompson, 1966; and Marx, 1906, especially Chapter 27: "Expropriation from the Agricultural Population from the Land," which is freely available at http://www.marxists.org/archive/marx/works/1867-c1/ch27.htm

Enclosure of common land was accomplished by literally erecting fences around previously open fields. Enclosure of knowledge commons, however, depends on restricting access or prohibiting certain uses of informational resources. James Boyle (2003) refers to the process of enclosing the knowledge commons as the Second Enclosure Movement, whereby increasingly protective intellectual property rights are restricting access to those things which were once considered common property.

Focusing more on the consequences of the enclosure of digital spaces, Mark Andrejevic uses the term *digital enclosure* to refer to the process by which two distinct classes are formed online: “those who control privatized interactive spaces (virtual or otherwise), and those who submit to particular forms of monitoring in order to gain access to goods, services, and conveniences” (Andrejevic, 2007, 3). In other words, Internet users, as a class, have nothing to sell but their data, which serves as a form of value production for Internet Service Providers (ISPs), which represent a class that controls the means of digital production. In this sense, the ISPs can restrict access to their sites unless users agree to the Terms of Service (ToS) or End User Licensing Agreement (EULA). These non-negotiable contracts place restrictions on how users may interact with the site. The effect of these agreements is to enclose informational resources, which are controlled by ISPs.

Concerns about digital enclosure are conceptually similar to Zittrain's (2008) arguments about how the generativity of the Internet is being threatened. In addition, Tim Wu (2010), argues that new technologies generally follow a pattern, which he simply calls “The Cycle.” The cycle begins when a new technology is introduced and a relatively chaotic period of experimentation and innovation occurs. Gradually, however, as multiple producers vie for control of the technology and, by extension, control of an

industry, a free, open, and competitive market tends to become institutionalized when one or a few firms control the vast majority of production. In such a state, competition becomes moribund and the dominant firms strategically work to maintain the status quo until a new disruptive technology enters the market and the cycle from an open to controlled market repeats itself. Murdock and Golding (1973) describe a similar situation, whereby media industries move from differentiation to concentration:

> Firstly, small-scale or personalized production of a cultural product expands. Distribution and selling become separated anc commercialized. As new technology enters the medium, production becomes industrialized and consumption becomes large-scale and impersonal. This process of differentiation is succeeded by a period in which the growth of the industry reaches saturation and is hit by a series of pressures due to rising costs, declining revenue, and a changing pattern of demand...The final stage in this sequence involves a developing tension between new technological potentialities on the one hand and economic concentration on the other (207).

Notably, for Murdock and Golding (1973), the focus is on general trends within industrial economic activity, and the authors situate new technological development between its potential for democratic use and its use in the service of capital. For Wu, on the other hand, the focus is on the technology itself and the ways that technology is used. Wu warns that the cycle of the Internet is trending toward a closed market, whereby it is being subject to greater regulation by governments and fewer big firms are wielding power. Indeed, McChesney (2013) demonstrates how a few dominant companies now control much of what takes place online. These companies are those that survived the dot-com crash of 2001 and have become recognizable names: Google, Amazon, eBay, Apple, Microsoft, and others. All of this suggests that the digital commons are indeed becoming enclosed in certain ways.

Digital enclosure would seem to follow a similar pattern to more traditional forms of enclosure. However, the process or act of enclosing a commons generally denotes a linear process with a predefined outcome. As capitalist relations expand, resources generally move from open to closed systems. In this sense, resources are generally thought of in dichotomous terms as either open or closed systems. While a finite resource such as land may be thought of in these terms, knowledge or digital commons have certain unique characteristics that resist such an easy interpretation. Knowledge can be reproduced and distributed more easily than finite resources and, as such, knowledge commons can contain elements that are proprietary as well as elements that are protected as a commons-based resource. Indeed, this is most often the case for FLOSS code that is supported by or used within proprietary software companies. This blending of both commons-based and proprietary resources can lead to struggles for control of informational resources, which are often legal battles for control over intellectual property.

As Rossiter & Zehle (2013) argue, however, the commons are not purely "given as a fragile heritage to be protected" against enclosure, but they must be actively constructed. FLOSS communities actively produce knowledge commons as code is produced and licensed under intellectual property licenses that permit users to use the code and adapt it for their own purposes. These alternative intellectual property licenses take many different forms. The original copyleft license to see widespread use was the GNU General Public License.[7] Other notable examples of alternative intellectual property licenses, particularly because of their widespread use, are the many variations of

7 The text of the GNU General Public License (GPL) can be found at http://www.gnu.org/copyleft/gpl.html (last accessed July 7, 2014).

the Creative Commons licenses,[8] which allow varying levels of use for the protected property under conditions set by the creator. For example, users may make their creation freely available and permit others to use it, as long as those users provide attribution to the original author. In addition to these licenses, certain companies have created licenses that have different levels of restriction and permission. Most often, these licenses are designed with a particular goal in mind, with the company wanting to allow certain uses of the code while protecting against others.[9] As CHAPTER VI demonstrates, the licenses created by corporations can often lead to conflicts over commons-based resources.

As an increasing number of corporations are choosing to get involved in FLOSS projects, there is a risk that FLOSS project development may increasingly be driven by corporate imperatives. In the final section of this chapter, the different types of involvement that corporations can have with FLOSS projects are discussed. The purpose of this final section is simply to introduce a typology for understanding these dynamics. While the typology identifies general tendencies for corporate involvement in FLOSS, this does not mean that contradictions or differences cannot be found within particular case studies. Indeed, a more in-depth discussion of the ways corporations are involved in FLOSS projects is reserved for the following chapters and includes the ways that corporations have profited from FLOSS, the ways they have tried to fight FLOSS projects, and the ways that the broader FLOSS community has resisted unwanted corporate encroachment into their projects.

---

8 The Creative Commons Licenses can be found at http://creativecommons.org/licenses/ (last accessed July 7, 2014).

9 A full examination of the rights granted by different types of alternative copyright licenses is beyond the scope of the present study. Certain licenses will be discussed in the case study chapters, but Wikipedia features a good comparison of free and open source software licenses for those who are interested. The Wikpedia page can be found at http://en.wikipedia.org/wiki/Comparison_of_free_software_licences (last accessed July 7, 2014).

Open Source Business Models

The previous sections of this literature review focused on the ways in which FLOSS as a knowledge commons is beneficial for development, production, innovation, and democracy. I also discussed why a political economy of communication approach can be useful for understanding the ways that FLOSS as a knowledge commons may be subject to undue corporate influence and, perhaps, enclosure. In this final section, the focus is on how FLOSS has been used by corporations, which establishes a framework for understanding the different ways that businesses have tried to profit from involvement in free software. This framework relies heavily on the typology developed by Deek and McHugh (2008), since they provide one of the few attempts at categorizing the different ways in which businesses have approached FLOSS.

As part of their broader treatment of open source software, Deek and McHugh (2008) develop a typology of open source business models (Deek & McHugh, 2008, 272). The typology contains five different models that have been used in trying to profit from FLOSS. Table 2.2 provides an illustration of this typology, providing the types of business strategies employed, a description of the strategy, and an example of a company or product that is representative of the strategy.

The first business model relies on *dual licensing*, in which the owner of copyrighted software provides free and open distributions for nonprofit users but requires for-profit customers to pay a fee to use the software. The exemplary case here is MySQL, which is an open source database management system. The company provides a free version of its software under the General Public License (GPL), which stipulates that any derivative software using the GPL-licensed software must also be made available under the same license. MySQL also provides an advanced commercial version of its

software to for-profit corporations, which can be customized to the users' specific needs or integrated with that company's proprietary software.

Table 2.2. Types of Open Source Business Strategies

Business Strategy	*Description*	*Examples*
Dual Licensing	Owner of copyrighted software provides free and open distributions for non-profit users, but requires for-profit users to pay a fee to use the software.	MySQL
Consulting	Company assists other companies with planning, strategy, and implementing appropriate open source solutions within their business.	OSSCube, Olliance Consulting (a division of Black Duck Software), LQ Consulting
Distribution & Services	Company provides services for non-expert computer users by handling the compilation of stable, updated, and prepackaged software suites that are distributed to users (clients).	Red Hat, Canonical
Hybrid Open/Proprietary – Vertical Development	Using open source software as a base upon which proprietary software can be built.	Google, Sun Microsystems (i.e., StarOffice and OpenOffice)
Hybrid Open/Proprietary – Horizontal Arrangements	For-profit company becomes directly involved in supporting open source projects to supplement its own business operations.	IBM, Microsoft

Source: Table is adapted from Deek & McHugh (2008, 272).

The second type of business model is one in which a company provides *consulting services* for FLOSS. Quite simply, companies that adopt this model assist other companies with planning, strategy, and implementing appropriate open source solutions within their business models. Among other things, Black Duck Software provides consulting services through its Olliance Consulting division.

The third business model is one in which a company provides *FLOSS distributions and services*, and the exemplary company here is Red Hat. Unlike MySQL, which owns the copyrights for its software, Red Hat creates and provides its own distribution of Linux. In addition, Red Hat provides training, eduction, documentation, and support for its Linux distribution. In other words, Red Hat provides a service for non-expert computer users by handling the compilation of stable, updated, and prepackaged software suites to be distributed to users. In some ways, then, Red Hat behaves similarly to a proprietary software provider, except that it does not own the intellectual property rights for the software it sells and services. Rather, the company sells and provides its own Linux distribution, which it is able to do because of the open licensing model of Linux.

Whereas the first three business models are solely related to FLOSS, the remaining two rely on a hybrid of both open and proprietary software. The fourth model is a *hybrid of both proprietary and open software* that relies on *vertical development with FLOSS*. Vertical development means using open source software as a base upon which proprietary software can be built. One of the major corporations that uses this model is Google. In fact, Google does not sell its software at all; it develops and maintains its own software in-house, while selling services provided by its software to other customers. Of course, Google's search engine is proprietary, but Google uses the Linux core to support its proprietary search services.

The final model is a *hybrid of proprietary and open software,* but one in which the company relies on *horizontal arrangements*. This is the business model that lies at the heart of this dissertation project. In these relationships, for-profit corporations become involved in open source projects. Drawing from Fogel (2005), Deek and McHugh (2008)

claim that the reasons for corporate involvement are diverse, but include everything from spreading “the burden, cost, and risk of software development across multiple enterprises to allowing companies to support open source projects that play a supportive or complementary role to their own commercial products” (277). IBM is one example of this type of business model. For example, IBM's WebSphere application, which enables end-users to create their own applications, was built using the Apache web server, which is open source. Thus, by supporting open source projects like Apache, IBM is indirectly supporting its own interests. Furthermore, IBM directly competes with Microsoft as a platform for applications. Because IBM supports Linux, it is not only investing in the reliability of its own products but may simultaneously weaken Microsoft's market position, especially because Linux is also a direct competitor of Microsoft.

In sum, then, this section has discussed how FLOSS has been used in differing ways by drawing on the typology developed by Deek and McHugh (2008). The most fruitful area of study for the purposes of this project was the hybrid open/proprietary model that relies on horizontal arrangements, although other projects are discussed, like MySQL, which represents other types of business strategies. The corporations that rely on horizontal arrangements are most interesting because of their direct involvement in FLOSS projects. Thus, these companies need to maintain a good relationship with the broader FLOSS community. When the norms of the community are violated by a company, the community can abandon a project, which can effectively end commons-based production on the project. In this sense, the FLOSS community leverages its collective labor power against undue corporate influence in its commons-based resources. This was the case when the Oracle Corporation acquired Sun Microsystems. This case will be discussed in greater detail in CHAPTER VI. For now, however, it is important to

note the two different examples of companies using hybrid horizontal agreements to two different ends. In the case of IBM, the company was able to maintain a relatively stable relationship with the open source community. In the other, Oracle overstepped its bounds by violating the norms of the community. As more and more corporations become involved in FLOSS projects, the relationships that exist between the community and the corporations that rely on their collective labor power will be subject to changes. These dynamics are the primary concern of this dissertation.

Summary

This chapter focused on how FLOSS production can be understood as both a *process* as well as the *products* created by that process. The focus was on how FLOSS can be understood as a commons and, more specifically, as a knowledge commons with certain unique characteristics. Mainly, they are resources characterized by low rivalry and low excludability, which do not make them susceptible to the same types of enclosure that befell common lands. Rather, knowledge commons can, at times, contain both proprietary and nonproprietary elements. This, in turn, can lead to conflict within the commons. Most often, this conflict comes in the form of licensing disputes when a corporation makes an ownership claim to the commonly held resource. In this sense, knowledge commons like FLOSS may be susceptible to total enclosure but, more often, are incorporated into a corporation's broader strategy. As such, corporations see a certain value in FLOSS production and FLOSS projects.

Because this dynamic can lead to conflict and contradiction within the commons, the political economy of communications approach can be a useful framework for understanding this phenomenon. Informed by critical political economy, this approach

focuses on the ways by which corporations wield power over communication resources. Drawing from Marx's dialectical understanding of labor and capital, the critical political economic approach to the study of commnication resources stresses the primacy of human labor that underlies communicative resources. Rather than focusing on the innovative or purportedly revolutionary nature of the technology itself, critical political economy responds by refocusing our attention on the specific cultural practices and collective labor that make up both the technology and its attendant practices. Indeed, this chapter argued that FLOSS production is powerful because of the scale of its collective and co-operative labor power. Furthermore, many of the unique characteristics of FLOSS labor make FLOSS projects an attractive option for corporations that are looking to harness such power. On the other hand, increasing corporate presence in the commons may have detrimental effects for the broader FLOSS community.

This points to a gap in the previous theoretical literature on FLOSS products and processes. FLOSS products have been understood as a form of knowledge commons, whereby anyone has the right to study, modify, adapt, or otherwise make changes to the resource to suit his or her own needs. The productive processes used within FLOSS communities have been theorized as commons-based peer production, which enables forms of non-market production. Finally, some of the previous literature has attempted to arrive at a typology of different strategies that businesses can use to profit from FLOSS products and services. What becomes clear, however, is that these treatments either overgeneralize and fail to address the idiosyncrasies of various types of FLOSS projects, or they establish hard boundaries between market-based and non-market production. This study seeks to complicate these understandings of FLOSS by providing examples of how corporations are making use of commons-based resources and commons-based peer

production by becoming involved in FLOSS projects. Rather than a unified theory to explain these strategies, the following chapters provide only certain examples of the different ways that corporations have approached involvement in FLOSS projects. Furthermore, this project seeks to identify strategies used by the FLOSS community to resist undue corporate influence. Before presenting these case studies, the following chapter explains how the research was conducted.

CHAPTER III

RESEARCH QUESTIONS AND METHODOLOGY

CHAPTER II laid out some of the main issues and contradictions that are at the center of this project. Specifically, I explained why a critical political economy of communication approach is a particularly useful research framework for addressing these questions because of its focus on the production, distribution, and exhibition or consumption of communications resources. More specifically, political economists are interested in how power relations manifest themselves within communication industries. Since FLOSS depends upon and is constituted by communicative activity, political economy is well suited to address the primary concerns of maintaining just and democratic forms of communication. Then, I focused on how FLOSS can be understood as a knowledge commons with certain unique characteristics. Finally, I complicated this framing of FLOSS resources by describing the ways that commons become enclosed and how FLOSS may be at risk of enclosure in certain ways. While CHAPTER II highlighted the central tension at the heart of this project, CHAPTER III focuses primarily on how the current project was approached methodologically and what specific research methods were used in the course of research.

To that end, this chapter begins by revisiting the research questions guiding the project. Following this review, the following section discusses the methodological approach employed in this study and the specific methods used to address the research questions. Finally, and because this study included human subjects as part of the research process, the chapter concludes with some brief information about the review and approval of this project by the institutional review board.

Research Questions

Free and open source software has attracted a great deal of attention from scholars and, increasingly, the broader business community. Most of this attention has focused on the novel productive process enabled by commons-based peer production or the potential for profitable business practices. Significantly less attention has been given to the dynamics that exist between FLOSS communities and the corporations that make use of their intellectual labor. While business models may change and adapt to emerging trends as corporations seek higher profits, the underlying labor that comprises the power of the productive process does not change – at least insofar as the products of collective intellectual labor are sold for profit. As discussed in the previous chapters, extant research has largely focused on how companies or society, writ large, can leverage the collective labor power of the FLOSS community to foster innovation or democratize productive processes. This project focused more specifically on the ways that corporations make use of this collective power in different ways and to what extent these corporations wield power within FLOSS communities to focus development on certain projects that are instrumental to the goals of the corporation. In sum, the current study is guided by the following research questions:

> RQ1: What is the relationship between proprietary, for-profit corporations and free and open source software (FLOSS) communities, and how has this relationship changed over time?
>
> > RQ1a: What are the power dynamics between corporations and the FLOSS community? In other words, which party holds the ability to exert influence on the other?
>
> RQ2: What constitutes value for each of these stakeholders? What value do corporations provide for the FLOSS community, and what value does the FLOSS community provide for corporations? Are there any external factors or other stakeholders who may profit from this relationship?

To expound on these research questions, RQ1 exists at a descriptive level: the question required an assessment of the current relationship between FLOSS communities and corporations, as well as an historical account of how this relationship has developed over time. RQ1a supplemented the descriptive information provided by RQ1 by investigating the power dynamics between corporations and FLOSS communities. The goal of this question was to determine who is able to influence what projects are undertaken, whether they will succeed, and whom they will benefit. FLOSS has been characterized as a more democratic form of organizing communication (Benkler, 2006), thus RQ1a investigated whether corporate involvement in FLOSS provides some evidence that this seemingly democratic form of organizing communication is becoming institutionalized.

RQ2 was essentially an economic question, which also required a description of how corporations and FLOSS communities provide value for one another. FLOSS communities have been described as gift economies (Söderberg, 2008) or a form of commons-based peer production (Benkler, 2006). However, the intention of this project was to critically assess these claims by determining what value FLOSS communities produce for corporations as well as what value corporations produce for FLOSS communities. While this question required analysis of what value each party holds for one another, or a *relational* value, the value *within* each community was studied as well. This question also allowed me to investigate whether FLOSS products can be called commodities in the traditional sense, and in what ways they differ, if at all. Finally, RQ2 necessitated an analysis of any additional stakeholders are involved in the relationship between corporations and FLOSS communities.

To address these research questions, a largely qualitative multi-method approach was used, including document analysis, semi-structured interviews, and some basic data mining of FLOSS projects. Since this project was concerned with gaining an understanding of the dynamics that exist between corporations and the FLOSS community, both document analysis and semi-structured interviews were used to understand corporate structures and strategies, as well as to understand FLOSS communities. FLOSS projects depend on extensive and accurate documentation to make the development of projects run effectively and efficiently, and these documents are made publicly available so that other developers can work on the project. The source code is one form of documentation, which enables users to understand how a project works, but many FLOSS projects also contain credits files, licensing disclosures, README files, and other documents that provide essential information to users. This information, as well as the information found on publicly available discussion lists, was combined with qualitative disclosures from interview subjects to understand the dynamics between the corporations and the community. Furthermore, the information gathered from these sources were combined with personal experiences using Linux and attending a variety of different events and meetings focused on FLOSS.[10] A more thorough discussion of the methods and materials used for this project follows the next section, which discusses the methodological approach.

10 Specifically, these included a trip to OSCON, the Open Source Convention, in Portland, OR, as well as involvement in Eugene Unix Gnu Linux User's Group (EUGLUG) meetings and public talks in Eugene, OR.

Methodological Approach

> Empirical observation must in each separate instance bring out empirically, and without any mystification and speculation, the connection of the social and political structure with production. The social structure and the State are continually evolving out of the life-process of definite individuals, but of individuals, not as they may appear in their own or other people's imagination, but as they really are; i.e. as they operate, produce materially, and hence as they work under definite material limits, presuppositions and conditions independent of their will (Marx, 1845, 41).

The quote from Marx comes from a section of *The German Ideology* that discusses the essence of historical materialism. The quote represents a *methodological* approach to inquiry that is guided by particular assumptions about how reality can be understood and described. The quote also nicely summarizes the goals of researchers working within the critical political economy of communication – that is, to connect the definite processes of material production with broader social and political structures.

Most often, the inquiries of critical political economists of communication are directed at large corporations that hold extensive market power and the ability to influence the production, distribution, exhibition, or access of communication resources. In the process of investigation, the aim of critical political economists is to empirically investigate the material operations of corporations and connect those operations to the broader social system. The connections made to the social system can be situated within national boundaries while accounting for the attendant institutions (religious, legal, cultural, etc.) that encourage or discourage certain types of behavior, but can also be made across those boundaries (internationally, regionally, globally).

By making these connections, political economists search for the general *tendencies* of corporate behavior within a particular social system rather than seeking to

establish absolute *laws.* This allows the inquiry to remain open to the possibility of contradictory factors, while also allowing for an account of diverse corporate practices both within and across media industries. Indeed, the contradictory factors provide the illuminating moments for critical researchers, particularly because they provide opportunity for critique and resistance. To this end, critical political economists of communication have provided important critiques of media corporations, especially the ways in which they operate in conjunction with the general tendencies of a broader capitalist system. As Meehan (1999) notes, "critical scholars share an ethical obligation to produce knowledge that accurately describes the media and reveals the hidden dynamics whereby media corporations attempt to commercialize and control expression in service to advertisers and ultimately to capital" (162).

To search for such hidden dynamics, the current study employed a critical interpretive methodological approach often used by critical political economists of communication (CPEC). Maxwell (2003) describes this approach as used by Herbert I. Schiller, a pioneering scholar working within the CPEC tradition. When working from a critical perspective, one situates research findings within broader bodies of knowledge and looks for disjunctures or contradictions arising from within the field of study. These contradictions or disjunctures can provide germane moments for research, from which previously accepted understandings can be challenged and refined. In this sense, CPEC scholars tend to resist interpreting research findings according to their face value or as *prima facie* evidence. Rather, the research findings are brushed against the grain of alternative bodies of knowledge as a way to situate the results within a broader set of relationships. Similarly, Mosco (2009) describes his epistemological stance as being *constitutive*. That is, CPEC scholars resist causal, linear determinations as well as the

assumption that units of analysis are fully formed wholes. Instead, critical political economists favor an epistemological position that is based on mutually constitutive processes, which act on one another throughout various stages of formation. In this sense, the approach is dialectical in that it considers both particular and more general phenomenon as part of a totality of processes. These concerns are carried with the researcher throughout the research process, regardless of what type of evidence is being investigated or how it is being gathered.

To facilitate this type of investigation, critical political economists use a variety of methods. However, the selection of method is often driven by the amount of access that the researcher has to the subject being studied. When direct access to corporation is available, critical political economists rely on research methods such as interviewing, participant observation, ethnographic methods, and other methods that allow for direct observation of the life-processes of definite individuals as they operate or produce materially. In turn, these observations can be linked with the "definite material limits, presuppositions and conditions independent of their will" (Marx, 1845, 41). When we do *not* have direct access to corporations, critical political economists rely on documentary evidence of corporate operations and the material production taking place within the corporation. Most often, this data comes from documents that are produced by and about the corporation. To that end, the following section discusses the specific methods used in this study.

Research Methods

The methods used in this study focused primarily on document or textual analyses, but documentary data was supplemented by qualitative, semi-structured

interviews with programmers and other representatives from the open-source community. The original intent of this project was to include similar interviews with representatives from the corporations under investigation, but these plans needed to be slightly altered during the course of study after I was unable to secure any meaningful interactions with corporate representatives, particularly at Microsoft. Consequently, I needed to rely more heavily on documentary sources throughout the study.

The documentary or textual, and interview analysis methods were conducted at two levels of analysis. On the one hand, the research focused on corporations or institutions involved in FLOSS projects, but they also focused on the broader FLOSS community, including programmers or others involved with FLOSS projects in some way. In what follows, I describe how each of these research methods were used in the course of study.

Document/Textual Analysis

Documents come in many forms, but in this study, documents are defined as any artifact which has as its central feature an inscribed text which contains intentional messages (Scott, 1990). This definition is sufficiently broad enough to include many different types of documents regardless of their material basis, which includes electronic or digital sources as well as material sources. To investigate the productive processes of corporations – including its corporate structure, ownership structure, financing, joint agreements, properties, and labor practices – CPEC scholars rely on a wide variety of documents, which come from both primary and secondary sources. That is, documents may be produced directly by the corporation and serve some function within its overall

operation (primary documents), or they may documents produced by another party about the corporation (secondary documents).

Primary documents include budgets, press releases, internal memos, financial statements, web sites produced by the organization, government filings, and other documents produced directly by the corporation. CPEC scholars have historically relied on government documents to investigate corporations. For example, to research the American Telephone and Telegraph Company and the Bell Telephone System, Danielian (1939) relied on documents from The Telephone Investigation, carried out by the Federal Communications Commission (FCC) in 1935, which produced "sixty volumes of transcript, more than 2000 exhibits and more than seventy volumes of staff reports," all of which was public record at the time but remained unpublished (preface, *i)*. The documents provided some of the most detailed and comprehensive data about AT&T's sixty-year history. Although the FCC's report from the investigation relates specifically to the problem of telephone rates, Danielian used the data contained within the documents to present a social evaluation of the company's immense market power, including its corporate structure, financial data, relations with independent telephone companies, public relations and propaganda campaigns, and its influence on radio and film industries.

Similarly, CPEC scholars have historically relied on disclosures made to the Securities and Exchange Commission (SEC) of the United States. The Securities Act of 1933 requires that all publicly held companies in the U.S. disclose their properties, business activities, certain financial information, and information about company management to the SEC. Although these documents may not reveal a complete or wholly truthful state of a corporation's structure and business activities, these documents can be

used as a means to gather data that would otherwise not be available in trade press publications and other popular media sources.

The SEC filings used in this study, specifically Form 10-K annual reports, often include favorable statements about the corporation's performance as well as “forward-looking statements,” whereby representatives from the corporation make predictive statements about the company's future business plans or performance. Some of the most useful disclosures in these documents come from statements about partnerships, acquisitions, key properties, or the ownership structure of the company. These disclosures can help researchers obtain the names of the key stakeholders involved in the corporation's operation.

However, documents of this type are not required of privately held firms, including private equity firms even if these firms hold an ownership stake in the public company. As Bettig (2009) points out, private equity firms are playing an increasing role in media industries, which limits the ability of citizens and researchers to determine the activities of media corporations. Many of the private equity firms and venture capital firms discussed in this study were not necessarily central to the analysis, but understanding their involvement can illustrated the massive investment in companies that specifically cater to FLOSS projects. To the greatest extent possible, I investigated private equity firms by relying on trade press publications, news stories, and other publicly available documents.

The increasing presence and lack of information about private equity firms is not the only challenge researchers face when using documents to analyze corporations. Scott (1990) identifies four central categories to consider when assessing documents: *authenticity, representativeness, credibility, and meaning*. The concern for authenticity is

related to whether a document originates from where it is said to have originated. In other words, does the document represent a direct disclosure from the source under investigation? A highly authentic document would be primary source material that originated directly from the primary author. Conversely, secondary sources may be authentic, but those documents do not originate from an original author, which may call into question the representativeness of the disclosures. Representativeness refers to whether the information contained in the document is representative of the phenomenon under investigation. Highly representative documents contain data that provides a clear and comprehensible picture of what it claims to represent. On the other hand, a document that is not representative may only provide a small amount of data about the phenomenon under investigation and may not be representative of general trends or tendencies. Credibility is somewhat related to representativeness, but refers specifically to whether the information contained within the document is trustworthy. That is, does the document accurately represent what it claims to represent? Moreover, the information contained in the document may need to be verified by consulting additional sources. Finally, meaning is associated with whether or not the information holds any value for the researcher, as well as what meanings are associated with the document itself. In this sense, the concept of meaning is doubly significant; it refers to both the content contained in the document as well as the entire document itself within a broader context.

Two other factors affecting the reliability of documentary sources are *selective deposit* and *selective survival* (Webb et al, 1981). Selective deposit refers to the information that is contained within the document as well as what may have been purposefully left out. After all, documents are created for specific reasons, especially by

corporations. Determining what information is contained in the document may also help the researcher determine why the information is there at all. Similarly, the concept of selective survival helps the researcher determine why the document itself was created. Documents are generally created because they fulfill some function for the corporation. Why was the document created, and for what reason does it still exist? These two questions draw attention to the broader context within which the document must be situated.

These concerns about documentary evidence not only apply to official primary documents created on behalf of the corporation, but to secondary documents as well. In the course of this research project, many different types of secondary documents were consulted, including trade press publications, news articles, and online tools like blogs, forums, and *listservs* that are specifically related to the topic under investigation. The advantage of researching FLOSS communities is that nearly all FLOSS projects have unique forums, bulletin boards, or wikis dedicated to providing documentation and facilitating communication about the project. These sources typically contain repositories of the project itself, but they also offer community discussion and historical data about the project's development. This, in turn, can provide documentary evidence of ongoing and past events in a way that is open to public. For example, the Fedora Project,[11] which is discussed in CHAPTER IV, features a wiki that contains extensive documentation about the project, including news, events, recent changes, user guides, and links to various sub-projects associated with the main Fedora Project (The Fedora Project, 2014). Similar sources can be found for all the FLOSS projects discussed in this study. In

11 Additional information about The Fedora Project can be found at http://fedoraproject.org/wiki/Fedora_Project_Wiki (last accessed July 9,2014).

addition to the project wiki, I also relied on documentation from Wikipedia as well, especially for links, figures, and disclosures about Oracle's acquisition of Sun Microsystems, which is discussed in CHAPTER VI. However, rather than drawing information only from the Wikipedia pages, additional sources were consulted to confirm that the information was correct.

Using wikis for this study was a conscious decision to try to lend some credibility to the wiki as a platform for conducting research, especially research about FLOSS projects and contributors. Because FLOSS projects rely on extensive documentation to facilitate development of the project and to coordinate productive activity, FLOSS communities comprise communities of practice whose productive activities are mediated by information technology. The collective productive activity of these communities can be found, nearly in its entirety, in the project's documentation, although full documentation of the project may be distributed across numerous platforms. For example, even conversations that take place via an online chat or Internet Relay Chat (IRC) system can be archived and made available somewhere online. In this sense, these conversations can be preserved in their entirety, and function similarly to minutes that are kept during corporate board meetings. Indeed, CHAPTER VI provides a discussion of the community's reaction to Oracle acquiring Sun Microsystems. Some of the most illuminating reactions were revealed during an IRC conversation with the board members of the OpenOffice project when the community members decided to leave the OpenOffice project.

In addition to these individual project pages, specialized trade publications and news sources that cater to open source were invaluable resources throughout the research project. Specifically, publications like *Ars Technica*, *ReadWriteWeb*, *CNet*, and *ZDNet*

offer coverage of free and open source developments. Moreover, these publications provide some of the best documentation, from an historical standpoint, of corporate maneuvering within the open source industry. Although these publications do not necessarily reveal opinions about how corporate activities were perceived, they do provide information about the key individuals, the venture capital firms, and the monetary value of some of the mergers and acquisitions discussed throughout this study.

Interviews

Aside from the non-reactive or non-invasive research of document analysis, I also relied on interview data for the study. These interviews were intended to come from two different sets of people: individuals representing the corporations discussed in the study as well as members of the broader FLOSS community. As a way to solicit perspectives from the corporations, I sought interviews with those individuals working directly with FLOSS projects within the companies. For example, some companies specifically employ an open source strategist or project coordinator. Other companies, like Red Hat, rely completely on FLOSS projects to support their business model, and I attempted to communicate with anyone working for the company, especially those involved in defining a strategic vision for the company.

In order to get a sense of how corporate involvement within the broader open source community is perceived, I relied on interviews with open source programmers, advocates, or activists. These individuals were primarily from local groups like the Eugene Unix Gnu Linux Users Group (EUGLUG), but others were located regionally and even internationally. All interview participants from the broader FLOSS community

were either supporters or directly involved with the development of FLOSS projects in some way.

Recruitment Process

The initial contact with interview participants from the FLOSS community was with those individuals whom I had already established contact. From these initial contacts, I used a snowball sampling method, whereby I asked these individuals if they were familiar with anyone else who would be interested in speaking with me about the topic. After this initial round of recruiting, I then turned to local groups involved in FLOSS, like the EUGLUG. After using similar snowball sampling from within these local groups, I then relied on personal contacts. Specifically, I consulted individuals in Brazil who are involved in FLOSS communities as either researchers or activists.

For individuals representing corporations, however, I used a more selective sampling process to solicit participation. Specifically, I was looking for those individuals with knowledge of the company's operations relating to FLOSS projects. In the case of Red Hat, the entire company is related to FLOSS projects, and I was able to attempt speaking with nearly anyone as a way to understand Red Hat's involvement in FLOSS projects. In the case of Microsoft, I was looking for very specific people since Microsoft is a large company with diverse operating segments. Most of these individuals were identified by using publicly available documents, like press releases and trade press publications, which specifically identified them as being involved in the company's FLOSS-related activities. Once I had identified these individuals, I attempted to contact them through a variety of means (email, phone, social networking sites, or personal

contacts). Whenever anyone was contacted for an official interview, I used the recruitment script that can be found in APPENDIX A.

Interview Setting

The interviews for this project were conducted in settings and formats that were most convenient for the interviewees. Because the members of the EUGLUG are locally based, I was able to meet with members of the group at public locations. However, I also attended the group's local meetings whenever possible. When travel was not possible in order to conduct an interview, the interview took place via the Internet by using a voice over Internet protocol (VoIP) service, like Skype. For example, I conducted certain interviews with personal contacts in Brazil using this method. In addition, I was also able to rely on email correspondence with those contacts with whom I had already established a relationship.

Interview Participants

In the end, nearly twenty interviews were conducted with members of the broader FLOSS community. All of these participants were either actively involved in coding FLOSS projects or had worked on FLOSS projects in the past. All participants tended to be supportive of FLOSS in general, although to varying degrees. Many of them recommended that I interact directly with the communities who were currently working on the FLOSS projects chosen for inclusion in this study. When I was unable to communicate directly with members of specific projects, I relied on the documentation found on the project's web pages and other sources.

The individuals interviewed from the broader FLOSS community were tremendously helpful to this study. Those interviews provided the initial impetus for exploring additional aspects of the chapter about Red Hat as well as Oracle's acquisition of Sun Microsystems. In addition, those interviews provided invaluable insight into some of the technical features of FLOSS projects, as well as providing some technical support when it was needed.

Those interviewed had varying degrees of direct experience with the cases chosen in this study. For example, some of the interviewees were actively using or working with Red Hat's products, including both Fedora and Red Hat Enterprise Linux. Others had direct experience with Sun's products before the acquisition by Oracle, and those individuals were really helpful in understanding the links between the various properties and their fate after the acquisition by Oracle.

The one shortcoming of the interview research, however, was a lack of connection with those directly associated with Microsoft's involvement in open source projects. The interview subjects were able to provide feedback on some of Microsoft's history in relation to the FLOSS community, especially the creation of the Shared Source program, but none of them had direct access to the projects being developed at Microsoft Open Technologies. Therefore, the focus of the Microsoft chapter shifted to concentrate more on the historical trajectory of Microsoft's involvement in FLOSS projects. To accomplish this, documentary evidence was used to a greater extent in the Microsoft chapter as a way to understand the company's relationship with the FLOSS community and its reasons for doing so. Particularly helpful in this regard were the court cases and subsequent antitrust investigation by the U.S. Department of Justice (DOJ). The findings of fact and final ruling documents prepared by the DOJ in their investigation, plus additional secondary

sources about the company's compliance with the consent decrees provided detailed information about Microsoft's operations. In addition, the publicly available documents by or about the Microsoft Open Technologies division provided the bulk of the information for that portion of the study. Because the division was newly created in 2012, however, the information available about the division was still somewhat limited and changed periodically throughout the study.

Data & Analysis

In the end, the data gathered from both documentary sources as well as interviews was used to identify key moments or projects in the history of corporate involvement in FLOSS that illustrated the dynamic between the community and corporations. For the interview data, I was specifically interested in the participants' perceptions, opinions, beliefs, or other qualitative disclosures about practices associated with corporate involvement in open source software. In addition, the interviewees were asked about specific cases that exemplify some of the ways that this relationship was perceived as successful, as well as examples of when the relationship was strained and how. From these disclosures, I was looking for moments of contradiction or disjuncture from information found in documents to determine whether there was a difference between the ways that members of the community perceive corporate involvement and the way that the corporations perceive their involvement. For this study, the primary unit of analysis was the corporation, and I used the qualitative disclosures from the interviews as a way to supplement the evidence from documents.

Human Subjects Research and Institutional Review Board

Because this study involved methods for research including human subjects (interviewing), the study was reviewed by the University of Oregon Institutional Review Board (IRB) and Research Compliance Services. The protocol for this research (04222013.022) was determined to be a minimal risk research protocol that qualified for exemption from IRB review under 45 CFR 46.101(b)(2) on May 28, 2013. Further review of this research would not be required unless the research continued for more than five years. Benjamin J. Birkinbine was the primary and sole investigator for this study, and he completed the Collaborative IRB Training Initiative (CITI) on October 24, 2012 with an expiration date of October 24, 2014. All interactions with the human subjects for this research strictly adhered to the regulations and ethical considerations set forth by the IRB and Research Compliance Services.

As approved by the IRB, interview subjects were emailed the approved Recruitment Letter or Email (Appendix A), notifying them of the purpose of the study and why they were being selected. The letter also indicated that their involvement was completely voluntary and they would be able to opt out at any time. In many cases, a recruitment letter was not needed. Rather, I approached many interview subjects directly through my involvement with the Eugene Unix Gnu Linux Users Group (EUGLUG) during weekly meetings. These meetings were held for general Linux questions or discussion, but more formalized topical presentations were scheduled once per month. For example, the EUGLUG hosted speakers from nearby organizations who were involved in FLOSS in some way, and the group also held workshops on a range of topics from cryptography and Bitcoin to interfacing with GitHub. Prior to conducting any formalized interviews with these individuals, however, interviewees completed the

Informed Consent Form (Appendix B), which listed the risks and benefits of participating in the study, as well as reiterating the voluntary nature of their involvement. In addition, the Informed Consent Form contained information about confidentiality, including the right for interview subjects to remain anonymous. In the case that interview subjects preferred to remain anonymous, they were asked to select a pseudonym. If an interviewee did not select a pseudonym, one was created for use in the study (DevOp1, DevOp2, etc.).

Summary

In sum, then, the methodological approach of this research project can be described as critical interpretive. The methods used in the research process were document or textual analysis as well as interviews. I drew from both primary and secondary sources, as well as interviews with representatives from the broader FLOSS community. The goal, again, is to arrive at an understanding of the relationship between corporations and the FLOSS community.

In the chapters that follow, I discuss three separate case studies are discussed. CHAPTER IV focuses on Red Hat, Inc., which is the largest and only publicly traded corporation operating solely on free software. The primary focus of the chapter is to describe how Red Hat has been able to accomplish this. CHAPTER V focuses on Microsoft Corporation's long and checkered history with the FLOSS community. The chapter charts the history of Microsoft's involvement in FLOSS projects by focusing on the company's antithetical stance to FLOSS during its earlier years to more recent attempts to become more involved in FLOSS projects. The primary focus of the chapter is to try to understand why Microsoft's position toward FLOSS has shifted. Finally,

CHAPTER VI focuses on Oracle's acquisition of Sun Microsystems. The primary focus of that chapter is to illustrate what happens when a company that is generally not supportive of FLOSS projects acquires a company that had previously supported FLOSS projects. The discussion details the ways in which the FLOSS community coped with the corporate acquisition by discussing the OpenSolaris operating system project, the MySQL relational database management project, and the OpenOffice office productivity software project.

CHAPTER IV

FROM THE COMMONS TO CAPITAL: RED HAT, INC. AND INTELLECTUAL PROPERTY

The previous chapters focused on establishing a framework for the present study. This chapter, as well as the two subsequent chapters, offer in-depth case studies on the different ways that companies are involved in free and open source software projects. And, in turn, how the FLOSS community has responded to this involvement. This chapter begins with an overview of Red Hat, Inc., which is the largest and only publicly traded company whose business model relies entirely on free software. Within the typology established by Deek and McHugh (2008), Red Hat's business model follows the *distribution and service* model. Beyond situating Red Hat's business practices within this typology, this chapter explains how Red Hat incorporates free and open source software protected under the GNU General Public License (GPL) by using trademark law since it does not (and cannot) restrict access to the underlying source code that is used to build its products. In doing this, Red Hat has found a way to move *from the commons to capital.*

To illustrate exactly how Red Hat represents this model, the chapter is structured accordingly: first, some basic historical information about the company is provided, including economic performance data and how the company has changed over time. The specific focus is on its core commodities – previously Red Hat Linux and now Red Hat Enterprise Linux – both of which rely on collaborative commons-based peer production from within the FLOSS community. Then, the ways in which Red Hat negotiates relationships with the FLOSS community through Contributor Licensing Agreements (CLAs) are explained. These agreements separate authorship from ownership in a way that protects Red Hat against any claims to ownership by community members. Since the

intellectual property rights of user contributions are centralized within Red Hat, the company then embeds its trademarked corporate logo into the distributions it sells, which gives it the ability to restrict access to and redistribution of its commodities. Finally, the chapter concludes with reflections about the Red Hat business model and what it means for the broader FLOSS community. The argument throughout this chapter is that Red Hat's business model is based on its ability to serve as an intermediary between the FLOSS community and other businesses. However, its ability to perform such a function requires that it centralizes commons-based peer production under its corporate trademark, which the company can then use to protect its core commodities.

The Political Economy of Red Hat, Inc.

Red Hat Software, Inc. was founded in 1995. At that time, Linux, the open-source operating system was still an emerging phenomenon but was growing rapidly. Linus Torvalds released the code for his Linux kernel project in 1991. Those who supported the open-source project were extremely dedicated to its cause, but the market for software and the market for operating systems in particular was still dominated by large firms, most notably Microsoft and its Windows operating system. In 1993, Bob Young formed a company called the ACC Corporation, which primarily sold Unix and Linux related accessories and books. In 1994, Mark Ewing created his own distribution of Linux called Red Hat Linux. One year later in 1995, Red Hat Software, Inc. (simply referred to as “Red Hat” from here) was founded after Bob Young's ACC Corporation merged with Mark Ewing's company. Red Hat was founded with the purpose of making open source a commercially viable business model by lending credibility to the emerging open-source phenomenon. In effect, Red Hat was a way to bring the power of open-source to other

businesses by providing packaged solutions to customers, while funneling their earnings back into the open-source community by supporting free software projects. As Bob Young described in 1999,

> We recognized the value of giving customers control of their software, and sought to bring brand reliability to the Linux product. We would offer support to customers and accelerate development of the operating system by investing our own R&D [research and development] dollars in new Linux technology that would then be given back for free to the community, for any Linux programmer or distributor to use. We had no intention of ever 'owning' the intellectual property we created. Instead, our business model was based on quickly expanding the market, and earning a small amount of revenue from a large number of customers who would buy a product that was better quality that that being offered by the industry leader, Microsoft (Young & Rohm, 1999, 10).

The “better quality” that Young is referring to is the Linux-based operating system. Previous chapters have already discussed the purported benefits of collaborative development, most notably its efficiency, innovativeness, and security, but Red Hat offered an operating system that could be easily adapted to the needs of the customer. This was particularly important in a time when hardware vendors were reliant on large, proprietary firms like Microsoft to develop operating systems that could make use of their hardware. The speed at which new versions of proprietary operating systems could be developed was much slower compared to the open-source options. Consequently, Red Hat received and continues to rely on strategic partnerships with hardware manufacturing companies like Intel, IBM, Dell, CISCO, Hewlett-Packard, Sony, and others.

These partnerships are beneficial to both Red Hat and their partners for a couple reasons. First, Red Hat is able to pursue its original goal of lending credence to free and open source software (FLOSS) solutions by receiving backing from major information

technology firms. Second, Red Hat continues to position itself as a leading company dealing solely in FLOSS. Third, Red Hat can continue to funnel funding back into FLOSS projects that ultimately benefit the developer communities who work on these projects. In this sense, Red Hat does serve as a intermediary between large information technology firms and the FLOSS community.

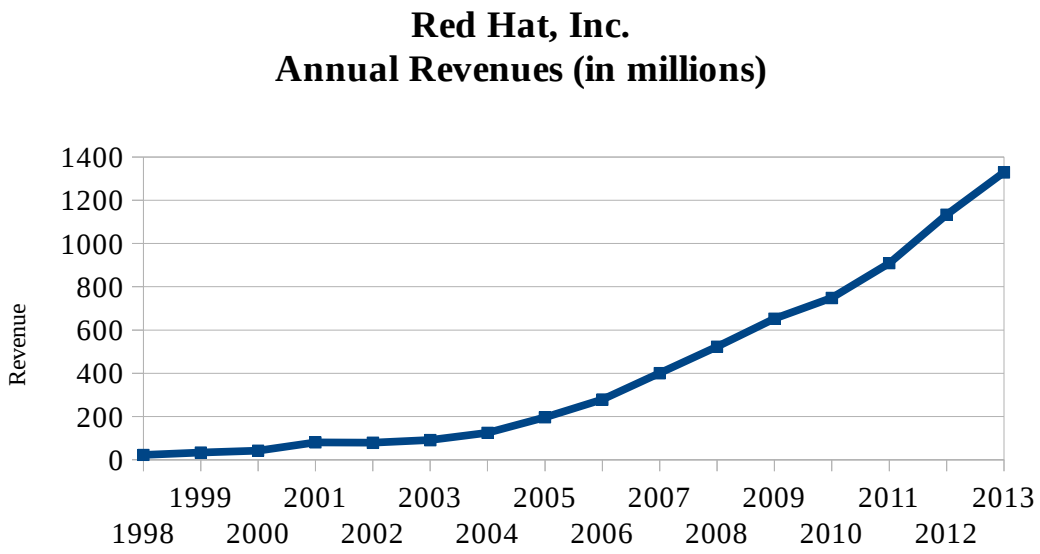
However, Red Hat also benefited from venture capital investment, particularly at a time when the “dot-com” investment bubble was on the rise. Frank Batten Jr., through Landmark Communication, was an early investor in Red Hat and committed $2 million to the company in 1997 (Young & Rohm, 1999). Landmark Communication was famous for investing in the Weather Channel, and the company remains a privately held investment firm but now operates under the name Landmark Media Enterprises. Red Hat also received investment capital from Greylock Limited Partnership and Benchmark Capital, a company based in Menlo Park, CA, and known for its investment in and support of the open-source community. All three of these parties – Landmark Communication, Greylock, and Benchmark Capital – became major shareholders in Red Hat after its initial public offering (IPO) in 1999.

Red Hat held its IPO in August of 1999. The previous rounds of investment coming from venture capital firms, plus the company's partnerships with major information technology companies, led to rapid growth of the firm's value. In September of 1999, Red Hat's stock price rose to more than $122 per share, which was up from its original price of $14 per share. At the time, Frank Batten Jr. owned 15 million shares of the company, while Greylock Limited Partnership owned 8.7 million shares, and Benchmark Capital owned 5.8 million shares (Kanellos & Shankland, 1999). However, in the interest of giving back to the FLOSS community that Red Hat relied upon for their

business model to work, the company tried to compile a list of all the FLOSS developers who contributed to Linux and other FLOSS projects. While arriving at a fully comprehensive list was not possible, the company managed to develop a list of approximately 5,000 developers. The intention was to make these developers stock holders in the company so they could benefit from the company's growth. Doing so was seen as a way for the company to give back to the FLOSS community. While Securities and Exchange Commission regulations prevented a large portion of these developers from becoming investors, more than 1,000 of the eligible 1,300 developers became early investors in the company (Young & Rohm, 1999). Making this effort at including members of the FLOSS community as shareholders in the company is one example of how Red Hat has tried to maintain a good relationship with the FLOSS community.

In the years following the IPO, Red Hat continued to enjoy growth in revenues. Figure 4.1 provides an illustration of Red Hat's rising revenues from 1998 until 2013. What is particularly striking about this illustration is the fact that Red Hat's revenues were not adversely effected by the dot-com bubble crash between 1999-2001. Red Hat emerged from this period and continued to grow. This is most likely because of the strategic partnerships Red Hat was able to negotiate with large information technology firms in the lead up to the dot-com crash. Those firms – Intel, Cisco, IBM, Dell, etc. – also survived the dot-com crash and have solidified their positions within the market for information and communication technologies. Even though Red Hat was a start-up company at the time, the partnerships that the company formed with these larger firms ensured that Red Hat would be supported by these businesses into the future.

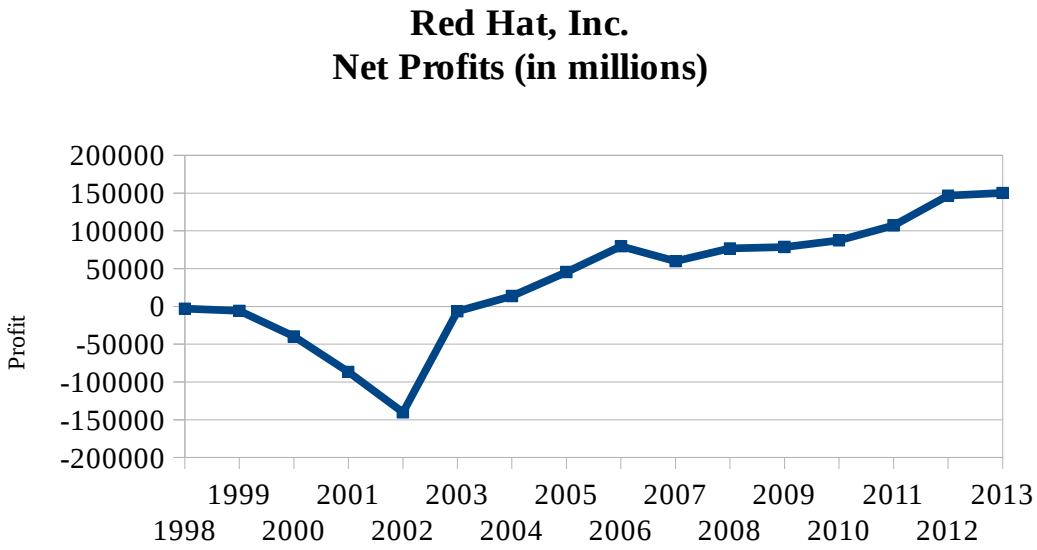
Figure 4.1. Red Hat Annual Revenues 1998-2013



Source: Data is from Red Hat's 10-K Annual Reports filed with the SEC for the years 1998-2013.

Looking at the trajectory of Red Hat's net profits, on the other hand, exhibited a noticeable dip during the dot-com bubble crash (see Figure 4.2). Red Hat's profits dipped from 1998 until 2002, but rose again in 2003. This performance almost perfectly coincides with changes in management, and can also be explained by a shift in Red Hat's business strategy during those periods. In 1999, the original co-founders of the company, Bob Young and Mark Ewing, left the company. In 2001, Paul Cormier joined the company and began to lobby for shifting the company's business model. Specifically, Cormier wanted to provide FLOSS solutions at the enterprise level rather than in the consumer market. To more fully explain the nuances of this shift, the following section contains an in-depth discussion of Red Hat's core commodities, how those commodities shifted focus over time, and how Red Hat was able to centralize intellectual property within its corporate structure.

Figure 4.2. Red Hat Annual Net Profits 1998-2013



Source: Data is from Red Hat's 10-K Annual Reports filed with the SEC for the years 1998-2013.

Red Hat's Core Commodities and Intellectual Property

Red Hat's business model relies primarily on its ability to provide an easy-to-use and accessible version of Linux by producing packaged distributions of the operating system, while also providing additional services and customer support that cater to its products. Red Hat's revenues then come from these two streams.[12] The majority of Red Hats' revenues comes from a subscription model, whereby clients get both products and support from Red Hat in exchange for a fee. The types of products and services provided depend on the level of subscription. The effectiveness of this subscription model is based, to a large degree, on two interrelated factors: Red Hat's recognition as a trustworthy provider of FLOSS products and services, as well as Red Hat's position as a legally recognized institution, which can be held liable for the products and services it provides.

12 Unless otherwise noted, the information from this section comes from Annual Reports (Form 10-K) filed with the Securities and Exchange Commission in the United States between the years 2000-2013.

Most importantly for its customers, Red Hat is able to provide a way to outsource services that may otherwise be too expensive to perform within one's own company. Indeed, any one of Red Hat's customers could perform the work done by Red Hat, especially because the underlying code that Red Hat relies on is available for free. Red Hat does not own the intellectual property rights for the free software that its services are based upon, and the company is not necessarily trying to exclude others from this intellectual property. Indeed, Red Hat has built its business model based solely on free software that is protected by the GNU General Public License (GPL). As such, any of its customers could, in theory, produce the same software that is sold by Red Hat, but they would need to perform the work themselves. However, Red Hat is a legally recognized institution that can be liable for the products and services it supplies. Because of this, its customers can feel comfortable with the sense of security they get when they sign a contract with Red Hat, and this, in turn, is how Red Hat has been able to become the market leader providing FLOSS distributions and services to earn its revenues. Prior to its founding, FLOSS projects had differing degrees of trustworthiness. By forming a corporate entity that could be held liable for the products and services provided, Red Hat served as a way to legitimize a system of production that was massively distributed and seemingly anarchic. To its customers, then, Red Hat serves as a legally recognized entity providing an assurance for the free software products and services it supplies.

But to understand the types of products and services that underlie Red Hat's market position, we will need to examine exactly how Red Hat has been able to profit from free software, especially as it does not own any of the intellectual property that makes up its business model. To do so, this section begins with a discussion of Red Hat Linux, which was the original operating system sold to customers from 1994-2004.

Then, the shift that occurred when Red Hat focused more on providing enterprise solutions through their Red Hat Enterprise Linux offering is discussed. The relationship between Red Hat's core commodities and Fedora, which is the primary FLOSS project supported by Red Hat, is also described.

Red Hat Linux

When Red Hat first began offering products and services in the early 1990s, it began selling a compact disc for approximately $50, which contained a Linux distribution called Red Hat Linux, some additional applications, and documentation. Red Hat Linux was based purely on computer code that was protected by the GPL – that is, code that needs to remain freely available for distribution, modification, adaptation, etc. Red Hat Linux provided the principle source of revenue for Red Hat during its early years. Revenues came primarily from sales of Red Hat Linux to distributors and original equipment manufacturers (OEMs) for inclusion on their hardware. These companies are some of the same who invested directly in Red Hat during its early years: Dell, Cisco, Hewlett-Packard, IBM, and Intel. Because Red Hat had the power of a potentially very large and distributed labor force behind it, like the FLOSS community, its business model was highly scalable. That is, Red Hat had the ability to quickly expand its market without incurring increased investment costs. In other words, the Red Hat Linux product could be distributed to a large number of customers without need for more investment. This was precisely Red Hat's strategy: to rapidly increase the market, deriving a small amount of revenue from a large number of transactions, while reinvesting its earnings back into the FLOSS community.

While Red Hat Linux provided the primary commodity for Red Hat during its early years, the bulk of its work was coming from the support it provided for this software. Red Hat's employees provided customer support, education, training, and technical support to its clients who were using Red Hat Linux. This strategy, along with Red Hat's strategic partnerships, allowed the company to begin picking up market share during its early years. While the company's revenues were still growing up until 2004, it still had not become a profitable business. This was, in part, due to its acquisitions of other software firms before the dot-com bubble crash, but also because the company had not yet found a way to substantially increase subscription sales at the enterprise level to other businesses. This is precisely the change that occurred when the company shifted its focus to Red Hat Enterprise Linux, which became its core commodity and continues to be today. The final stable version of Red Hat Linux was released in 2003, which was the same year that Red Hat Enterprise Linux was released.

Red Hat Enterprise Linux and the Fedora Project

In 2003, Red Hat split its Red Hat Linux project into two separate projects: Red Hat Enterprise Linux and the Fedora Project. Red Hat Enterprise Linux continued as a core commodity for Red Hat in the same way that Red Hat Linux had been before. The Fedora project, however, became a community-based FLOSS project. Red Hat Enterprise Linux relied on the same model as Red Hat Linux in terms of providing packaged distributions of a free operating system but, rather than selling individual compact discs containing the software, Red Hat Enterprise Linux was made available through a subscription model. Depending on the level of subscription, customers could get access to customized versions of the Red Hat Enterprise Linux operating system plus

different levels of support services for the operating system. In effect, Red Hat Enterprise Linux was a sufficiently similar product to Red Hat Linux with a different model for how the product would be provided to customers. Red Hat then used the revenues from sales of Red Hat Enterprise Linux to support the Fedora Project. The relationship between these two projects provides perhaps the most interesting insight into how Red Hat incorporates the commons.

The split into Red Hat Enterprise Linux and the Fedora Project in 2003 was made with the intention of finding a mutually beneficial way for the community and Red Hat to collaborate on developing software. Red Hat Enterprise Linux continues to serve as the primary core commodity of Red Hat, and the company profits from subscription sales to its customers. The Fedora Project was meant to be a community-sponsored project that would provide an incubator for innovation. In return, the innovation that occurred within the Fedora Project could then be implemented into Red Hat's commercial offerings. This was possible was because of the ownership and governance structure of the Fedora Project, as well as the worker contracts established with contributors to the project.

Ownership, Governance, and Intellectual Property in Fedora

The Fedora Project is ultimately owned by Red Hat. However, the Fedora Project is led by the Fedora Project Board, which has nine members.[13] Of those nine members, five are elected by the community of developers who contribute to the Fedora Project. The remaining four members are appointed directly by the Fedora Project Leader, who is a full-time employee of Red Hat. The Fedora Project Leader also serves as Chair of the

13 Information about the Fedora Project Board is publicly available on the project's wiki, which is available at: http://fedoraproject.org/wiki/Board (last accessed July 7, 2014)

Fedora Project Board and holds veto power over any decision made by the board. Any person involved in the Fedora Project, whether a Red Hat employee or community member, may be elected to serve on the Fedora Project Board. This suggests, however, that Red Hat ultimately holds power over the decisions made by the community. While Red Hat does have an incentive not to abuse this veto power, the power still exists. CHAPTER VI will discuss in greater depth how the community reacts to an abuse of governance power by corporations that sponsor FLOSS projects.

The ownership and governance structure of the Fedora Project ultimately gives Red Hat the power to direct the types of development that occur within the community. Red Hat supports the community by sponsoring the project, but it then uses the work performed by the community in its commercial offering, Red Hat Enterprise Linux. The second reason Red Hat is able to appropriate the labor performed within the community is because all contributors to the Fedora Project have had to sign a contributor's agreement. These agreements have changed throughout the history of the Fedora Project, but both have similar effects. Originally, contributors needed to sign the Individual Contributor Licensing Agreement (ICLA), which effectively assigned the contributors' copyright to the Fedora Project.[14] However, the ICLA was later abandoned for the Fedora Project Contributor Agreement (FPCA), which no longer assigned copyright to Red Hat but specified the types of licenses that could be included in the Fedora Project.[15] This shift made it possible for code that had already been licensed under a previous licensing

14 Information about the Individual Contributor Licensing Agreement can be found on the project's wiki at http://fedoraproject.org/wiki/Legal:Licenses/CLA (last accessed July 7, 2014)

15 Information about the Fedora Project Contributor Agreement can be found on the project's wiki at http://fedoraproject.org/wiki/Legal:Fedora_Project_Contributor_Agreement (last accessed July 7, 2014).

scheme to be included in the Fedora Project, as long as the licenses were compatible with the guidelines established by Fedora.

Both the ICLA and the FPCA enable Red Hat to centralize control and ownership of commons-based peer production into into its corporate structure. In the case of the ICLA, it provided a direct assignment of a contributor's copyright to Red Hat, whereas the FPCA does not necessarily assign copyright to Red Hat. In this sense, the FPCA can be viewed as less restrictive because it allows contributors to assign licenses to their work prior to submitting the work to the Fedora Project. However, those licenses must be compatible with the goals of the Fedora Project, and the Fedora Project wiki maintains a Software License List that identifies the acceptable and unacceptable licenses that can be included in Fedora.[16] Code protected by these licenses can still be legally defended by Red Hat. In the event that content other than code is included in the submission (text, images, logos, etc.), the contributor must waive his or her moral rights to the content. This ensures that Red Hat will not be subject to infringement claims. In effect, these licensing agreements provide a way for Red Hat to control what is included in the commons-based project (Fedora) so that when that material is included in their commercial offering (Red Hat Enterprise Linux or other software), the company will not be subject to intellectual property infringement claims.

By taking these preventative measures to control what is included in Fedora, Red Hat can provide its customers with a guarantee that they will not need to fear a potential claim against intellectual property infringement. Red Hat does this through its Open Source Assurance Program. The details of the program are codified in the Open Source

16 The Software License List can be found at http://fedoraproject.org/wiki/Licensing:Main?rd=Licensing#Software_License_List (last accessed July 7, 2014).

Assurance Agreement[17] contract, which states that in the event that a third party alleges infringement of intellectual property in the software provided to the client by Red Hat, the company will,

> (i) defend Client against the Claim and (ii) pay costs, damages and/or attorneys fees that are included in a final judgment against Client (without right of appeal) or in a settlement approved by Red Hat that are attributable to Client's use of the Covered Software; (Red Hat, Inc., 2014)

Furthermore, if the Client's use of Red Hat's software is found to infringe the third party's intellectual property rights, then Red Hat will

> (i) obtain the rights necessary for Client to continue to use the Covered Software consistent with the Support Agreement(s); (ii) modify the Covered Software so that it is non-infringing; or (iii) replace the infringing portion of the Covered Software with non-infringing code of similar functionality (subsections (i), (ii) and (iii) are the "IP Resolutions"); provided that if none of the IP Resolutions is available on a basis that Red Hat finds commercially reasonable, then Red Hat may terminate the Support Agreement(s) without further liability under this paragraph, and, if Client then returns the Covered Software that is subject to the Claim, Red Hat will refund any prepaid subscription fees related to Covered Software. (Red Hat, Inc., 2014).

From Red Hat's perspective, then, this is the legal-juridical benefit of controlling what is included in the Fedora Project as well as centralizing control of the intellectual property rights within its corporate structure. Red Hat relies on the FLOSS community to perform the cooperative labor of developing new features, fixing bugs, or otherwise improving the Fedora Project so that these features can be included in its commercial offerings. In order to assure its customers that they will not be subject to intellectual property infringement claims from third parties, Red Hat requires contributors to assign licenses to their work

17 The full text of the Open Source Assurance Agreement can be found at http://www.redhat.com/legal/open_source_assurance_agreement.html (last accessed July 7, 2014).

that will allow Red Hat to continue providing its services. In effect, Red Hat is separating authorship from ownership, which is one of the primary critiques of intellectual property laws (see Bettig, 1992). However, Red Hat does not use copyright to prevent authors or anyone else from making use of the code in other ways. Basically, Red Hat is just trying to ensure that the code in Fedora can be legally defensible, while allowing the company to provide assurances to its customers. Red Hat's method for protecting its core intellectual property does not come from copyright, but the company still prevents exact redistributions of its property through trademark law.

Red Hat, Trademark, and CentOS

As stated earlier in this chapter, Red Hat does not own the intellectual property that makes up its core commodities. Most of the code that makes up Red Hat's core commodities is covered by the GPL, which allows others to freely copy, modify, and redistribute the code. Therefore, rather than relying on copyright to protect its core commodities, Red Hat relies on trademark law to protect its properties. The details of this strategy can be found in Red Hat Trademark Guidelines[18] document (Red Hat, Inc., 2006). In theory, anyone could make an exact copy of Red Hat's open source software and begin selling it, but they would be prevented from including any registered trademarks. These trademarks include the logos and names of software, which means that exact copies of Red Hat's open source software would need to be given a different name. Red Hat's trademarks also prevent products from having names that are sufficiently similar, like “Green Hat” or “Red Cap,” or “Redd Hatte.” While these

18 The Red Hat Trademark Guidelines are available at http://www.redhat.com/f/pdf/corp/RH-3573_284204_TM_Gd.pdf (last accessed July 7, 2014).

restrictions exist, CentOS provides an example of a project that served as an exact replacement for Red Hat Enterprise Linux.

CentOS began in 2004, and served as a functionally compatible version of Red Hat Enterprise Linux. Indeed, CentOS was based on the publicly available code for Red Hat Enterprise Linux. Rather than competing with CentOS or trying to prevent them from using code included in Red Hat Enterprise Linux, Red Hat was largely ambivalent about CentOS. This is, in part, due to the perception that customers who want to use CentOS will probably continue to use it, but also because those customers could switch to Red Hat Enterprise Linux at any time because the two operating systems were basically the same. However, whatever tension may have existed between the two operating systems became a moot point in 2014, when Red Hat officially became a sponsor of the CentOS project. The move was perceived as a way to meet users' demands across the three major versions of Red Hat's software – Red Hat Enterprise Linux, Fedora, and CentOS – by giving users access to features that may not be included across all versions of the operating system (Vaughn-Nichols, 2014). As part of Red Hat's new sponsorship of the CentOS project, all CentOS trademarks were transferred to Red Hat.

Red Hat's use of trademark law to protect its market position is used in conjunction with its ability to control the intellectual property included in its commercial offerings. By sponsoring the CentOS project, Red Hat is able to increase its intellectual property holdings, while also eliminating a rival form of free software that was offering a functional equivalent of its commercial software. In this sense, Red Hat's sponsorship of the CentOS project functions similarly to a corporate acquisition or an instance of horizontal integration.

Core Commodity Conclusions

What becomes clear from this discussion of Red Hat's core commodities and the services provided for those commodities is that Red Hat, as an institution, may be viewed in at least two different ways. On the one hand, Red Hat can be viewed as a pragmatic way to centralize commons-based peer production in a capitalist system. In effect, Red Hat serves as a gateway for access to commons-based peer production by being dialectically situated between capital and the commons. In other words, Red Hat leverages the power of the commons by finding a way to centralize production into a few core commodities that can then be sold to other businesses as information technology solutions. This offers another part of the explanation for why Red Hat maintains a good reputation with the FLOSS community. Red Hat is clear about its intentions for being involved in FLOSS projects. Indeed, Red Hat's entire business model is founded on finding a way to sell the power of FLOSS production to other businesses. In return, Red Hat reinvests in the FLOSS community by supporting FLOSS projects, acquiring new businesses and releasing source code to the community. The relationship between Red Hat and the FLOSS community is one of mutual benefit: Red Hat's financial success benefits the FLOSS community, more revenue for Red Hat means more investment in FLOSS projects, and more investment in FLOSS projects means higher quality products and services that Red Hat can offer to its customers.

However, Red Hat can also be viewed as an institution that operates no differently than other corporations operating under a capitalist system. Red Hat relies on centralizing production within its corporate structure, separating authorship from ownership through workers agreements, and protecting intellectual property through

trademark laws all for the purpose of making profit. The difference in Red Hat's case is that it cannot prevent others from using its intellectual property through copyright laws. Moreover, Red Hat does not directly employ its entire labor force. As such, the company does not compensate all of its laborers through wages, but must rely on other informal ways of compensating laborers. Because the company relies on this labor force, it must maintain a good relationship with that community, or the community may move production elsewhere. CHAPTER VI will describe some of the specific ways that the community maintains this ability to leverage its labor power. The purpose of this chapter is to demonstrate the ways in which one company centralizes commons-based peer production into a commercial product.

From the Commons to Capital

Red Hat offers an example of how a distributed system of commons-based peer production can become centralized within a corporation and turned into a profitable business. In part, Red Hat's success can be explained by its strategic partnerships with large information technology companies. These partnerships can at least explain how Red Hat was able to survive the period immediately following its initial public offering. Interestingly, it became a publicly traded company at the same time that many dot-com companies were the targets of massive capital investment, a period referred to as the “dot-com bubble.” Red Hat was also one of the earliest companies to position itself as the leading company providing services for FLOSS. As such, Red Hat sought to lend an element of professionalism to the emerging FLOSS phenomenon by establishing the formally recognized institution of a publicly traded corporation that could be legally liable for the services provided. Consequently, Red Hat needed a way to hold the rights

to the commons-based peer production that made up its core commodities. The company accomplished this through Individual Contributor License Agreement (ICLA) and later the Fedora Contributor License Agreement (FCLA) that granted the company rights to use the production that was performed by developers.

The CLA is a striking example of how authorship is separated from ownership. This separation is essential to Red Hat's business model because it grants exclusive rights to Red Hat so that the company becomes both legally liable for the products it is selling as well as legally able to defend those products in the event of a violation. However, it is important to note that Red Hat is not alone in using these types of worker agreements. The issuing of contributor licensing agreements (CLAs) is common practice in FLOSS projects. These CLAs represent the most striking example of how institutions, whether for-profit or non-profit corporations, or any other type of legally recognizable organization, centralize commons-based peer production by separating authorship from ownership.

However, the peculiar thing about Red Hat is that it does not use contributor agreements to protect copyrights. Most, if not all, of the code underlying its core commodities is protected by the GPL, which grants the right to copy, modify, or redistribute the work. In addition, the GPL requires that modified versions of the intellectual property be protected by the same license. By using code that is protected by this license as well as similar licenses, Red Hat cannot rely on copyright law to prevent others from making exact copies of the code it makes publicly available. Rather, Red Hat relies on trademark laws that protect the names and logos for their products.

For all the rhetoric of revolutionary productive processes, massively decentralized and distributed systems, FLOSS as a process and product still exists within a capitalist

system that requires commercial entities to be held liable for the products and services they provide. Therefore, productive activity under capitalism still takes the form of centralization, control, and appropriation of surplus value. Even in the case of so-called "non-market production," the labor performed under these conditions can still be appropriated for corporate gain. Even if the producers are not centralized within a particular institution, corporations require any claims to the knowledge commons be surrendered so that the commons may be exploited for corporate gain. In the case of Red Hat, the company provides a legally recognized and formalized institution that makes use of trademark laws to effectively brand commons-based peer production. This may be viewed as a mutually beneficial relationship that provides a pragmatic solution to the problem of how to organize commons-based peer production in a way that allows members of the community to earn a living. However, this condition only benefits a portion of the community.

In the event that contributors to the knowledge commons are not employed by one of the institutions supporting a FLOSS project, their payment comes to them informally when they attend public events or trade shows where institutions like Red Hat provide sponsorship or other goods and services for the community. This informal economy is only sustainable for as long as the institutions supporting FLOSS projects remain transparent about their intentions for the products of FLOSS developers' labor and continue to support the community through the provision of paid employment, sponsorship of additional FLOSS projects and events, and informally through gifts given to the community.

This chapter has demonstrated how profitable Red Hat has become. This is, undoubtedly, due in part to its relatively low labor costs in comparison to the size of its

workforce. Red Hat offers an example of a way that commons-based peer production can be centralized within a corporate structure. Red Hat has grown because of the relationships it has negotiated, the strategies it uses to control its intellectual properties, and its willingness to give back to the FLOSS community in a variety of ways. The following chapter charts the very different history of the Microsoft Corporation's long and checkered relationship with the FLOSS community.

CHAPTER V

SHIFTING TOWARD THE COMMONS: MICROSOFT'S LONG AND WINDING HISTORY WITH FREE SOFTWARE

The Microsoft Corporation (“Microsoft” hereafter) offers perhaps the most contentious relationship with the open source community. Primarily, this is due to Microsoft's core business model, which relies on the sale of proprietary software. Through strategic partnerships, strong intellectual property protections, and a robust strategy for capturing the consumer market for personal computer (PC) sales, Microsoft grew to become one of the largest software companies in the world. At its peak, Microsoft enjoyed a nearly 97% market share of all computing devices in the year 2000 (Tu, 2012). This was before the company was found to be in violation of the Sherman Antitrust Act by the U.S. Department of Justice (DOJ). However, the antitrust decision did little to curb Microsoft's economic growth at the turn of the 21st century. Rather, the company's profits continued to grow, and Microsoft still ranks as one of the largest and most dominant software companies in the world. What has changed, particularly after the antitrust ruling, is the company's relationship to the broader free and open source software community.

As mentioned in the introduction to this dissertation, Microsoft's former Chief Executive Officer, Steve Ballmer, referred to Linux – the open source operating system – as “a cancer” in 2001. Slightly more than eleven years later, the company opened an entire division of its company devoted to the promotion and development of open source software. In this chapter, the history of Microsoft's checkered relationship with free and open source software (FLOSS) is charted, focusing on three specific moments that illustrate this relationship. First, the company's initial growth and its rise as one of the

most dominant software companies in the world is described. During this time, the company took an adversarial approach to open source software. This includes Bill Gates' “Open Letter to Hobbyists” in which he decried the widespread culture of freely sharing software in the hobbyist community, as well as the leak of internal documents known as “The Halloween Documents” in 1998, which clearly outline the company's views on open source software. The second section discusses the U.S. Department of Justice's investigation and, ultimately, its conviction of Microsoft for violating the Sherman Antitrust Act. Findings from the investigation and the subsequent decrees issued to the company in the wake of the conviction are detailed. The final section focuses on the most recent history of Microsoft, including its Shared Source program as well as its decision to create Microsoft Open Technologies, a wholly owned subsidiary dedicated solely to promoting and developing open source software, open standards, and open technologies.

The Microsoft case study exemplifies the clash between two opposing systems of production. On the one hand, Microsoft relies upon strong intellectual property protections to exclude others from making use of its products. Those products have been produced in-house as part of Microsoft's core business model. Microsoft uses these intellectual property rights not only to protect its own works, but to threaten FLOSS projects with infringement lawsuits. It is within this context that we can view Microsoft's long history of railing against the lack of intellectual property within the FLOSS community, beginning with Bill Gates' “Open Letter to Hobbyists” in 1976, through Steve Ballmer's “Linux is a cancer” claim. What changed after the DOJ antitrust ruling is that Microsoft shifted its position toward FLOSS projects in general by submitting its own licenses for approval by the Open Source Initiative (OSI). The shift in Microsoft's

stance toward FLOSS after the antitrust ruling represents an important moment for Microsoft, specifically, but also for the software industry in general. The shift can be read as a humble admission that the business model upon which Microsoft relied for most of its history had been mostly usurped by a more efficient and effective model of production. But it can also be read within the broader context of the dot-com bubble burst that hit the economy at the end of the 20th century, which coincided with many Internet-related companies failures but also the emergence of the Web 2.0 phenomenon. It was during this time after the DOJ ruling that Microsoft not only readjusted its positioning with respect to FLOSS projects, but also attempted to become more directly involved in FLOSS projects. The company's reasons for doing so were primarily to comport with the consent decrees that the company agreed to as part of the antitrust ruling, but also because the commons-based peer production of FLOSS had proven to be a viable and successful business model. In short, Microsoft was basically forced to adopt a more open stance to the broader FLOSS community – first because of the consent decrees and, second, because of broader historical forces affecting the software industry.

Ultimately, the goal of this chapter is twofold: first, to argue that the antitrust conviction in 2001 marks a critical moment in Microsoft's history that, when paired with the bursting of the dot-com bubble and the emergence of the so-called Web 2.0 phenomenon, caused a shift in Microsoft's business strategy whereby the company tried to find ways of harnessing the power of commons-based peer production. Second, it demonstrates how Microsoft's own history is contradictory to its stance against the open sharing of ideas. In fact, many of Microsoft's most successful products have incorporated or licensed design features that were developed by others. By making these two points, the chapter shows how Microsoft's relationship with the FLOSS community can be

understood as a strategic readjustment that was undertaken in response to Microsoft's declining market share at the same time that Linux-based systems were gaining market share. Although not a complete transformation of its initial stance, Microsoft's shift in its relationship to the broader FLOSS community can be described as moving *from capital toward the commons.*

The Rise of Microsoft 1975-1990

Microsoft was founded in 1975 after Paul Allen and Bill Gates developed the Altair BASIC interpreter. An interpreter is a computer program that directly performs functions written in a programming language. In the case of Altair BASIC, the interpreter was designed to execute functions written in the BASIC (Beginner's All-purpose Symbolic Instruction Code) programming language so that they could be performed on the Micro Instrumentation and Telemetry Systems (MITS) Altair 8800 microcomputer. Altair BASIC became Microsoft's first product, which was distributed by MITS under contract with the newly created company. From its very beginnings, then, Microsoft focused on providing software solutions that could be included on hardware devices. Microsoft's business model relied on establishing contracts with hardware providers, which would allow Microsoft products to be included on hardware. However, the company has consistently exhibited an antagonistic position with respect to alleged infringements on its intellectual property. The first example of such behavior came from unauthorized copying of its original Altair BASIC interpreter.

The Altair 8800 microcomputer has been credited as the device that ushered in the microcomputer revolution (Garland, 1977). The Altair 8800 became widely popular after being featured on the cover of the January 1975 edition of *Popular Electronics*. From the

magazine, readers could order kits for the computer, which could then be assembled by hobbyists interested in experimenting with the device. As part of the order, readers could purchase the Altair BASIC language for a fixed price. Since the Altair BASIC language could be included with orders for the Altair 8800, Altair BASIC also became widely used. However, hobbyists often made copies for friends or others as a way to allow them to experiment with the device as well. This made Altair BASIC subject to unauthorized copying, which prompted Bill Gates to publish an "Open Letter to Hobbyists" on February 3, 1976.[19]

In the letter, Gates noted that "hundreds of people who are...using BASIC" have all provided positive feedback about the interpreter. However, he claims that "most of these 'users' never bought BASIC," as "less than 10% of all Altair owners have bought BASIC," and the "amount of royalties [Gates and Allen] have received from sales to hobbyists makes the time spent of [sic] Altair BASIC worth less than $2 per hour" (Gates, 1976, 2). Gates continued by decrying the fact that most hobbyists steal software, and asked whether this is a fair practice because it ultimately prevents good software from being written. In effect, Gates was arguing that the time, labor, and resources spent on developing software ought to be returned to him in the form of fair payment for use of the software.

Gates' open letter signaled what would become a recurring theme throughout Microsoft's history. Specifically, Gates and Microsoft accused members of the hobbyist community of infringing on their intellectual property rights. The hobbyist community, then, represented a threat to Microsoft's business model, which was one founded on the

19 The "Open Letter to Hobbyists" is available via the Wikimedia Commons here: https://upload.wikimedia.org/wikipedia/commons/1/14/Bill_Gates_Letter_to_Hobbyists.jpg (Last accessed July 3, 2014)

need to protect its products by using strong intellectual property protections. Indeed, some of the responses to Gates' open letter focused more on the business strategy, especially the shortcomings of Microsoft's contractual negotiations with the hardware vendor (Hayes, 1976). In the years that followed the Altair BASIC beginnings, Microsoft pursued a course of action that sought to do exactly that. By ingratiating itself with large hardware manufacturers, Microsoft rapidly gained market share and became one of the most dominant software companies in the world.

MS-DOS

Microsoft's business strategy during its early years focused primarily on providing BASIC interpreters, but the company shifted its focus to operating systems in the early 1980s. From the 1980s until the mid 1990s, Microsoft primarily relied on its Microsoft Disk Operating System, or MS-DOS, as its core commodity. MS-DOS originated in 1981 after IBM put out a request for an operating system to use on its IBM-PC line of personal computers (PC). Shortly after the initial request from IBM, Microsoft acquired the rights to the 86-DOS, an operating system from Seattle Computer Products, which it renamed MS-DOS.[20] Microsoft customized the newly acquired operating system to the specifications required by IBM and licensed the operating system to IBM, which included it in its IBM-PCs under the name PC DOS.

Microsoft's contract with IBM was not without controversy, however. The rise of the PC was made possible by advances in integrated circuit, or microchip, technology. Microchips for the consumer market were first used commercially in calculators, which

20 The original name for 86-DOS was actual QDOS, which stood for “Quick and Dirty Operating System,” but Seattle Computer Products changed the name to 86-DOS once it began marketing the product.

were manufactured by companies like Hewlett-Packard and Texas Instruments. As demand for higher performance calculators increased, Intel was commissioned by Busicom, a Japanese firm, to produce the first commercially available microprocessor that could receive digital data and process it according to its programmed functions. The new microprocessor was called the Intel 4004 (Nairn, 2002). However, these new chips still needed language capable of converting instructions into signals that the chip could process. This operating system came from Gary Kildall, who had authored a language capable of performing such functions. Eventually, Kildall's language was transformed into the first operating system for personal computers, known as CP/M. The rights to CP/M were held by Kildall's company, Digital Research, Inc., or DRI.

Throughout the late 1970s, CP/M became the industry leader in operating systems for personal computers. When IBM announced its initial line of personal computers, the company chose Intel as the provider for microprocessors, but it also needed a supplier for the operating system. Both Microsoft and DRI were consulted about providing an operating system. The exact details about what transpired during the negotiations are a bit murky,[21] but we know that Microsoft eventually won the contract, which resulted in the acquisition of 86-DOS that was subsequently rebranded as MS-DOS. Kildall, however, would claim that MS-DOS infringed on his copyright for CP/M. Kildall confronted both Gates at Microsoft and IBM about the alleged infringement but, on advice from lawyers, decided not to sue. Instead, Kildall chose to license CP/M to IBM for inclusion on their personal computers. When the IBM PCs were eventually released,

21 There are many different accounts of what happened. One of the most popular stories claims that Kildall snubbed the executives from IBM by choosing to go flying in his personal airplane at the time the meeting was scheduled. Other accounts claim that Kildall's wife killed the deal by insisting on changes to the contract, and others claim that Kildall did not want to release the source code for CP/M to IBM. These stories are recounted on the DRI Web site, which can be found at http://www.digitalresearch.biz/HISZMSD.HTM (last accessed May 14, 2014)

IBM offered a choice of operating system: $240 for CP/M or $40 for DOS (Hamm & Greene, 2004). In effect, Microsoft became the clear choice for consumers, and DRI was eventually purchased by Novell in 1991.

Microsoft's contract with IBM was perhaps the biggest turning point on its path to becoming the largest software company in the world. As part of Microsoft's contract, it reserved the right to sell its operating system to third-party vendors as well, which allowed the company to exploit sales of its operating system to any hardware manufacturer. Employing this strategy, Microsoft grew tremendously from 1981-1995, with an increase in annual revenues from $16 million in 1981 to more than $6 billion in 1995 (Campbell-Kelly, 2001). Although exact figures are not publicly available, some estimates suggest that MS-DOS held nearly a 90% market share of the PC market (Gilbert, 1995). Although MS-DOS would continue to be produced until September 14, 2000, Microsoft began to focus its efforts on developing an operating system that would use a graphical user interface (GUI). The product that it ultimately developed, Microsoft Windows, would continue Microsoft's dominance of the personal computer software industry.

Microsoft Windows

Operating systems featuring a GUI did not start with Microsoft. Researchers working at Xerox's Palo Alto Research Center (PARC) first developed the GUI, which was used on the Xerox Alto computer in 1973. However, Xerox did not successfully exploit the GUI commercially. Since the market for personal computers and operating systems was already dominated by IBM and Microsoft, Xerox found it difficult to focus its efforts on commercially exploiting the GUI. Consequently, Xerox invited Steve Jobs

and other representatives from Apple to its PARC for access to its prototypes in exchange for a $1 million investment in Apple prior to its initial public offering (Ward, 2013). During this visit, Jobs viewed prototypes of a computer mouse used for navigation as well as the ability to move text around on the screen. From this meeting, Jobs is said to have refocused efforts at Apple toward developing a GUI operating system. However, others have argued that assigning too much causality to Jobs' single visit is an erroneous assumption, as other Apple engineers had ties to the PARC and Jobs himself made more than one visit (Pang & Marinaccio, 2000). Whatever the inspiration, Apple worked on developing a GUI operating system for its Macintosh personal computers. However, Apple still lagged behind IBM and Microsoft in developing applications for its operating system.

Since Microsoft had established itself as a leader in the market for operating systems for PCs, and had previously worked with Apple by producing the SoftCard, a microprocessor designed to run programs designed for CP/M on the Apple II computer, Microsoft negotiated a licensing agreement for access to the Mac operating system in 1985. At this point, Microsoft had already been working on Microsoft Windows, its GUI operating system, which was announced in 1983. The purpose of the license was to allow Microsoft access to certain visual elements of the Mac operating system so that Microsoft could develop applications for the Macintosh (The History of Computing Project, 2014). Indeed, Microsoft used its powerful position in the PC software market by threatening to "cease development work on important Mac applications unless such a license was granted" (Nairn, 2002, 375). Windows version 1.0 was released the same year that the license was granted in 1985.

Both Microsoft and Apple then worked on GUI-based operating systems as a way to provide an easy-to-use solution for consumers. Although neither the first Microsoft Windows release nor the Macintosh computer proved to be commercially successful, GUI-based operating systems soon allowed massive diffusion of PCs to the consumer market. Microsoft held its IPO in 1986, which earned $61 million in cash, which the company used to invest heavily in developing its Microsoft Windows operating system. Microsoft emerged as the clear winner during this period, and the company's relationship with IBM ensured that its operating system would be installed on IBM-compatible computers. Microsoft's growth during this period was immense, as reported earlier in this chapter when its market share rose to 90% by some estimates (Gilbert, 1995). This growth in market share coincided with an increase in revenues, and the Windows operating system with its GUI elements was the key product that fueled the growth. However, Apple challenged Microsoft's claims to the GUI elements of Windows, claiming that Microsoft had infringed its intellectual property.

Apple Computer vs. Microsoft Corporation

In 1988, Apple began a copyright infringement lawsuit against Microsoft. Apple claimed that Microsoft had infringed on 189 elements of its GUI, which, when taken together, constituted a “look and feel” of its Macintosh operating system that was protected by copyright. Apple claimed that the infringements occurred in version 2.03 and, later, 3.0 of Microsoft Windows. The lawsuit stemmed from the initial licensing agreement that was negotiated between Apple and Microsoft when Apple granted Microsoft access to its GUI for developing applications for the Mac. The resulting litigation lasted four years, but the case was interrupted by Xerox bringing a suit against

Apple, whereby Xerox claimed Apple had violated its copyrights by using some of the GUI elements originally featured in its PARC operations. Xerox further claimed that Apple was guilty of unfair business practices because of its copyright claims on the GUI, which made it difficult for Xerox to license the technology to other customers. The case against Apple grew out of the meetings held between Xerox and Apple when Steve Jobs and other Apple representatives visited the Xerox PARC to see prototypes of the GUI in exchange Xerox's ability to acquire stock prior to Apple's IPO.

Xerox's claims against Apple were ultimately dismissed, as Apple claimed that, while it may have borrowed *ideas* from Xerox's PARC, those ideas were not able to be protected by copyright, and Xerox ought to settle any remaining dispute with the Copyright Office (Pollack, 1990). Similarly, Apple's case against Microsoft was rejected. Of the 189 claims of copyright infringement, all but 10 were dismissed. In the end, the district court ruled in favor of Microsoft, claiming that the remaining 10 claims were over ideas rather than expressions that could be protected by copyright. Furthermore, the original licensing agreement signed between Microsoft and Apple that allowed Microsoft access to the GUI developed by Apple granted Microsoft the “right to transfer individual elements or design features using its 'Windows' program” (*Apple Computer, Inc. v. Microsoft Corporation, 1994*).

While the 1994 case may not be directly related to corporate involvement in FLOSS, it does illustrate a couple things about software development, intellectual property, and Microsoft. First, the case demonstrates that early software development, particularly of those features that we may take for granted today like the GUI, was not the result of rugged individuals developing the technology alone. Rather, technological

development is a collective and collaborative process in which the ideas of others can influence the direction of development.

Second, the case is instructive for the exploitation of intellectual property, specifically because it illustrates how original authorship can be separated from ownership (Bettig, 1992). While the idea and design for the GUI may have originated in Xerox's PARC, Xerox had not commercially exploited its designs in the same way that Apple and Microsoft sought to do. Through a series of licensing agreements – first between Apple and Xerox, and later, between Apple and Microsoft – the rights to the individual elements of the GUI became diffused as they were shared among peers. Microsoft was already in a strong market position to be able to exploit the GUI through its Microsoft Windows operating system, whereas Apple relied on assistance from Microsoft for developing applications for its emerging Macintosh computer. By doing so, however, Apple gave access to its GUI operating system to Microsoft. In turn, Microsoft honored the stipulations of its original licensing agreement with Apple, but it would later continue development of its Windows operating system by making use of some of the same elements that Apple had been using. Furthermore, because Microsoft maintained strategic alliances with major information technology manufacturers, the company was in a position to ensure that its operating system could be commercially exploited as its market share for personal computer operating systems rose to nearly 90% during the 1990s (Gilbert, 1995).

Third, there is a great contradiction at the heart of this case when compared with the history of Microsoft. Although the company benefited from sharing ideas to develop its Windows operating system, the company relied heavily on strong intellectual property protections to exclude others from its software as it ruthlessly defended its position atop

the software industry throughout the 1990s. As we will see, however, this ruthlessness is ultimately what led to investigations for antitrust violations.

Microsoft in the 1990s

Microsoft's partnership with IBM was what ultimately allowed the company to solidify its strategic position atop the computer software industry. Sales of the IBM PC and its clones reached nearly 16 million by 1990, which represented nearly 84% of the market share for personal computers (Reimer, 2005). Originally, Microsoft had teamed with IBM to produce the OS/2 operating system, which IBM intended to be included on its PCs, but Microsoft was busy working on its Windows operating system. By the time Windows 3.0 was released in 1990, the relationship between IBM and Microsoft had become strained to the point that the companies decided to terminate their Joint Development Agreement,[22] which specified the partnership between the two firms for the purpose of working on OS/2 (TechInsider.org, 2013). Because the Windows operating system was much more developed when the companies ended their relationship, Microsoft rapidly picked up market share as its operating system was included on sales of IBM-compatible PCs. Indeed, the relationship between IBM and Microsoft was what initiallly drew attention from the United States Federal Trade Commission (FTC) in 1990.

The investigation by the FTC was initiated because of a joint news release by IBM and Microsoft during the Comdex trade show in Las Vegas, NV, on November 13, 1989 (Wallace & Erickson, 1992). In the press release, the companies claimed that

22 A digitized version of the Joint Development Agreement is available at http://tech-insider.org/os2/research/acrobat/871126.pdf (last accessed July 3, 2014).

“Microsoft would hold back features for Windows in order to help industry acceptance of the OS/2 operating system” (Wallace & Erickson, 1992, 373). The FTC was concerned that the companies were colluding to control the market for operating systems. Ultimately, the FTC investigation ended in 1993 when the commissioners were split 2-2 on whether to bring an administrative action against Microsoft. In the same year, however, the Antitrust Division of the United States Department of Justice (DoJ) picked up the investigation, which would eventually lead to Microsoft's conviction for antitrust violations. The main issues in that case, however, did not center around Microsoft's control of the operating system market but business practices associated primarily with its Internet browser, Internet Explorer. Around the same time that Microsoft was seeking to solidify its position atop the computer software industry, however, at least three concurrent technological developments and their attendant cultural practices were emerging as challengers to the model used by Microsoft in its rise to power. These developments were the emergence of the World Wide Web, the development of graphical web browsers, and the creation of Linux. While the introduction to this dissertation discussed the rise of Linux in the early 1990s, the following section will focus more specifically on the so-called “browser wars” that followed the rise of the World Wide Web.

The Browser Wars

In November of 1990, Tim Berners-Lee and Robert Caillau authored a proposal for a hypertext project called the World Wide Web, which would provide “a way to link and access information of various kinds as a web of nodes in which the user can browse at will” (Berners-Lee & Caillau, 1990). The creation of such a project relied on server-

level applications to manage the nodes stored on the server and to facilitate the display and access of those nodes with a browser. Browsers served as the application running on a user's machine that could request access to the nodes stored on the server and display those nodes to the user. Finally, web pages would need to be created that could store textual, graphical, or other types of information that could be accessed by users. By the end of the year in 1990, models of all these components had been created, and companies began developing browsers that would allow users to access the burgeoning technology of the World Wide Web.

Mosaic and Netscape

In 1993, the Mosaic web browser was developed by a team of researchers at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. The browser had the ability to display graphical content on the web and, although it was not the first browser to do so, Mosaic would drastically increase the popularity of browsing the web. Prior to its creation, most of the pages on the World Wide Web had been primarily text-based. However, Mosaic's place in the history of web browsers is perhaps best illustrated by tracing the history of its ownership and, ultimately, its transformation into the open-source web browser, Mozilla Firefox.

From its beginnings at the NCSA at the University of Illinois, the Mosaic browser spawned at least two primary companies that sought to commercially exploit the browser's technology. One company was called Mosaic Communications, and the other was Spyglass. The code base for the Mosaic browser was handled by Spyglass after an agreement was signed between the company and the University of Illinois, whereby Spyglass would retain the rights to commercially exploit the code. The other company,

Mosaic Communications, created the Mosaic Netscape browser. In fact, many of the employees at Mosaic Communications had worked previously on the Mosaic browser at the NCSA, although the Netscape browser was built entirely by the team at Mosaic Communications. What was truly novel about the Netscape browser, however, was that it was made freely available to the general public for personal use, which was unprecedented up to that point. Moody (2001) describes the significance of this strategy:

> Along with a beta-testing program on a scale that was unprecedented, the decision to allow anyone to download copies of Netscape free had another key effect: It introduced the idea of capturing market share by giving away free software, and then generating profits in other ways from the resulting installed base. In other words, the Mosaic Netscape release signaled the first instance of the new Internet economics that have since come to dominate the software world and beyond. (187).

Indeed, the Netscape browser began to pick up market share, and the University of Illinois noticed. To resolve any additional trademark disputes with the university, Mosaic Communications changed its name to Netscape Communications and reissued its browser under the name Netscape Navigator (Moody, 2001).

Netscape Navigator quickly picked up market share from 1994-1996, reaching its peak at nearly 90% in April 1996, according to some sources (Cusumano & Yoffie, 1998). Riding this extraordinary wave of enthusiasm for Netscape, the company held its IPO in August 1995. On the day of its IPO, shares of the company began selling at $28 and reached $58.25 by the end of the day, valuing the company at nearly $3 billion after only 18 months of operation (Moody, 2001). At that point, Netscape's IPO was the largest in history. The success of Netscape was not lost on Microsoft, and the company began to focus its efforts on developing a browser to rival Netscape. Thus began the first "browser wars."

Microsoft Responds

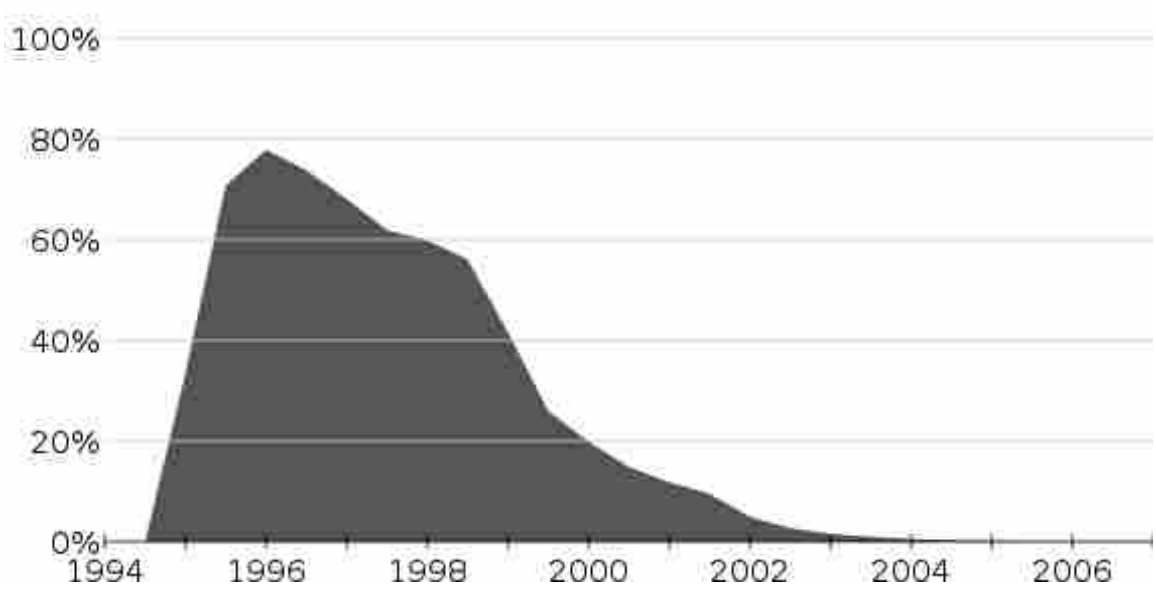
Since Microsoft had not devoted any significant amount of time and resources to developing a web browser of its own, the company decided not to build its browser from scratch. Rather, Microsoft approached Spyglass, which held the rights to the code base of the original Mosaic browser. Spyglass had been developing its own version of Mosaic, known as Spyglass Mosaic. Microsoft negotiated a license to use the Spyglass Mosaic code base in exchange for royalty payments for each copy of the browser issued, with an annual cap of $5 million (Elstrom, 1997).[23] The resulting browser was called Internet Explorer (IE), which was based on the same foundation as Netscape. As evidence of how aggressively Microsoft pursued its new Internet strategy, Page and Lopatka (2007) note that the company only had five or six employees working in the browser department in 1995 but, the company had more than 1,000 by 1999.

In addition to assigning more employees to the browser division, Microsoft began packaging IE with distribution of its Windows operating system. As Microsoft had nearly 90% of the market for operating systems because of its contractual relationships with Original Equipment Manufacturers (OEMs), the company was able to quickly make gains in the market for web browsers. In effect, Microsoft was giving away copies of IE for free by bundling it with its Windows operating system. To do so, the company began distributing versions of IE to OEMs by sending discs to the manufacturers, and eventually required the OEMs to install IE with Windows 95. OEMs were prohibited from "modifying or deleting any part of Windows 95, including Internet Explorer, prior to shipment" because of a non-negotiable licensing restriction that Microsoft placed on

23 This agreement would become a point of contention between Spyglass and Microsoft, as tracking the exact number of IE copies issued proved to be incredibly difficult. Ultimately, the dispute was settled in 1997 after Microsoft agreed to issue a one-time payment of $7.5 million and an additional $500,000 in "software and other considerations" to Spyglass (Elstrom, 1997).

OEMs (United States, 1999, see Finding 158). This restriction did not allow OEMs to ship new PCs without IE installed. The effect on the market for web browsers was almost immediate. Figure 6.1 shows the sharp rise in market share for the Netscape browser, and its eventual sharp decline.

Figure 5.1. Netscape Navigator Usage Data 1994-2006



Source: Image has been released to the public domain, and is available via Wikimedia Commons at http://commons.wikimedia.org/wiki/File:Netscape-navigator-usage-data.svg

Because of these tactics, Microsoft and its Internet Explorer won the first browser wars. Microsoft was simply too big and had too much power to influence the market for Netscape to compete. However, the novelty of distributing software freely for personal use was not lost on Microsoft. Netscape's Navigator browser had rapidly picked up market share by using such a tactic, and Microsoft effectively gave away its IE browser by bundling it with its Windows operating system. Just as Microsoft was reaching its most dominant market position and using tactics that eventually led to its conviction for antitrust violations, Linux and the open-source model of production was beginning to grow as a potential threat. Indeed, after Netscape Navigator had lost significant market

share to Microsoft, Netscape released the source code to the broader community in 1998 as a way to attract development for a new browser. That new browser would eventually become Mozilla's Firefox, which was first released in 2002. Microsoft took notice of this general trend toward open source as well and, in 1998, a series of leaked documents demonstrated exactly how Microsoft viewed this emerging threat. The Halloween Documents[24] were made publicly available and their authenticity was later confirmed by Microsoft (Harmon & Markoff, 1998). They will be discussed later in this chapter. Before doing so, however, Microsoft's conviction for antitrust violations needs to be discussed. In many ways, the antitrust conviction marks an important turning point, not just in Microsoft's history but in the broader history of the software industry.

The United States vs. Microsoft Corporation

Microsoft's behavior during the Browser Wars was what ultimately led to its conviction for violations of Sections 1 and 2 of the Sherman Act. Section 1 of the Sherman Act prohibits “every contract, combination..., or conspiracy, in restraint of trade or commerce...” (15 U.S.C. §1). Section 2 states it is unlawful for any person or firm to “monopolize...any part of the trade or commerce among the several States, or with foreign nations...” (15 U.S.C. §2). The court ultimately found Microsoft to be in violation of both sections of the Act. Microsoft violated Section 1 by unlawfully tying its web browser – Internet Explorer – to its operating system. Furthermore, the company violated Section 2 by maintaining its monopoly power by anticompetitive means and attempting to monopolize the web browser market.

24 The Halloween Documents can be found at http://www.catb.org/esr/halloween/ (last accessed July 3, 2014).

These convictions rested upon the fact that Microsoft engaged in anticompetitive behaviors in its contractual relationships with Original Equipment Manufacturers (OEMs). In particular, Microsoft used "contractual and, later, technological shackles in order to ensure the prominent (and ultimately permanent) presence of Internet Explorer on every Windows user's PC system, and to increase the costs attendant to installing and using [Netscape] Navigator on any PCs running Windows" (United States, 2000, 11). In addition, Microsoft restricted OEMs from reconfiguring Windows 95 and Windows 98 in ways that could lead to greater use of Netscape Navigator. Finally, Microsoft "used incentives and threats to induce" certain OEMs from designing "distributional, promotional and technical efforts" that would favor Internet Explorer instead of Navigator (United States, 2000, 11).

The final judgment in the antitrust case found that Microsoft had violated sections one and two of the Sherman Act, as well as more than 35 state law provisions in 19 states plus the District of Columbia. In light of these violations, the U.S. District Court Judge, Thomas Penfield Jackson, ordered Microsoft to divest its operating systems business operations from its applications business operations. Furthermore, all the intellectual property rights previously held by the two businesses were to be transferred to the Applications Division, which was required to grant a perpetual, royalty-free license to the operating systems business so that it could license, develop, and distribute modified or derivative versions of the intellectual property. However, the Operating Systems Division was prohibited from doing this with the intellectual property related to the Internet browser (Internet Explorer). Aside from divesting the operations of these two businesses, Microsoft was ordered to transfer all the assets from either one of the divisions into a newly formed company, for which the transfer of ownership was to be

accomplished by a distribution of stock to shareholders not connected with Microsoft. The intent of these decrees was to separate Microsoft's operating system business from the business operations that handled its web browser development. These actions would prevent Microsoft from engaging in the same types of anticompetitive behavior that it had used during the Browser Wars.

Effects of the Decision

In 2001, District Judge Thomas Penfield Jackson recused himself from the case because of some public comments that he made, which gave the impression that he had a personal bias or prejudice against Microsoft (Wilcox, 2001). In his place, U.S. District Judge Colleen Kollar-Kotelly took over the case and, in late 2001, approved a settlement between the parties. The approved settlement would no longer seek the break up of Microsoft's Operating Systems and Applications Divisions. Instead, Microsoft agreed to a series of consent decrees in November, 2002, whereby the company would be prohibited from retaliating against any OEM that develops, distributes, promotes, uses, sells, or licenses any non-Microsoft products (United States, 2002). In addition, Microsoft would need to establish a clearly documented schedule of all royalties that would be received from OEMs for its Windows Operating System.

These provisions were aimed at prohibiting Microsoft from engaging in any anticompetitive behaviors, but most importantly for the purposes of this analysis, Microsoft would also be forced to promote interoperability for its products. This would ensure that other companies could develop products that would operate smoothly with Microsoft's products. As such, Microsoft was ordered to disclose its Application Programming Interfaces (APIs), which would specify how software components should

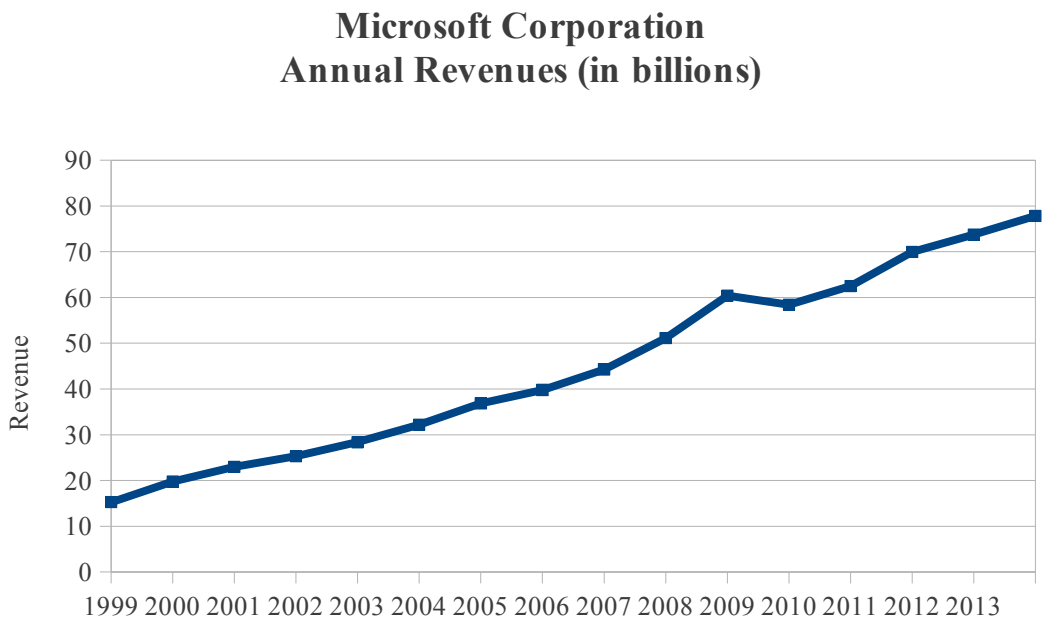
interact with one another. By releasing its APIs to independent hardware vendors (IHV), independent software vendors (ISV), OEMs, Internet Access Providers (IAPs), and Internet Content Providers (ICP), those parties could develop software that could operate on and interact with Microsoft's operating systems and other software. Microsoft would also need to make *any* communications protocol available to third parties for the same purposes. The consent decrees to which Microsoft agreed were supposed to last five years from the decision in 2002. However, these decrees were renewed twice – once in 2006 and again in 2009 – and finally expired May 12, 2011 (Chan, 2011).

In effect, the antitrust ruling against Microsoft did not seek a breakup of the company into distinct operating units, but focused more specifically on Microsoft's intellectual property practices. The decrees forced Microsoft to disclose its APIs to third parties as a way to encourage and support interoperability with its products. The logic was that doing so would curb the anticompetitive behavior Microsoft had displayed during the browser wars and in its contract bargaining with OEMs, while promoting competition within the software industry. It is within this context that Microsoft's shift toward (but not completely to) open source can be viewed.

Nevertheless, the consent decrees had little effect on the economic performance of the company. Figures 5.2 and 5.3 illustrate Microsoft's economic performance in the wake of the antitrust conviction and the consent decrees. Figure 5.2 presents the company's annual revenues in billions of dollars. Clearly, the company's annual revenues have continued to grow, and revenues were not affected by the dot-com crash that negatively affected the United States economy during 2001. Indeed, the same could be said of the company's profits during the same time, which are shown in Figure 5.3. The

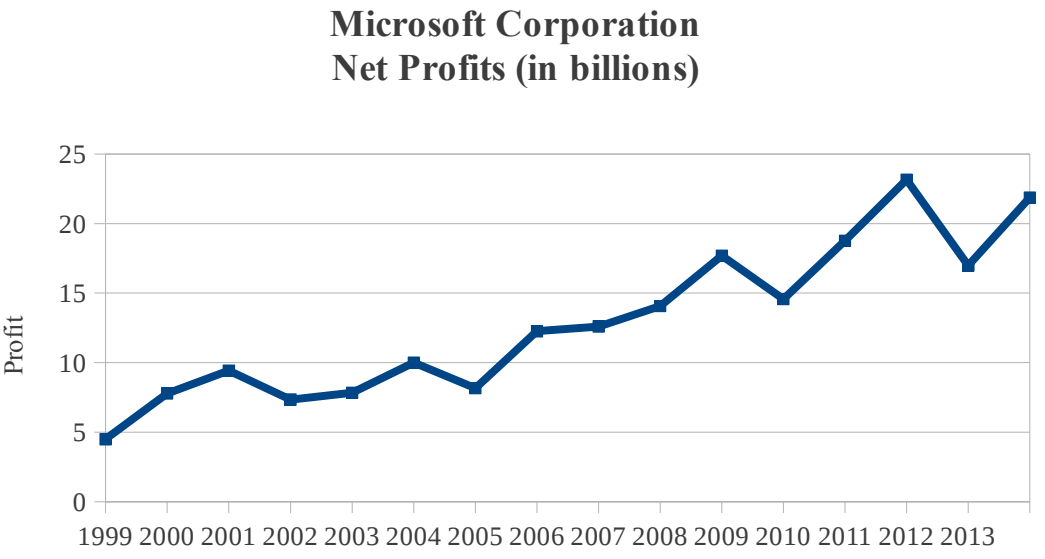
company experienced a dip in profits in 2001, but still maintained nearly $7 billion in profits during this time with a substantial jump in the 2005-2006 fiscal year.

Figure 5.2. Microsoft Annual Revenues 1999-2013



Source: Data is from Microsoft's 10-K Annual Reports filed with the SEC for the years 1999-2013.

Figure 5.3. Microsoft Annual Net Profits 1999-2013



Source: Data is from Microsoft's 10-K Annual Reports filed with the SEC for the years 1999-2013.

In sum, the consent decrees did little to affect the overall economic performance of Microsoft. However, along with broader shifts occurring in the software industry at the time, they did have the effect of changing some of Microsoft's practices associated with open source. Indeed, the date of the consent decrees perfectly coincides with Microsoft's creation of the Shared Source program. Furthermore, the end of the consent decrees in May, 2011, coincides with the creation of the Microsoft Open Technologies Division in 2012. To understand more fully Microsoft's relationship with FLOSS, the remainder of the chapter charts the company's history with FLOSS, beginning with the Halloween Documents, then discusses the Shared Source program and Microsoft Open Technologies. The previous discussion in this chapter, provides an important context within which Microsoft's shift toward FLOSS can be interpreted.

The Halloween Documents

In October 1998, Eric Raymond, who was a well-known member of the free and open source software community and author of *The Cathedral and the Bazaar*, received a series of internal documents from a confidential source that outlined Microsoft's strategy against Linux and open source software. These documents were subsequently released to the public by Raymond and their authenticity was later verified by Microsoft. These documents became known as the Halloween Documents because many were released near the end of October over different years. The Halloween Documents focus on Microsoft's assessment of the strengths and weaknesses of open source software, including Linux, and how the company could combat the growing popularity of the movement. What is clear from the documents is that Microsoft viewed free software products as a genuine threat to its own products, especially because the free software

projects had “acquired the depth and complexity traditionally associated with commercial projects” (Raymond, 1998a). As such, the Halloween Documents contain information about how Microsoft planned to combat open source software as a competitor.

In Halloween Document I,[25] Vinod Valloppillil discusses open source software as a potential threat to Microsoft. Rather than focusing on a particular open source project or organization, however, Valloppillil focuses on the *process* used in open source development. Valloppillil writes, “to understand how to compete against OSS [open source software], we must target a process rather than a company” (Raymond, 1998a). The author goes on to assess possible strategies for combating open source software, and gives special attention to “FUD tactics,” which is an acronym for Fear, Uncertainty, Doubt. FUD is a tactic used in sales, marketing, public relations, and propaganda, whereby one attempts to instill those feelings in consumers about the quality of a competitors' products. For example, in an advertisement for Microsoft Server 2003, Microsoft claimed that research demonstrated “Linux was found to be over 10 times more expensive than Windows Server 2003” (BBC News, 2004). Microsoft was asked to change the advertisement by the Advertising Standards Authority in the United Kingdom because the results of the study were deemed to be misleading to consumers. In effect, the advertisement was viewed as a way to instill FUD in consumers about the total cost of ownership for Linux.

Halloween Document II[26] largely contains a much more detailed technical analysis of Linux's functionality when compared to other products. The author also describes his

25 Halloween Document I, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween1.html (last accessed July 3, 2014).

26 Haloween Document II, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween2.html (last accessed July 3, 2014).

personal experience with installing the DHCP Client Daemon and ultimately claims that, even though he was a poorly skilled UNIX programmer, he was able to easily figure out how to extend the DHCP client code and "the *feeling* was exhilarating and addictive" (Raymond, 1998b). Importantly, however, the conclusion of the document suggests possible strategies for competing against Linux. While the author concedes that Linux was the greatest threat to Microsoft in the server market, he claims that a possible strategy for fighting Linux may have been patent and copyright litigation.

Halloween Document III[27] is a document from Microsoft Netherlands in which Aurelia van den Berg, Press and Public Relations Manager, responds to the leak of the two internal documents in 1998. Her response tends to downplay the significance of the leaked documents, claiming that all companies conduct assessments of their competitors, and that the leaked documents do not represent official Microsoft positions. At the end of the document, however, van den Berg still manages to criticize FLOSS in general for its inability to be a long-term solution. Alluding to the need for strong intellectual property protections, van den Berg claims, "unless Linux violates IP rights, it will fail to deliver innovation over the long run" (Raymond, 1998c).

Documents VII, VIII, and X are the other documents that were directly leaked from Microsoft. The remaining documents are commentaries, satires, and criticisms of Microsoft created by others in response to the leaked documents. Halloween Document VII[28] provides the results of an internal survey conducted by Microsoft in 2002 about attitudes and opinions of FLOSS in general, Linux in particular, and the general

27 Halloween Document III, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween3.html (last accessed July 3, 2014).

28 Halloween Document VII, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween7.html (last accessed July 3, 2014).

familiarity with Microsoft's newly created Shared Source program, which will be discussed later in this chapter. The results of Microsoft's internal survey showed that FLOSS in general and Linux in particular were viewed favorably by those included in the survey, which mainly included policymakers, decision makers, and corporate executives selectively chosen by Microsoft. The survey also showed that messaging designed to criticize or question the quality of FLOSS, Linux, or the GPL were not effective (Raymond, 2002a). In light of these findings, the authors recommend that Microsoft could more effectively compete with FLOSS by focusing on the total cost of ownership (TCO) of Microsoft products when compared with Linux. In addition, Microsoft should focus on the benefits of its newly created Shared Source program.

Halloween Document VIII[29] was an internal email sent by Orlando Ayala, Group Vice President of Microsoft's Worldwide Sales, Marketing, and Services Group, to the heads of Microsoft's subsidiaries in 2002. The message was sent as a reaction to the fact that many governments and other large institutions had begun to transition to Linux. As such, Ayala suggests that Microsoft and its subsidiaries needs to be better prepared to respond to those types of announcements by communicating these announcements internally so the company can try to respond to these cases directly. In short, the document suggests that Microsoft's internal communication needed to be more fully integrated to respond to their declining market share, particularly among large institutions.

29 Halloween Document VIII, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween8.html (last accessed July 3, 2014).

Finally, Halloween Document X[30] was leaked in 2004 and features an internal email from the SCO Group in which the author discusses, albeit somewhat vaguely, the relationship between SCO Group and Microsoft. The email appears to disclose the amount of money paid to SCO on behalf of Microsoft. Although not discussed at length here, the SCO Group was a software company that became infamous for engaging in a number of legal battles over alleged intellectual property infringement in Linux related software. The SCO Group went bankrupt in 2007, but between 2003 and 2011 the company alleged that various Linux vendors had infringed copyrights belonging to the SCO Group. These vendors notably included IBM, Novell, and Red Hat, but also included claims against Daimler-Chrysler and AutoZone. Particularly relevant for this discussion is that Document X suggests that Microsoft was contributing large amounts of money to the SCO Group as a way to fuel the intellectual property litigation against Linux and its vendors. This would be consistent with some of the suggestions in the previous documents that possible strategies for combatting Linux would be copyright and patent litigation.

In sum, the Halloween Documents allowed direct access to Microsoft's assessment of FLOSS in general and Linux in particular. What becomes clear from the documents is that Microsoft believed Linux was a legitimate threat to its own products. However, Microsoft correctly placed the true value of FLOSS projects within the process of production. To compete against the perception that FLOSS projects provided at least the same level of quality as those of proprietary companies, Microsoft used FUD tactics to suggest that the open source model of production was inherently unstable or not

30 Halloween Document X, along with Eric Raymond's commentary, can be accessed at http://www.catb.org/esr/halloween/halloween10.html (last accessed July 3, 2014).

secure. Ironically, Microsoft's own survey data suggested that these tactics were not effective, nor were any attempts to criticize the FLOSS development model. Instead, Microsoft needed to shift their strategy to focus more on the quality of its own products, including its newly developed Shared Source program. The Halloween Documents provide an illuminating perspective on the internal culture of Microsoft during the critical years from 1998-2004 when it underwent somewhat of a transformation. The antitrust suit against the company began in 1998 and was ultimately decided in 2001, and the company developed its Shared Source program in 2001.

Shifting Toward the Commons

Three concurrent factors ultimately led to Microsoft's change of position in regard to FLOSS. First, the company was convicted of antitrust in 2001 and agreed to a series of consent decrees in 2002 that sought to curb the company's anticompetitive practices by requiring Microsoft to disclose its APIs to third parties. Second, the dot-com bubble burst, which marked the end of the massive speculative investment in web-based companies. Third, the rise of Linux and Linux-related businesses had demonstrated the commercial viability of FLOSS-based business models. Microsoft responded to these factors by initiating a couple different projects that were claimed to be dedicated to FLOSS principles, although these initiatives were met with different levels of acceptance by the broader FLOSS community. The next sections chart the rise of two such projects: the Shared Source Initiative and the Microsoft Open Technologies Division. Because the Microsoft Open Technologies division is still relatively new, however, extensive information about its operations is limited. Therefore, I attempt to position the opening

of the division within the broader historical trajectory of Microsoft's shift after the antitrust ruling.

Microsoft Shared Source

The Shared Source Initiative (SSI) began at Microsoft in 2001 as a way to provide access to certain source code for debugging and reference purposes. While Microsoft had been releasing portions of its Windows source code to academic institutions and OEMs as early as 1991, the SSI expanded the range of code that was made available in 2001. The code made available under this program was protected by a number of different licenses, including the Research Source Licensing Program, Enterprise Source Licensing Program, ISV Source Licensing, OEM Source Licensing, Windows CE source code access, and others. While a full description of the rights granted by these licenses and programs is well beyond the scope of this analysis, these licenses are mentioned here as a way to demonstrate that the sharing of source code by Microsoft was not entirely new to the period following the antitrust ruling. However, these licenses were not considered free software or open source in its true sense because Microsoft still claimed copyright protection on the underlying source code. Under most of these licenses, code was made available for academic and reference purposes, but the company prohibited redistribution of the code or limited distribution to those working on Microsoft software. In effect, these licenses served as a way to allow others to *view* the source code, but it could not be modified unless it adhered to the limitations set forth in the licenses.

What was novel about the SSI in 2001 was the expansion of its Shared Source program by the release of more types of source code as well as the creation of new licenses that were designed to grant different types of rights to users. Most notably for

the purpose of this project are the two licenses that were submitted to the Open Source Initiative (OSI) for official registration as open source licenses: the Microsoft Public License and the Microsoft Reciprocal License. Both were approved by the OSI in October of 2007 (Open Source Initiative, 2007). This marked the first time that Microsoft officially had a license approved by the open source community, even though these licenses were still not fully compatible with the GPL.

Indeed, some within the broader community viewed Microsoft's Shared Source Initiative and its new licenses as simply a marketing ploy. Even Michael Tiemann, the president of OSI, the organization that approved the licenses, claimed:

> Shared source is a marketing term created and controlled by Microsoft. Shared source is not open source by another name. Shared source is an insurgent term that distracts and dilutes the Open Source message by using similar-sounding terms and offering similar-sounding promises. And to date, “share source” has been a marketing dud as far as Open Source is concerned (Tiemann, 2007).

Of course, Microsoft's views differed from Tiemann's claim. In a speech in 2001, Microsoft Senior Vice President, Scott Mundie, noted that Microsoft's expansion of its Share Source Initiative may be viewed by some as a failed attempt at becoming an open source company. Mundie claimed this assertion would be false because, “Shared Source is Open Source” (Mundie, 2001). Mundie continued by saying Microsoft would be incorporating many of the positive aspects of the FLOSS development, while continuing to preserve the company's strong intellectual property protections. Mundie went on to claim that FLOSS production was unstable as a business model in the long run because it was not secure and subject to “unhealthy 'forking'” (Mundie, 2001).

These vastly different assessments of the SSI are indicative of the contentious relationship that exists between Microsoft and the FLOSS community. Although

Microsoft had shifted its position toward FLOSS, the community still maintained a healthy skepticism about Microsoft's involvement in FLOSS projects. After all, Microsoft had a history of threatening intellectual property infringement suits against firms using Linux, even though this stance began to thaw around the same time that Microsoft's Shared Source licenses were approved by the OSI. In 2006, Microsoft agreed not to sue Novell's Linux users in exchange for a share of Novell's open source revenue, as Microsoft claimed that Novell was infringing its intellectual property. As a result of the agreement, Novell claimed that its Linux business had increased 243% through the first three quarters of the 2007 fiscal year (Lai, 2007). This agreement, as well as other similar agreements between companies using Linux and Microsoft, caused somewhat of a split within the FLOSS community as to whether companies should be signing such agreements. While the split existed in 2007, the lines of this split have blurred significantly in the years since these types of agreements began. Indeed, Microsoft has now opened an entire division of its company dedicated to open source, which will be discussed in the next section.

Microsoft Open Technologies

Microsoft Open Technologies opened in 2012 as a wholly owned subsidiary to build “bridges between Microsoft and non-Microsoft technologies” (Microsoft Open Technologies, 2014a). To do so, the subsidiary claimed to promote interoperability through open standards and open source. One of the primary ways this is accomplished is the building of open source code, which is hosted on the popular web-based development platform GitHub, as well as Microsoft's own CodePlex platform. As of 2014, the company claimed to have 25 projects available on GitHub and CodePlex

(Microsoft Open Technologies, 2014b). These projects appeared to serve a variety of purposes that were grouped into six thematic areas: cloud-based services, data and business intelligence, device applications, open web, virtual machines, and devops.[31]

At the time of writing, it was still too early to tell whether the specific projects hosted by Microsoft Open Technologies would be successful. More generally, however, the creation of an entire subsidiary dedicated to open source at least signals a shift in Microsoft's relationship to the broader open source community. Throughout Microsoft's history, isolated individuals or smaller working groups may have advocated for greater involvement in open source projects, but the creation of an entirely new subsidiary marked the first concerted institutional effort at direct involvement. Notably, the creation of the new subsidiary coincided with two major events at Microsoft. The first was the expiration of the consent decrees in 2011, and the second was the resignation of Steve Ballmer as Chief Executive Officer.

The consent decrees required Microsoft to make its APIs more openly available so that developers could create technologies that could easily interact with Microsoft's technologies. In other words, the consent decrees provided an impetus for forcing the promotion of greater interoperability between Microsoft and non-Microsoft technologies. In addition, Microsoft expanded its Shared Source initiative as a way to make its code more openly available to the broader community. However, this move was met with some skepticism by the FLOSS community, particularly because most of the licenses that protected the code did not comply with open source standards. This changed in 2007 when the OSI approved two Microsoft licenses as open source.

31 “devops” is a portmanteau combining the terms “development” and “operations,” which is used to describe a software development method.

In addition to the changes brought about by the consent decrees, Microsoft experienced a change in leadership shortly after Microsoft Open Technologies opened. CEO Steve Ballmer, who is credited with the “Linux is a cancer” indictment, announced his resignation on August 23, 2013. He ultimately resigned in 2014, and Bill Gates stepped down as Chairman of the company. However, Gates was invited to serve as technology adviser to the newly appointed CEO, Satya Nadella. The shift in leadership could similarly signal a new direction for Microsoft, although it is still far too early to tell. What is clear, however, is the notable shift in Microsoft's stance toward open source.

Why Open Source? Why Now?

Microsoft's transformation in regards to open source can be interpreted within broader historical shifts in web technology. On the one hand, the company's initial strategy of relying on strong intellectual property rights and enforcing them ruthlessly while simultaneously framing open source as an adversary ultimately led to an antitrust conviction shortly after the turn of the 21st century. Throughout the 1980s and 1990s, Microsoft's closed-source strategy led to tremendous growth within the software market. The findings of the antitrust case, however, revealed the darker side of this growth. Mainly, that the company engaged in monopolistic practices by using its dominance in the market for personal computer operating systems to distribute copies of its Internet Explorer web browser. This marked an historical turning point not just for Microsoft, but of a more general trend that saw the end of the dot-com bubble and Web 1.0.

Microsoft was, and still remains, the largest software company in the world, and the company managed to survive the burst of the dot-com bubble. Indeed, as demonstrated in this chapter, the company was able to thrive in its wake. But in the years

shortly after the dot-com bubble burst in 2001, a host of new web-based companies arose that promised interactivity and a focus on the consumer. This era, which marks the rise of so-called “Web 2.0” companies, was characterized by companies providing services rather than packaged software, controlling robust data sets that expanded as more people use them, trusting users as co-developers of companies' products and services, harnessing collective intelligence, relying on customer self-service, providing software across multiple devices, and featuring lightweight user interfaces, development models and business models (O'Reilly, 2005). These technological features functioned ideologically insofar as they gave the illusion of participation, collaboration, and egalitarianism when, in fact, they merely justified the provision of personal data to corporate Internet Service Providers (ISPs), who, in turn, harvested and sold that data to advertisers (see Fuchs, 2011b).

This suggests that the antitrust ruling cannot be viewed as the sole factor that affected Microsoft's business model. Rather, the antitrust decision combined with the other emerging historical forces within the technology field – Web 2.0, the commercial viability of Linux, and the ideology of romantic individualism within start-up culture – to effect a change in Microsoft's business strategy. In 2002, only a year after the antitrust ruling, Microsoft launched its “shared source” program, which provided greater access to some of its source code, but still placed restrictions on its modification and redistribution. Consequently, the program was widely viewed as somewhat of a marketing ploy and a strategy to gain a better reputation with the open source community.

When viewed in this way, Microsoft needed to embrace open source – not only because the consent decrees required a more open approach, but because the industry in general was trending toward collaboration and Linux was proving to be commercially

viable. In part, Microsoft has an interest in promoting interoperability and open standards, which enables it to keep up with the always-changing technological landscape. But the company's turn to open source may also be viewed as a humble recognition that the commons-based peer production taking place within the FLOSS community was an efficient and effective model of production that could supplement its own business practices.

In sum, Microsoft represents an example of how a corporation that was widely viewed as the antithesis to the FLOSS ethos transitioned toward a more open stance toward it. In effect, Microsoft is now seeking to incorporate elements of FLOSS production within its broader corporate structure. While not fully transforming to an open source business, Microsoft has shifted its position even while it maintains strong intellectual property protections over some of its core software. Consequently, Microsoft does not seem poised to fully embrace FLOSS, but it also does not seem to be fully competitive. The decision to collaborate or compete with the broader FLOSS community will most likely be based on the company's assessment as to its relative strengths and weaknesses in certain areas of software.

In the meantime, Microsoft will need to attract FLOSS developers to work on its open source projects. However, this is not without potential pitfalls. The following chapter presents a case in which a company that supported FLOSS projects was acquired by another company that had other intentions for those projects. In response to this undue corporate encroachment into their FLOSS projects, the community took certain measures to resist such involvement, ultimately abandoning production on those projects. More specifically, the following chapter discusses Oracle's acquisition of Sun

Microsystems and the effect this acquisition had on three software projects: OpenSolaris, MySQL, and OpenOffice.

CHAPTER VI

CONFLICT IN THE COMMONS: ORACLE CORPORATION AND ITS ACQUISITION OF SUN MICROSYSTEMS

The previous two chapters focused on case studies of Red Hat and Microsoft, respectively. Red Hat demonstrates how free software can be turned into a profitable business, and the company still maintains a good relationship with the broader open source community today. Microsoft demonstrates how a company that depends on strong intellectual property to protect its proprietary software eventually shifted to embrace open source, albeit in limited and only certain ways. This chapter will look at how another one of the largest software companies in the world, the Oracle Corporate (simply "Oracle" hereafter), has tried to incorporate open source projects into its corporate structure. Oracle did this by acquiring Sun Microsystems, which supported open source software projects. Whereas Sun Microsystems (simply "Sun" hereafter) maintained a good relationship with the open source community, these relations became strained after Oracle acquired the company in 2010. After the acquisition, Oracle used a differnet strategy with regards to Sun's open source projects. In certain cases, Oracle ended open source activities, in others it tried to influence open source development to meet its own goals, and in other it altered the way that the project was governed. In response, the community employed different strategies to protect their commons-based resources.

In this chapter, I focus on the histories of three such projects: the OpenSolaris operating system, the MySQL relational database management system, and the OpenOffice productivity software that was designed as an alternative to Microsoft Office. Throughout the chapter, I focus on the ways that the FLOSS community maintains a unique ability to leverage its collective labor power against corporate encroachment into

its projects by using technical, legal, and governance strategies that allow them to abandon a project without losing the products of their labor. This has a similar effect to a factory walk out, whereby workers halt the productive process by abandoning the site of production. When dealing with software, however, production is not reliant on a particular space. Rather, productive activity can simply be moved to a new location. And, because of the unique legal institutions and technical features of open source software, a project can be "forked," whereby project can be copied and moved to a new location under a new name without violating the intellectual property protections of the original project. As we will see, this is one of the primary ways that the FLOSS community leverages its collective labor power against undue corporate influence.

To this end, I have structured the argument in the following ways. First, I provide some background about Oracle and Sun. Next, I discuss the histories of each of the three projects – OpenSolaris, MySQL, and OpenOffice – by focusing on their initial development, their acquisition by Sun, and their fates after Oracle acquired Sun in 2010. Finally, I conclude with some thoughts about why it will be important for the FLOSS community to maintain its ability to leverage its collective power.

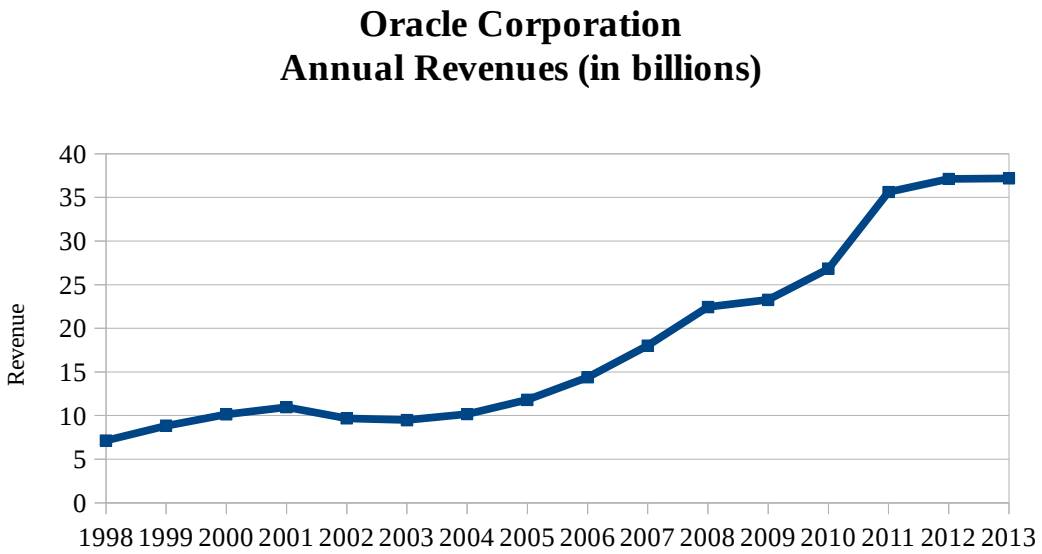
The Oracle Corporation and Sun Microsystems

The Oracle Corporation (hereafter simply Oracle) is one of the largest software companies in the world. The company has three main operating segments: software business, hardware business, and services.[32] In turn, these three segments are divided into seven smaller operating divisions: 1) new software licenses and cloud software

32 Unless otherwise noted, all of this information was derived from Oracle's annual filings (Form 10-K) with the Securities and Exchange Commission (SEC) of the United States, which is available here:http://www.oracle.com/us/corporate/investor-relations/financials/fy2013-form-10k-1966521.pdf (last accessed March 4, 2014)

subscription service; 2) software license updates and product support; 3) hardware systems products; 4) hardware systems support; 5) consulting services; 6) managed cloud services; and 7) education services. However, of the three main operating segments, Oracle earns nearly 75% of its total income from the software business segment. In 2013 alone, the company earned more than $37 billion in total revenues and employed approximately 120,000 people. Figure 6.1 provides an illustration of Oracle's total revenues from 1998-2013. If calculated by total revenues, Oracle is the third largest company in the global software market behind only IBM and Microsoft. Oracle has remained competitive within the global software market, in part, because of its strategic acquisitions. One of the company's largest acquisitions took place when it acquired Sun Microsystems in 2010.

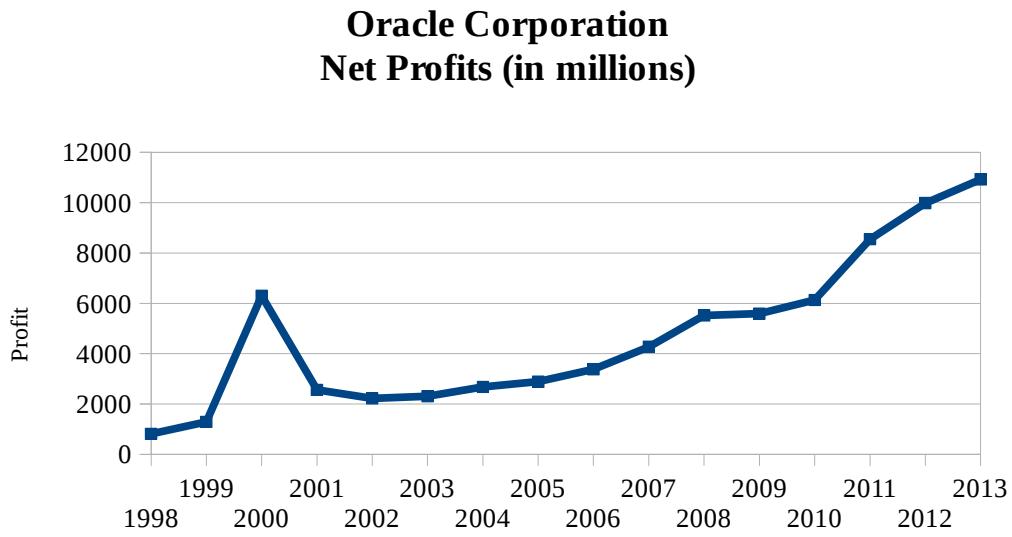
Figure 6.1. Oracle Corporation's Annual Revenues 1998-2013



Source: Data is from Oraclet's 10-K Annual Reports filed with the SEC for the years 1998-2013.

Figure 6.2 shows the rise in Oracle's net profits in the wake of its acquisition of Sun Microsystem. The company's net profits dipped in 2001 after the dot-com bubble burst, which had an effect on the entire economy in the U.S. at the time. However, Oracle has enjoyed a steady rise in profits since that time, with a noticeable spike in profits between 2010 and 2013. The company's profitablity can be directly tied to its acquisition of Sun Microsystems.

Figure 6.2. Oracle Corporation's Annual Net Profits 1998-2013



Source: Data is from Oraclet's 10-K Annual Reports filed with the SEC for the years 1998-2013.

Prior to its acquisition by Oracle in 2010, Sun Microsystems provided network computing infrastructure solutions, which included software, systems, storage, and microelectronics. In 2009, the final year of its independent operation, Sun reported approximately $11.45 billion in revenues and employed approximately 29,000 employees in more than 100 different countries. The majority of the company's revenues (42%) came from its Systems operating segment, which includes the sale of servers that provide computing and storage power to customers as a key part of Internet infrastructure. The

other core brands owned by Sun Microsystems were the Java technology platform, the Solaris Operating System, MySQL database management software, Sun StorageTek storage solutions and the UltraSPARC processor. Because the company relied on the provision of infrastructure-based services and products, the company was a large supporter of interoperability. Interoperability, here, is simply defined as the ability for different programs to exchange data with one another by using common formats. To facilitate innovation and interoperability, Sun made its key intellectual properties freely available as a way strategy to support open standards, open interfaces, and open source software. By making a commitment to open source, Sun was viewed favorably by the open source community and maintained a relatively good relationship with the community because it was transparent about its corporate goals. To better understand the reasons for Sun open-sourcing some of their key intellectual properties, we need to consider some of the historical development for corporate involvement in FLOSS projects.

A Brief History of the Market for Operating Systems

Throughout the 1980s, the market for operating systems was dominated by proprietary versions of Unix-based operating systems. For example, Hewlett Packard offered HPUX, IBM offered AIX, and Sun Microsystems offered SunOS. These operating systems dominated high computing, or infrastructural level computing, while the consumer market was dominated by Microsoft DOS, which was not based on Unix but developed entirely by Microsoft. Importantly, the proprietary Unix-based systems were *source-incompatible*. In effect, although these systems were all based on Unix, the development of separate proprietary versions had caused the code to diverge in such a

way that programmers could no longer assume interoperability between the systems. As a result, programmers had to maintain separate code bases for each system, and companies could sell entire stacks of software to their customers who had to accept the entire stack. This resulted in an inefficient system that was dominated by proprietary software vendors, while simultaneously increasing the workload for programmers. During the mid-1980s, however, the Free Software Foundation began as a response to the overly protective intellectual property restrictions placed on software. This, in turn, led to the development of free and open source software, which was collaboratively developed as a commons-based resource for others to study, use, adapt, or modify in any way.

Because this model of development was so successful, by the mid-1990s Linux, an open-source operating system, had become the dominant Unix-like operating system. Linux undercut the competition by offering a comparable product at a significantly lower cost. Furthermore, because Linux is licensed under the GNU General Public License (GPL), an alternative form of intellectual property ("copyleft"), improvements to Linux could be shared by everyone, which improved the quality and stability of Linux. The proprietary companies could not compete with Linux because the commons-based peer production driving Linux constituted a larger labor force than any of the individual companies could employ. Rather than competing directly with Linux, certain proprietary companies began to open source their products as a way of joining forces with the free and open source software community. Sun Microsystems was one of those companies.

Although Sun supported many different open source projects, I will focus on just three here. Sun open-sourced their Solaris operating system, which became OpenSolaris. They also open-sourced the MySQL database management software, as well as

StarOffice, which became OpenOffice. As I mentioned earlier, Sun maintained a good relationship with the broader FLOSS community because of their commitment to and support for FLOSS projects. After the company was acquired by Oracle, this relationship was strained in certain ways. In what follows, I will discuss how the developers working on the three projects mentioned above – OpenSolaris, MySQL, and OpenOffice – strategically resisted the corporate acquisition.

OpenSolaris

In 1987, Sun Microsystems and AT&T announced that they were going to merge some of the most popular Unix-based operating systems into a single project. This project eventually became Solaris, which was a proprietary operating system held by Sun that contained both open-source and closed-source components. To attract interest in the project and build a community of users and developers around the project, Sun Microsystems created OpenSolaris. OpenSolaris was an open-source version of the Solaris operating system, although OpenSolaris did contain some elements in the code that were not open source. After attracting a larger community of interest in the project, a Community Advisory Board (CAB) was created to direct the project. The CAB was comprised of two Sun employees, two members who were elected by the broader community, and one member who was appointed by Sun from the broader free software community. In effect, most of the CAB members were connected with or appointed by Sun, and Sun made clear what its intentions were for the OpenSolaris project.

Sun's strategy for the OpenSolaris project was to incorporate some of the developments from OpenSolaris into their proprietary Solaris operating system. In turn, Sun could sell the proprietary version of Solaris to other enterprises. The money earned

from sales of the Solaris project could then be used to support the developers and community involved in the OpenSolaris project. To facilitate this type of strategy, Sun protected OpenSolaris under a free software license created by the company called the Common Development and Distribution License (CDDL). This license enabled Sun to include proprietary, free software, or software protected under any other license in their Solaris and OpenSolaris operating systems. Consequently, Sun could use the OpenSolaris community as a way to drive development, quality control, or innovation that could be included in their proprietary Solaris offering. Importantly, however, Sun made this strategy very clear to the OpenSolaris community, and Sun was supportive of the broader FLOSS community, which gave it a good reputation within the community. Once they acquired Sun, Oracle took a very different approach to this strategy.

After Oracle acquired Sun, they announced plans to discontinue the regular distribution and development model of OpenSolaris (Laishram, 2010). Instead, Oracle would focus its development strategy on a new proprietary version of Solaris called Solaris Express. In effect, the new strategy from Oracle would not allow the community of developers that supported OpenSolaris to continue their work. In response, the Community Advisory Board directing the OpenSolaris project decided to fork the project. When a project is forked, developers take a copy of the source code for a particular project and begin to develop it as a distinct form of software. The resulting fork of the OpenSolaris project is called OpenIndiana, which was created to continue the development and distribution of the OpenSolaris project. Currently, Oracle still continues development on the proprietary Solaris Express operating system, while the community of developers supporting OpenSolaris have left Oracle to work on the forked version of OpenSolaris called OpenIndiana.

In the case of the OpenSolaris operating system, Oracle's strategy was simply to discontinue the open source project and focus development on a proprietary version of Solaris under the new name Solaris Express. This represents the most direct strategy for ending open development. Oracle announced that the open source project would be discontinued and, in response, the community had to fork the project to continue development under a new name. This is a similar fate to that of MySQL and OpenOffice, but Oracle's strategy for ending development took different forms in each case.

MySQL

In 2008, Sun Microsystems acquired MySQL AB for approximately $1 billion (MySQL, 2008). At the time, MySQL was growing in the market for relational database management software (RDBMS), and Sun's acquisition of MySQL would allow the company to compete directly with Oracle in that particular market. Only one year later, however, Oracle acquired Sun, and the MySQL property was one of the key properties that drew Oracle's interest. Indeed, the Sun-Oracle merger was originally approved by regulators in the United States, but the European Union (EU) did not immediately approve the deal specifically because of concerns that Oracle's acquisition of the MySQL property would lead to an anticompetitive market for RDBMS in Europe (Chapman & Newman, 2009). Consequently, the EU pressured Oracle to divest the MySQL property as a condition for approval of the merger. As leaked documents provided to the whistle-blowing site WikiLeaks have since shown, the United States Department of Justice communicated directly with the European Commission's Directorate General for Competition in support of the merger in October of 2009 (United States Mission to

European Union, 2009). Less than three months later, in December of 2009, the merger was approved without the divestiture conditions sought by the EU.

MySQL relied on a dual licensing approach that was similar to the licensing of OpenSolaris. The dual license model for MySQL would allow the code base for MySQL to be protected by the GNU GPL copyleft license, but proprietary versions could be created for enterprises that wanted customized installations. When the Sun-Oracle merger was approved, employees working for MySQL had reservations about Oracle's intentions for the GPL-protected code base of MySQL. Most notably among them was Michael "Monty" Widenius who authored the original version of MySQL and co-founded MySQL AB, which was the original owner of MySQL. Widenius later sold MySQL AB to Sun before Sun was acquired by Oracle. Widenius along with other MySQL developers were concerned that Oracle would try to discontinue MySQL or make it a closed-source program by using the same strategy it had with OpenSolaris. In response, Widenius urged MySQL users to "Help MySQL" by starting an online petition. Leading up to the acquisition of Sun, however, Oracle pledged to keep the same licensing strategies in place that had been negotiated with current customers for an additional five years (Whitney, 2009). That commitment is set to officially expire in December of 2014.

Fueled by the concerns about Oracle's intentions for MySQL, the developers forked the project to create MariaDB.[33] The code base for MariaDB is protected by the GNU GPL, and is designed to be a drop-in replacement for MySQL. As a forked project of MySQL, MariaDB allows its community of developers and users to ensure that the code will continue to be protected by the GNU GPL regardless of what Oracle decides to do with MySQL. Furthermore, although MySQL remains dominant in the RDBMS

33 MariaDB is just one fork of the MySQL project. Others include Drizzle and Percona Server.

market with an approximately 58% market share, MariaDB has now grown to claim approximately 18% of the market (Fydorenchyk, 2014). MariaDB has experienced increased growth in the database market in part because of some notable companies switching from MySQL to MariaDB, including Google and the Wikimedia Foundation.

MariaDB once again illustrates how the community of developers and users of an open source software can protected their projects from unwanted corporate encroachment. In the case of MariaDB, the project has gained additional attention from some of Oracle's competitors who have invested directly in the project. Most notably, SkySQL recently invested nearly $20 million to support the growth of MariaDB. Backed by capital from Intel and other venture capital firms, SkySQL is directed by some of the founding members of MySQL as well as former executives who left the company after Oracle acquired the project. SkySQL recently announced a merger with The Monty Program AB, which is led by Monty Widenius, the original author of MySQL. The merger reunites the original members of MySQL and transfers ownership of the MariaDB trademark to SkySQL. The resulting partnership will focus on developing MariaDB to compete with MySQL.

Furthermore, both the Monty Program AB and SkySQL belong to the MariaDB Foundation. The MariaDB Foundation is a non-stock, non-profit corporation, which was established to provide legal and technical support for the MariaDB project and provide a platform for supporters to contribute money to the project. For example, the MariaDB Foundation sells corporate memberships beginning at $50,000 and corporate sponsorships beginning at $5,000. According to the Foundation's web site, corporate memberships allow “engagement with the governance of the Foundation,” although no further details are provided about exactly what that entails (MariaDB Foundation, 2014).

In sum, MariaDB represents another example of how communities of FLOSS projects maintain the ability to protect their commons-based resource against unwanted corporate influence. In this case, however, Oracle's strategy was not to discontinue the open source project, *per se*. Rather, Oracle's acquisition of Sun would allow the company to gain a greater market share of the RDBMS market, and Sun's ownership of MySQL was one of the primary properties that attracted Oracle to acquire Sun. Although development of MySQL still continues under Oracle, many of the community members resigned from Sun, and Oracle's commitment to maintain the same licensing agreements for MySQL are set to expire at the end of 2014. To resist what could ultimately be a similar fate to that of OpenSolaris, the MySQL community forked the project to develop MariaDB. Furthermore, MariaDB has the additional benefit of having received investment capital from some of Oracle's competitors, which ensures the survival of the project for at least the foreseeable future. By establishing the MariaDB Foundation, the community has a legally recognizable organization to provide technical and legal support for the project, while also collecting additional donations to the project. In the third and final example provided in this paper, I focus on a series of office productivity software that eventually led to another forked project.

StarOffice, OpenOffice, LibreOffice

During the dot-com bubble in the mid- to late-1990s, Sun Microsystems experienced dramatic growth that allowed the company made some key acquisitions. In 1999, Sun acquired the German company, StarDivision, which developed StarOffice. StarOffice was designed as a proprietary office software featuring a word processing, spreadsheet, presentation, drawing, database, and formula programs. When Sun acquired

StarDivision, the company continued to develop StarOffice as a proprietary software. However, Sun forked the project and relicensed the software so that the source code could be made open source under a free and open source license. Once again, Sun's strategy was to use the newly open-sourced software, known as OpenOffice, to develop new features and fix bugs in the software. Then, the changes made to OpenOffice could be integrated into StarOffice, which contained certain proprietary elements. OpenOffice could continue to remain free to consumers, while Sun would try to monetize StarOffice by selling the software and services to customers who wanted the additional features. The upshot for Sun was the maintenance and support for essentially two different versions of the same software: OpenOffice 1.0 was a forked version of StarOffice 6.0, and Sun maintained the legal rights to both properties, although they were protected by different licenses.

The early versions of OpenOffice were protected by the Sun Industry Standards Source License (SISSL) and the GNU Lesser General Public License (GNU LGPL). Later versions were protected by an updated version of the LGPL after Sun discontinued the SISSL. The LGPL was chosen because it had less restrictive requirements for integrating free and open source software components into proprietary versions of the software. Although a full discussion of the distinctions between free and open source software licenses is beyond the scope of this essay, the basic differences between the GNU General Public License (GPL) and the GNU LGPL can be summarized quickly. The GPL requires that any modified or derivative software produced using a GPL-protected software as its base must be redistributed under the same licensing requirements. This ensures that free software remains free software rather than being exploited by commercial companies. The LGPL is a more permissive license that allows

free software elements to be incorporated into proprietary software. The only restriction on using LGPL-protected software is that the end-user must have the ability to modify the source code. By protecting OpenOffice in this way, Sun could ensure that developments in OpenOffice could be used in their proprietary StarOffice.

Thus, the symbiotic relationship between StarOffice and OpenOffice continued under Sun because Sun was transparent about what its intentions were for the two properties. Importantly, however, OpenOffice was governed by a Community Council comprised primarily of members from the broader OpenOffice community but also including a Sun employee as well. The Sun member on the Community Council was responsible for communicating Sun's intentions to the community. Once again, however, this relationship was strained when Oracle acquired Sun in 2010.

Since Oracle had discontinued the OpenSolaris operating system, members of the OpenOffice Community Council decided to create The Document Foundation and fork the OpenOffice project under the name LibreOffice until Oracle made its intentions clear for the OpenOffice project. Both The Document Foundation and LibreOffice were established with the intention of being temporary projects until Oracle made its intentions clear. In the event that Oracle ultimately decided to discontinue OpenOffice, however, the Community Council would be able to move development to the newly created LibreOffice. Furthermore, The Document Foundation was established as a non-profit organization to manage the LibreOffice project and promote the use of open-source document software more broadly. The initial governance of The Document Foundation was directed by a temporary steering council featuring some of the same members of the OpenOffice Community Council. Oracle viewed the Community Council members' positions on two governing boards as a conflict of interest and asked members on the

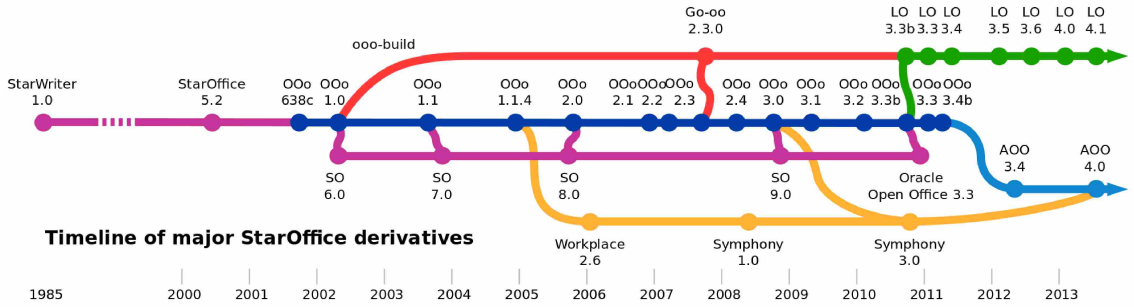
Community Council to step down from their positions (OpenOffice Community Council, 2010). This move effectively ended community support for OpenOffice and the project was renamed Oracle OpenOffice. Oracle OpenOffice became the proprietary software offering from Oracle that was meant to replace Sun's StarOffice.

While the official position of Oracle was to cite a conflict of interest, members of the broader open source community viewed Oracle's broader strategy as simply wanting to discontinue open source projects that existed under Sun because they did not provide any real value to the company. In response to this, however, The Document Foundation continued its development of LibreOffice. Since LibreOffice had strong community support, LibreOffice essentially surpassed OpenOffice within one release. In effect, all of the collective labor behind the development of OpenOffice abandoned the project but continued to work on LibreOffice. Because OpenOffice had been abandoned, Oracle announced that it would end development on the project entirely and fire the majority of OpenOffice developers. Ultimately, Oracle donated the code base for OpenOffice to The Apache Software Foundation, which has resumed development on the project under the name Apache OpenOffice.

To summarize this somewhat confusing history of a software that has been forked numerous times, Figure 1 illustrates the development history of StarOffice, its transition to OpenOffice (OOo) under Sun, the dual development of StarOffice (SO) alongside OpenOffice, the forks into LibreOffice (LO) and Oracle OpenOffice after Oracle acquired Sun in 2010, and the donation of OpenOffice back to The Apache Software Foundation to be developed as Apache OpenOffice (AOO). Figure 1 also includes additional forked projects that have not been discussed in this paper, which includes IBM Lotus Symphony (Symphony) and Go Open Office (Go-oo). As illustrated in the figure, the developments

from Go Open Office were ultimately included in LibreOffice and IBM's Symphony was ultimately included in Apache OpenOffice.

Figure 6.3. Development of StarOffice Derivatives



Source: David Gerard, 2013 under Creative Commons CC0 1.0 Universal Public Domain Dedication, available from http://en.wikipedia.org/wiki/File:StarOffice_major_derivatives.svg

In sum, the forking of OpenOffice into LibreOffice once again illustrates how members of the broader free and open source software community can protect the products of their collaborative labor against unwelcome influence from corporations. In the case of OpenOffice, Sun Microsystems maintained a good relationship with the community by supporting open-source development and being transparent about their intentions for the project. In addition, Sun was able to facilitate a symbiotic relationship between the open source OpenOffice and its proprietary StarOffice by using a licensing scheme that allowed them to integrate open source components into their proprietary software. Moreover, the OpenOffice project was governed by a Community Council, which included members from Sun who served as liaisons between Sun and the community. When Oracle acquired Sun, the Community Council was effectively disbanded after Oracle cited council members' involvement on LibreOffice steering committee as a conflict of interest. The OpenOffice and LibreOffice cases offer

examples of how the FLOSS community uses legal, technical, and governance strategies to protect their commons-based resources.

Protecting the Commons

Throughout this chapter, I have demonstrated how the FLOSS community maintains the ability to leverage its collective labor power against undue corporate influence by employing technical, legal, and governance strategies to protect its commons-based resources. On the one hand, FLOSS has unique technical characteristics that allow it to be reproduced and distributed widely without any significant cost. This allows FLOSS projects to be forked so that development can occur collaboratively, simultaneously, and continuously throughout the life of the project. Although dispersed development occurs, however, the community employs certain governance strategies for effectively coordinating development and protection of the project. These governance strategies include the establishment of non-profit organizations, which hold the intellectual properties for projects. These organizations provide a legally recognizable entity that can more effectively defend the intellectual property and licensing requirements of the project. Furthermore, more direct governance of the development project can occur through governing councils that are democratically elected or appointed by the community.

The legal strategies for defending FLOSS projects relies on alternative intellectual property protections like copyleft or other free and open source software licenses. These licenses free the software from overly protective copyright and allows the community to fork the project in the event of undue corporate influence. On the other hand, corporations can also use licensing strategies to their benefit as well. In the case of Sun,

the company used licensing that allowed for free and open source software development but that were less restrictive to the corporation. These licenses allowed the company to incorporate some of the commons-based peer production of FLOSS projects into their proprietary offerings. This strategy was understood and accepted by the FLOSS community because Sun was clear about its strategies but also because Sun supported FLOSS development projects. In a sense, then, licensing a project becomes a site of struggle, especially because a single project may contain code that is protected by different licenses. These licenses may have competing or conflicting terms that need to be resolved or the project becomes susceptible to intellectual property litigation. As was the case during Oracle's acquisition of Sun, the licenses can be changed as a way to direct development toward different ends. Sun was transparent about its licensing strategies as a part of its broader commercial strategies, while Oracle made either temporary commitments to use existing licensing strategies (i.e., MySQL) or sought to change those licensing requirements altogether (i.e., OpenSolaris).

However, the dynamics that exist between FLOSS communities and corporations are comprised of a combination of technical, legal, and governance strategies. The particular forms that these strategies take will vary depending on the individual project, but the FLOSS community's ability to defend its commons-based resources depends, in part, on a shared consciousness of what is permissible within the community. In a sense, this shared consciousness constitutes a sort of moral economy (Thompson, 1971). The FLOSS community leverages its collective labor power against corporate power by protecting its commons-based resources. When a corporation infringes on the moral economy of the community, the community rebels by forking the project and abandoning the project that has been overly influenced by the corporation. This moral economy has

foundations in the shared ideals of peer-to-peer relationship building, collaborative development, transparency, and community.

Even though the FLOSS community maintains the ability to leverage its power against undue corporate influence, community members are still in a somewhat precarious position as digital laborers. One definition of success in open source projects is to receive backing from a company, which at least ensures the project's survival if not its overall attractiveness. However, the FLOSS community depends on keeping projects protected under free software licenses, albeit of many different types, so that the community maintains the ability to keep the code for the program open. This is particularly true in cases where hybrid models of proprietary and free software are used in FLOSS projects. Throughout this paper, I have demonstrated how such struggles can occur, particularly after corporate mergers, acquisitions, or take overs.

In the face of growing corporate involvement in FLOSS projects, the broader FLOSS community must maintain its ability to protect its commons-based resources. At the same time, however, the protection of these resources depends, at least in part, on a shared collective understanding of how the community can leverage its collective labor power against increasing corporate involvement. The lessons to be learned from Oracle's acquisition of Sun Microsystems need to remain salient if similar strategies are to be effective. Most important, however, is the recognition that the struggles taking place within the FLOSS community are just one part of a broader social struggle. As Christian Fuchs (2008) has observed, commons-based production is not truly possible until we have a commons-based society. Until that time, commons-based movements like FLOSS will be subjected to increasing corporate encroachment that threatens to abate, assimilate, or altogether annihilate progress toward alternative economic configurations.

CHAPTER VII

CONCLUSION

Throughout this study, I have demonstrated the different ways that FLOSS projects have been incorporated into the corporate structures of various firms. CHAPTER II emphasized how a critical political economic perspective can be used to counteract some of the sweeping and, at times, overly celebratory treatments given to FLOSS communities in the theoretical literature. If we accept the claim that FLOSS as a process of production constitutes a form of commons-based peer production or non-market production that makes use of the knowledge commons (Benkler, 2006; Ostrom, 1990), then a critical political economic approach can both temper and complicate our understanding of these claims by emphasizing how these forms of production have been incorporated into larger corporate strategies. Each of the case studies discussed in this project have different implications for our understanding of commons-based peer production as a process, the knowledge commons as a resource, and FLOSS processes and products within the broader capitalist order.

In what follows, then, I discuss the major findings from each case study. Next, I explain how these novel findings can enrich our understanding of FLOSS products as commons-based resources and FLOSS processes as commons-based peer production. After establishing the major findings and their implications for our understanding of FLOSS products and processes, I discuss the limitations of the present study as well as areas that will be germane for future study. Finally, the chapter concludes with some final thoughts about the nature of the commons and commons-based peer production under capitalism.

Major Findings

The current study posed three primary research questions, which each sought to address certain characteristics of the dynamics existing between FLOSS communities and the corporations that are involved in FLOSS projects. Each of the questions was specific enough to address a core concern of the research project, while simultaneously broad enough to allow for careful attention to the complexity and diversity of different cases. The following section demonstrates how the case studies addressed these research questions.

Research Question #1

Research question #1 asked what is the relationship between proprietary, for-profit corporations and free and open source software communities, and how has this relationship changed over time?

The case studies demonstrated three different ways that corporations are involved in FLOSS projects. Consequently, the answer to this research question cannot be addressed without considering the contributions of each case study. The relationships between proprietary, for-profit corporations and FLOSS communities are diverse and do not always follow specific patterns. However, the cases of Red Hat and Microsoft most directly address this research question.

Red Hat, Inc.

In the case of Red Hat, which still maintains a relatively good relationship with the FLOSS community, the company was able to harness (which is to say, centralize) the collective labor power of the FLOSS community and transform it into a profitable

business strategy. Red Hat was created with the intention of providing a formalized institution that could bring the power of free software to the market. However, since the underlying source code for free software was protected by the Gnu General Public License (GPL), Red Hat was unable to rely on using copyright protection to exclude others from providing similar software and services. As a result, the company began offering customized versions of free software that could be packaged and protected under the Red Hat corporate logo. As such, the company's products could be protected by trademark. The software that the company provides, then, is protected by the Red Hat trademark, and the company sells customized subscriptions for its software and services. However, Red Hat still needed a way to protect its customers against potential intellectual property infringement claims. Consequently, the company needed a way to control the types of licenses allowed in its software offerings. To accomplish this, Red Hat first required all contributors to its software to sign a Individual Contributor License Agreement (ICLA), which would assign the rights to protect the code to the company. The ICLA later changed to the Fedora Project Contributor Agreement (FPCA), which served as a mechanism to control the range of possible licenses that could be included in contributions to its Fedora project. Nonetheless, the consequence of controlling the commons was the same.

From a particular point of view, then, Red Hat can be seen as a pragmatic solution to the problem of organizing commons-based peer production under capitalism. In effect, Red Hat has been able to establish itself as a trusted company that can accept liability for the products and services it provides. In effect, the problem of organizing commons-based peer production under capitalism was solved by establishing a legally recognizable and formal institution that serves as a mediator between corporations and the commons.

In doing so, however, Red Hat needed to find a way to control what types of code – or at least the types of intellectual property licenses – were included in its software so that it could protect itself and its clients against intellectual property infringement claims. In this sense, Red Hat functions as a *curator of the commons*. Just as a curator is responsible for collecting, organizing, and interpreting artifacts for the purpose of public display, Red Hat performs a similar function for its subscribers. In each case, the curator charges a fee to the public for entrance to a purposefully organized and constructed display of artifacts that has been interpreted in a particular way. The key difference, however, is that Red Hat does not rely on the collection of artifacts exactly as they existed previously. Rather, Red Hat relies on commons-based peer production from its FLOSS project, Fedora, for inclusion into its customized distributions of Red Hat Enterprise Linux. Moreover, the contributions to Fedora are controlled by worker agreements that all contributors to the Fedora project must sign. Importantly, however, because Red Hat is transparent about its intentions, the company has been able to enjoy a relatively good relationship with the broader FLOSS community throughout its history. This, of course, differs from the case of Microsoft.

Microsoft Corporation

Microsoft has a long history of opposition to FLOSS. This stance began as early as 1976 when Bill Gates authored the “Open Letter to Hobbyists,” in which he railed against the culture of sharing software within the community. He argued that this practice harmed the ability of others to produce software and be compensated for their work. However, this stance contradicts some of Microsoft's own history, as it relied on others' designs to produce some of its most successful software. This was particularly the

case for the MS-DOS operating system and the graphical user interface of Windows, which were built on top of previously existing technologies developed in Gary Kildall's CP/M operating system and Apple's graphical user interface. Both of these technologies were instrumental to Microsoft's success throughout the 1980s and 1990s, especially when paired with its strategic partnerships with IBM and other OEMs, which allowed the company to gain widespread adoption of its software. The same can be said of its Internet Explorer web browser, which the company packaged with distribution of its Windows operating system. This practice ensured that the company's web browser would win the first of the browser wars, but it also was one of the primary business practices that led to its conviction for antitrust violations by the Department of Justice.

Microsoft's ascent to the top of the personal computer software market culminated around the same time that it was being investigated for antitrust violations. When the DoJ issued its decree in 2001, Microsoft was forced to divest its operating system and applications operations. However, after the original District Court judge recused himself from the case after making some public comments that gave the impression of bias against Microsoft, the subsequent judge no longer sought divestment. Rather, Microsoft would need to agree to a series of consent decrees that were designed to prevent the type of predatory and uncompetitive behaviors that led to its conviction. The consent decrees were intended to last for five years, but they were renewed twice and finally came to an end in 2011. However, the decrees did little to affect Microsoft's economic performance, as the company's annual revenues and profits continued to climb in the wake of the DoJ's decision. Nevertheless, as argued in CHAPTER V, the antitrust suit marks a major historical moment both for Microsoft and the larger software industry. Most notably, the antitrust suit forced Microsoft to make its APIs more openly available to other developers

so they could design software that could interact with Microsoft's technologies. More generally, however, the antitrust decision coincides with the bursting of the dot-com bubble in 2001, the emergence of Linux as a commercially viable business model, and the emergence of the so-called Web 2.0 era, which shifted the business focus of many high-tech companies during that era.

If Microsoft needed any additional convincing that it could no longer rely on its old business model, the antitrust conviction signaled to Microsoft that the company needed to find new ways of doing business. Because Linux was becoming more widespread, Microsoft could no longer take an antagonistic stance toward open source. Instead, it needed to find ways to ensure that its products could function on devices that use Linux. To facilitate greater interoperability between Microsoft and non-Microsoft technologies, Microsoft expanded its Shared Source program and, in 2012, opened an entire division of the company dedicated to promoting and supporting open source, open standards, and open platforms. This shift is indicative of the fact that FLOSS, by many measures, has proven to be an effective and commercially viable production model. The shift in supporting open source projects suggests that Microsoft is trying to accomplish two primary goals: harnessing the power of commons-based peer production to supplement its own commercial goals as well as promoting interoperability between its technologies and other systems.

The Microsoft case study is indicative of a company undergoing a transformation in its stance to FLOSS. In part, this shift was driven by the antitrust conviction in 2001, but the leaked Halloween Documents suggest that the company was already concerned with the FLOSS phenomenon and how to combat it in 1998. Perhaps not coincidentally,

this is the same year that the antitrust investigation began. In this sense, Microsoft represents a strategy of *incorporating the commons*, or at least attempting to do so.

Research Question #1A

As a supplement to the first question, research question 1A asked about the power dynamics between corporations and the FLOSS community? In other words, which party holds the ability to exert influence on the other, if at all?

Oracle's Acquisition of Sun Microsystems

The third case study, Oracle's acquisition of Sun Microsystems, most directly addressed this question. That chapter illustrated how the FLOSS community has coped with undue corporate influence into its projects by focusing on three different FLOSS projects that were supported by Sun Microsystems prior to its acquisition by Oracle: the OpenSolaris operating system, the MySQL relational database management system, and the OpenOffice office productivity suite of software. What becomes clear from the case study is that FLOSS projects may not be able to avoid corporate influence altogether, especially when those projects are sponsored or supported by a particular company. However, given the nature of FLOSS code, the community maintains the ability to effectively abandon production on a particular FLOSS project by forking the project and continuing development under a new name. This is precisely what happened in each of the three cases discussed in Chapter VI.

Furthermore, the case study also provides evidence that FLOSS projects are not immune from the corporate maneuvering – acquisitions, integration, takeovers, buyouts, etc. – that is commonplace in a capitalist system. That is, although the projects may find

a corporation willing to provide support through sponsorship, financing, or partnerships, those relations can become strained in the wake of an acquisition in which the acquiring company is unwilling to provide the same level of support as the previous company. If this is the case, the community of developers who contribute to the FLOSS project have technical, legal, and governance strategies at their disposal to resist undue corporate influence in the project. Technically, code can be reproduced *ad infinitum* without any substantial reinvestment costs. Legally, most code that is used in FLOSS projects is protected by permissive licenses that allow the community to fork their project and begin development under a new name. Coinciding with the process of forking the project is the transitioning of the governing board members to oversee the new project.

The Oracle Corporation's acquisition of Sun Microsystems illustrates how the power dynamics existing between FLOSS communities and the corporations that rely on their projects are complex and varied. While the community still retains the power to abandon production on a project in the face of undue corporate influence, this still places the community in a precarious position with respect to the long-term survivability of their projects. The community retains the ability to fork the project and begin new development, but it cannot rely on the same level of support it received from its corporate sponsor unless it can find new investors. For instance, the OpenIndiana, MariaDB, and LibreOffice projects were able to find additional investment capital, although to varying degrees. In other words, the ability to fork a project is just one step in assuring productive autonomy. However, the productive autonomy of those who contribute to projects that are sponsored by other organizations may always be at risk of undue influence. In those situations, the community can take steps to try to reduce such influence.

Research Question #2

The second research question asked about value for each of these stakeholders. What value do corporations provide for the FLOSS community, and what value does the FLOSS community provide for corporations? Finally, do any external factors or stakeholders exist that may profit from this relationship?

In the case studies presented in this project, the value derived from FLOSS projects becomes quite clear. The value of all FLOSS projects comes from the software developers, programmers, and others who contribute time and labor to the project. Whether the developers perform the labor out of love for the project or work for a corporation that wants to support the project, the sheer number of contributors who focus on a particular project tends to be much larger than any single corporation could directly employ to work on developing a project. As such, the FLOSS community represents a large pool of collective laborers whose labor power is derived from the scale of their collective productive capacity. Because the FLOSS community features a potentially large pool of labor, the contributions of each individual, no matter how small, can be incorporated into FLOSS projects. These small, incremental changes can lead to the rapid completion of complex tasks when spread throughout an entire community of developers. Consequently, FLOSS projects tend to innovate more quickly, tend to be more secure, and tend to be competitive with their commercial counterparts. The literature discussed in CHAPTER II tends to focus on exactly these qualities of FLOSS projects and processes, but very few go as far as assigning the true value to the labor that makes such qualities possible.

The value that corporations hold for the FLOSS community is derived from their ability to provide support for FLOSS projects by funneling money or other resources into

a project, which is to say, a community. As mentioned previously, not all FLOSS projects require a direct corporate sponsor. However, some of the larger projects that do not have a direct corporate sponsor may still be governed by a non-profit entity that indirectly receives support from other firms or individual contributors. Direct sponsorship by a corporation may be another way to attract contributions to a project, as this ensures that it is likely to survive for awhile. Furthermore, this offers an avenue for contributors to signal their abilities to members within the project who may be working for a company that could provide employment to others who are looking for work.

Finally, the external stakeholders who exist in the relationships between FLOSS projects and the corporations who become involved in their projects are those who make use of the technology developed within the FLOSS community. Many of these technologies are used without much public awareness, like the Linux kernel, but others are used extensively and are highly recognizable, like the Mozilla Firefox web browser.

Contributions of the Study

This study makes a number of contributions to the existing scholarship in digital media studies and the political economy of communication, as well as our understanding of the commons and commons-based peer production under capitalism. As suggested in CHAPTER II, the internal dynamics of FLOSS communities and their models of production have been studied somewhat extensively. The individual motivations of contributors to FLOSS projects is diverse and varied (Deek & McHugh, 2008), but the FLOSS community as a whole generally believes in protecting the right to productive freedom (Coleman, 2013). The broader implications of the FLOSS community's practices have received their most notable theorizations in the work of Yochai Benkler

(2006), who used the term commons-based peer production to refer to the “radically decentralized, collaborative, and nonproprietary” forms of production that are based on “sharing resources and outputs among widely distributed, loosely connected individuals who cooperate with each other without relying on either market signals or managerial commands” (60). Benkler continues by arguing that this new modality of organizing production engenders greater degrees of freedom and democracy.

Benkler's assessment of commons-based peer production is largely celebratory in that he focuses on the institutional novelty of the arrangements and the capability of such production to facilitate high degree of innovation and entrepreneurship. However, this form of production still exists within a broader capitalist system. As such, capitalist firms have found a way to harness the entrepreneurship and innovation of some FLOSS communities and incorporate them into their broader corporate structures.

This study complicates and extends theorizations of commons-based peer production by investigating sites where the idealism of FLOSS production meets with the material realities of capitalism. These contested sites make up the case studies in this research project, for they are where commons-based peer production has been incorporated into the corporate structures of capitalist firms. By employing a critical political economic approach, this study focused on the power relations that exist between corporations that rely on capitalist, market-driven production, and the broader FLOSS communities that rely on non-market, commons-based peer production. An important part of this focus was to shift the discussion of the FLOSS community's innovativeness away from its instrumentality to business and couch its contributions in terms of collective labor and the collective labor power of the broader community. By focusing on the community's labor power, CHAPTER VI in particular was able to identify some of

the technical, legal, and governance strategies used within FLOSS communities to resist undue corporate influence.

Furthermore, the case studies provided the opportunity to investigate the unique ways that different corporations have incorporated the commons-based peer production of FLOSS communities. In previous literature, major projects like the Linux kernel or Wikipedia have been lauded as examples of effective and productive commons-based peer production can be (Benkler, 2006; Lessig, 2006; Weber, 2004). Significantly less studied, however, is how capitalist firms can use commons-based peer production to supplement their commercial offerings. The case studies for this project, particularly the discussion of Red Hat and Sun Microsystems, provided an in-depth look at how capitalist firms rely on the innovations and bug fixes from within the FLOSS community for implementation in their commercial products. That said, however, these case studies should not necessarily be viewed as generalizable across all FLOSS projects. The broader ecosystem of FLOSS projects features certain projects that are completely supported by its community of developers and do not rely on investment or sponsorship from corporate firms. Additional studies could continue to investigate the extent to which FLOSS projects rely upon or seek corporate sponsorship. Moreover, additional studies could investigate the extent to which sponsorship or capital investment is linked with the long-term survivability of a FLOSS project.

By selecting cases in which capitalist firms are incorporating commons-based peer production, this study was able to yield a novel insight into how intellectual property is used both within the FLOSS community and corporations. Specifically, the case of Red Hat demonstrated how a firm is able to profit off of intellectual property that is covered by the GPL and, therefore, not amenable to enclosure by traditional copyright.

Because Red Hat cannot exclude others from using its source code by relying on copyright, the company uses its trademarks to prohibit competitors from making a direct use of its products. However, Red Hat's trademarks cannot prevent someone from using the underlying source code, which is protected by copyleft. Indeed, this was the case of CentOS, which was designed as a functionally equivalent operating system to that offered by Red Hat Enterprise Linux, Red Hat's core commercial product. Similarly, Red Hat controls the types of licenses that can be included in its Fedora project, which is the FLOSS project that generates the code included in its commercial offerings. The ways in which Red Hat controls the intellectual property included in its commercial offerings complicates the claims made about the productive autonomy within FLOSS communities.

In the vast majority of work on FLOSS, one of the defining features of its novelty is often traced back to its protection under more permissive copyright licenses, or copyleft licenses (Benkler, 2006; Stallman, 2002; Lessig, 2001). In addition, the software industry has been broadly plagued by a surge in patent infringement claims. However, the issue of trademark is an often overlooked feature of software development, most likely because it has not been used as a traditional method for enforcing intellectual property protections. Red Hat uses trademark protections to circumvent the permissive nature of the GPL and the other licenses that do not allow it to claim exclusive ownership of the code used in its core products. To my knowledge, the extent to which other firms are using this strategy has yet to be investigated, particularly within the FLOSS community. Although Red Hat is just one example and, perhaps, an exceptional one, the case serves as a counter-factual example to the overarching claims made about the degrees of freedom, democracy, and autonomy within FLOSS production.

Further complicating these claims are the often-overlooked Contributor Licensing Agreements within FLOSS production, particularly when a project has a corporate or other institutional sponsor. While these agreements are not uniform across all FLOSS projects, the organizations that issue them rely on these agreements to maintain control over their projects. However, control is achieved in at least a couple different ways. The CLAs may ask contributors to surrender the rights to their submissions so that the organization can defend itself from intellectual property claims. Similarly, the CLAs may be used to control the types of licenses that are allowed into the code base. This was seen in the Red Hat case study, whereby Red Hat wanted guarantee its customers that they would not be in danger of intellectual property infringement suits. A common theme running throughout the Red Hat chapter was the extent to which copyright separates authorship from ownership. In this sense, the current project contributes to this critical understanding of copyright by demonstrating how FLOSS laborers are forced to abandon claims to ownership of their work in order to contribute directly to certain FLOSS projects. Further studies could investigate the differences between these agreements, which organizations are using them, and whether or not these agreements deter some contributors from becoming involved in projects.

Limitations of the Study

Despite these contributions, the present study was limited in certain ways. This is particularly the case with respect to the case study selections as well as the methodological choices. By choosing to operate from a critical political economic perspective, the study focused primarily on the power dynamics that exist between the FLOSS community and the corporations that rely on their labor. This directed attention

more toward corporate structures and strategies, as well as how these operations affected the FLOSS community, particularly how and why corporations were involved in FLOSS projects. In addition, the third case study, in particular, focused on the FLOSS community's response to such involvement.

The cases chosen were purposively selected because of their prominence within both corporate and FLOSS communities. Red Hat, Microsoft, and Oracle represent some of the largest and most publicly visible software companies in the world. This is primarily the reason for selecting these companies, but also means that the findings from each case study may not be applicable to a broader range of corporations or FLOSS projects. In this sense, the study can only provide a snapshot of some of the dynamics occurring at the intersection of corporations and the commons.

Furthermore, the study tended to concentrate more on the institutional arrangements between corporations and FLOSS communities. This was driven mainly by the theories drawn upon for the study. The intent was to demonstrate what happens if we accept the claims made by Benkler (2006) about commons-based peer production and non-market production and, in turn, contrast those claims with the dynamics existing at the intersection of corporations and communities of commons-based peer production. By taking this position, the study did not delve into the internal dynamics of different FLOSS communities. Indeed, one of the shortcomings was the constant reference to FLOSS communities, writ large, while each community has unique governance structures, unique relationships to its sponsoring organization (if it has one at all), and a unique culture. This is indicated by the point made about contributors to FLOSS projects and their support for FLOSS projects in general often being masked by their very particular preferences for certain software projects over others. For example, a contributor's

support for a particular project may be driven by the culture of the community developing the software or the proclaimed ideology of the project. On a related note, the FLOSS projects covered in this study all have (or had) corporate sponsors. However, this is not the case for all FLOSS projects. Consequently, future studies could do more to account for the diversity of FLOSS projects' goals, as well as the development community's internal dynamics. One notable example of this type of work is Gabriella Coleman's work on the Debian community (Coleman, 2013).

One final factor to consider in relation to the selection criteria for this study is the relatively recent opening of the Microsoft Open Technologies division. Because the subsidiary is still growing, the analysis was not able to offer a clear picture as to where the company is ultimately headed in its involvement in FLOSS projects. Throughout the course of the research, the publicly available information about the subsidiary changed extensively. The web page, for example, was continuously adding new information and organizing that information in new ways. Consequently, official press releases and secondary sources were used for the limited amount of information included about the newly formed subsidiary. As the MS Open Tech subsidiary grows and begins to develop more projects, we may be able to get a better sense of the exact types of projects that the company will be supporting.

Concluding Thoughts: **Capital and the Commons**

Commons-based peer production offers the potential to provide a truly novel form of organizing collective and collaborative production. However, the emergent or novel forms of organizing still exist within a broader capitalist order. Therefore, commons-based peer production should not be viewed as a comprehensive solution to the unequal

social relations of a capitalist system. Rather, commons-based peer production may be viewed as one part of a broader social struggle against global capital. More specifically, commons-based peer production can be viewed within the context of a broader resistance movement that seeks to reclaim commons of all types, whether they be tangible goods like land, water, and air, or the intangible goods of data, information, or knowledge that provide the infrastructure for social relations.

When Karl Polanyi authored *The Great Transformation*, he critiqued the then-emerging market fundamentalism of the Austrian School of economics, exemplified by Friedrich Hayek and inspired by the work of Ludwig von Mises, for its disembedding of market relations from social relations. For Polanyi, the market and market relations had historically been embedded within social relations, such that the social bonds connecting communities of people together were not subjected to a market logic. Rather, the market existed *within* and as a part of social relations. This, however, transformed after the market became elevated to a degree whereby all other relations became molded according to its logic. This disembedding of the market from social relations has the normative effect of creating certain "fictitious commodities," like land, labor, and money that had all previously been important infrastructural elements of social life. In other words, when land becomes a commodity, concerns about its long-term sustainability become subsumed under a market logic that seeks profit from its exploitation. The same applies to labor, which is to say, human beings, who become exploited and valued according to a market logic. Finally, money becomes something to be hoarded for its intrinsic or future value rather than its function as a universal equivalent for exchanging different goods.

Polanyi's critique could, perhaps, be expanded to include information as a fictitious commodity. This would offer a framework for situating information

dialectically between the market and social relations, as well as the increasing tendency to extract information out of its social function and treat it as a commodity. Indeed, Schiller (2007) draws this distinction between information as a commodity and information as a resource. When treated as a commodity and enclosed by intellectual property protections, information becomes highly valued as a privileged resource that can only be accessed by those who are willing to pay for access. When treated as a resource and made freely available for all, information can be studied, modified, adapted, and redistributed to others who can also benefit from access to it. Thus, we arrive at two conceptualizations of information: as a privately owned resource transformed into a commodity, and as a commonly held resource available for all.

Corporations, like Microsoft, have sought to transform information into a privately owned resource that can be protected by copyright. The FLOSS community has sought ways to preserve information as a commonly held resource for all to use, most notably through copyleft licenses like the GPL. By doing so, the community has been able to establish a knowledge commons that resists enclosure. However, the knowledge commons under capitalism may be facing a similar fate to the commons of the past, although with certain careful distinctions. This project has demonstrated that how capital has readjusted its relatively inflexible position in relation to commons-based production. It needed to reorient its strategies to *incorporate without enclosing* the commons. By doing so, capitalist firms pursue profits while finding a variety of ways to give back to the community, whether by making code freely available under free software or open source licensing, or by supporting the informal institutions that govern various open source projects. While this may provide *ad hoc* support for commons-based production, it may not provide a long-term solution to commons-based labor. Instead, commons-

based peer labor may be placed in an ever-more-precarious position of depressed or non-existent wages while corporations make commercial use of their contributions. What will be needed as this type of involvement continues is a sustainable way to protect the commons, but also a way to ensure investment in commons-based peer labor. In other words, not just investment in institutions, organizations, technologies, or innovations, but long-term and sustainable investment in the true source of their value, which is to say, people.

APPENDIX A

RECRUITMENT LETTER OR EMAIL

Dear [insert name],

My name is Ben Birkinbine, and I am a Ph.D Candidate from the School of Journalism and Communication at the University of Oregon. I am writing to invite you to participate in my research study about corporate involvement in open source projects. You're eligible to be in this study because of your involvement in such projects.

If you decide to participate in this study, you agree to be interviewed about your experiences, attitudes, beliefs, or opinions about corporate involvement in open source projects.

I would like to record audio of our interview. I plan to use this recording as a way to accurately represent your perspective on the research topic. However, you will have the option to not be recorded. You can indicate your preference on the Interview Consent Form.

Your participation in this study is completely voluntary. You can choose to be in the study or not. If you'd like to participate or have any questions about the study, please email or contact me at bjb@uoregon.edu or XXX-XXX-XXXX.

Thank you very much.

Sincerely,

Benjamin J. Birkinbine

Ph.d Candidate, Media Studies

School of Journalism & Communication

University of Oregon

APPENDIX B

INFORMED CONSENT LETTER

University of Oregon, School of Journalism & Communication
Informed Consent for Participation as a Subject in "Free Software and Capital"
Investigator: Benjamin J. Birkinbine
Type of consent *Adult Consent Form*

Introduction
You are being asked to be in a research study that investigates corporate participation in open source software projects. You were selected as a possible participant because of your involvement in open source software projects. Please read this form and ask any questions that you may have before agreeing to be in the study.

Purpose of Study:
The purpose of this study is to solicit participants' perceptions, opinions, beliefs, or other disclosures about practices associated with involvement in open source software. Participants in this study are either representatives from corporations involved in open source software projects or representatives from the broader open source community, whether they be programmers, advocates, members of non-profit groups, community organizations, or any other group involved in open source software development.

Description of the Study Procedures:
If you agree to be in this study, I will ask you to do the following: agree to a semi-structured interview in which I will be asking you for disclosures about your experiences, attitudes, opinions, beliefs, or other feelings associated with free and open source software. Most interviews will last anywhere from 30-60 minutes, but certain interviews may last for a longer or shorter duration.

Risks/Discomforts of Being in the Study:
There are no reasonable foreseeable (or expected) risks. This study may include risks that are unknown at this time.

Benefits of Being in the Study:
The purpose of the study is to gain a better understanding of the relationship between for-profit corporations and the broader free and open source software community. By participating in this study, you are contributing to this understanding and have a chance for your voice to be heard.

Payments and Costs:
You will not be receiving any payment for participating in this study. There are no costs to you for participating in this research study.

Confidentiality:
The records of this study will be kept private. In the final published report, I would like to be able to identify you as well as your affiliations unless you request to remain

anonymous. You will have the opportunity to indicate your preference at the end of this form. If you choose to remain anonymous, I will not include any information that will make it possible to identify you. Research records will be kept in a locked file.

I would also like to keep an audio recording of our interview. If you consent to being recorded, the digital audio files will be kept on a password protected personal computer and destroyed after the final written report is published. You will have the opportunity to indicate your preference for being recorded at the end of this form.

Access to the records will be limited to the researchers; however, please note that the Institutional Review Board and internal University of Oregon auditors may review the research records.

Voluntary Participation/Withdrawal:
Your participation is voluntary. If you choose not to participate, it will not affect your current or future relations with the University. You are free to withdraw at any time, for whatever reason.
There is no penalty or loss of benefits for not taking part or for stopping your participation.

Contacts and Questions:
The researcher conducting this study is Benjamin J. Birkinbine. For questions or more information concerning this research you may contact him at XXX-XXX-XXXX or bjb@uoregon.edu
If you have any questions about your rights as a research subject, you may contact: Research Compliance Services, University of Oregon at (541) 346-2510 or ResearchCompliance@uoregon.edu

Copy of Consent Form:
You will be given a copy of this form to keep for your records and future reference.

Statement of Consent:
I have read (or have had read to me) the contents of this consent form and have been encouraged to ask questions. I have received answers to my questions. I give my consent to participate in this study. I have received (or will receive) a copy of this form.

Participant Preferences (please mark one for each preference):

☐ I agree to have audio from the interview recorded, **OR** ☐ I DO NOT agree to be recorded.

☐ I agree to be identified by name and affiliation, **OR** ☐ I would like to remain anonymous.

Signatures/Dates

_____

Study Participant (Print Name)

_____

Signature **Date**

APPENDIX C

INTERVIEW GUIDE

Interviews will be semi-structured. Questions provided here serve as a base from which additional follow-up questions may be asked.

General Questions (for all participants):

How are you involved with free and open source software?

How long have you served in that role?

Do you know how to program or code using free and open source software?

Do you currently contribute to developing free and open source software? If so, which project?

What are your thoughts about the relationship between corporations and the open source community?

More broadly, do you think cooperation has a place in a competitive economy?

Do you think the relationship between corporations and the open source community has changed over time?

Questions Specifically For Participants Representing Corporations:

Why is your company supporting free and open source software projects?

How are these projects licensed?

How long has your company been contributing to or supporting open source software projects?

Do you currently employ people specifically responsible for open source projects?

How does your company benefit from open source projects?
Do open source projects tend to be profitable? If so, how?

How do you measure an open source project's benefits to your company?

Do you have any data about your company's open source projects?

Do you know how many people contribute to your projects from outside the company?

Does your company plan to continue supporting open source projects?

Are there any risks to your company by becoming involved in open source projects?

REFERENCES CITED

Andrejevic, M. (2012). Exploitation in the data mine. In Christian Fuchs, Kees Boersma, Albrechtslund and Sandoval (eds.), *Internet and surveillance: The challenges of Web 2.0 and social media*. New York, NY: Routledge, pp. 71-88.

Andrejevic, M. (2007). *ISpy: Surveillance and power in the interactive era*. Lawrence, KS: University Press of Kansas.

Apple Computer, Inc. v. Microsoft Corporation, 35 F.3d 1435 (1994)

Bagdikian, B. (2004). *The new media monopoly*. Boston, MA: Beacon Press.

Bauwens, M. (2013). Thesis on digital labor in an emerging P2P economy. In Scholz, T. (ed.). (2013). *Digital labor: The Internet as playground and factory*. New York, NY: Routledge, pp. 211-224.

BBC News, Inc. (2004, August 26). Microsoft Linux ad 'misleading.' *BBC News*. Last accessed May 22, 2014 from http://news.bbc.co.uk/2/hi/technology/3600724.stm

Benkler, Y. (2006). *The wealth of networks: How social production transforms markets and freedom*. New Haven, CT: Yale University Press.

Berners-Lee, T., & Caillau, R. (1990, November 12). WorldWideWeb: Proposal for a hypertext project. Last accessed May 20, 2014 from http://www.w3.org/Proposal.html

Bettig, R. (2009). Private equity, private media. *Democratic Communiqué*, 23(1), pp. 22-44.

Bettig, R. (1992). Critical perspectives on the history and philosophy of copyright. *Critical Studies in Mass Communication,* 9(2), pp. 131-155.

Boyle, J. (2008). *The public domain: Enclosing the commons of the mind*. New Haven, CT: Yale University Press.

Boyle, J. (2003). The Second Enclosure Movement and the construction of the public domain. *Law and Contemporary Problems*, 66, pp. 33-74. Last accessed May 26, 2014 from http://scholarship.law.duke.edu/cgi/viewcontent.cgi?article=1273&context=lcp

Braverman, H. (1974). *Labor and monopoly capital: The degradation of work in the twentieth century.* New York, NY: Monthly Review Press.

Campbell-Kelly, M. (2001). Not only Microsoft: The maturing of the personal computer software industry, 1982-1995. *The Business History Review*, 75(1), pp. 103-145. Last accessed May 14, 2014 from http://www.jstor.org/stable/3116558

Cassidy, J. (2002). *Dot.con: The greatest story ever sold*. New York, NY: HarperCollins.

Chan, S.P. (2011, May 11). Long antitrust saga ends for Microsoft. *The Seattle Times*. Last accessed http://seattletimes.com/html/microsoft/2015029604_microsoft12.html

Chapman, P., & Newman, M. (2009, September 3). Oracle faces in-depth EU probe over Sun purchase (update 2). *Bloomberg.* Last accessed August 4, 2014 from http://www.bloomberg.com/apps/news?pid=newsarchive&sid=aCWYuHl5bHC8

Computing History Project. (2008). Ada Lovelace - The Babbage Engine. *Computing History Project*. Last accessed August 1, 2014 from http://www.computerhistory.org/babbage/adalovelace/

Coleman, G. (2013). *Coding freedom: The ethics and aesthetics of hacking*. Princeton, NJ: Princeton University Press.

Copeland, B.J. (2006). The modern history of computing. *Stanford Encylopedia of Philosophy*. Last accessed August 2, 2014 from http://plato.stanford.edu/entries/computing-history/

Cusumano, M.A., & Yoffie, D.B. (1998). *Competing on Internet time: Lessons from Netscape and its battle with Microsoft*. New York, NY: The Free Press.

Danielian, N. R. (1939). *A.T. & T: The story of industrial conquest*. New York, NY: The Vanguard press.

Deek, F.P., & McHugh, J.A.M. (2008). *Open source: Technology and policy*. New York, NY: Cambridge University Press.

Deleris, B. (2006, December 1). Battling bugs: Embedded debugging tactics. *EDN*. Last accessed August 2, 2014 from http://edn.com/electronics-news/4317260/Battling-bugs-embedded-debugging-tactics

Elstrom, P. (1997, January 22). Microsoft's $8 million goodbye to Spyglass. *Businessweek.com*. Last accessed May 20, 2014 from http://www.businessweek.com/bwdaily/dnflash/january/new0122d.htm

Fairclough, N. (2001). The discourse of new labour: Critical discourse analysis. In Simeon Yates, Stephanie Taylor, and Margaret Wetherell (eds.), *Data as Discourse: A Guide for Analysis,* pp. 229-245 Thousand Oaks, CA: SAGE.

Festa, P. (2001). Governments push open-source software. *Cnet News*. Last accessed May 30, 2014 from http://news.cnet.com/2100-1001_3-272299.html

Fitzpatrick, M. (2012, August 10). “What is the Syrian Electronic Army?” *Mashable.com*. Last accessed February 1, 2013 from http://mashable.com/2012/08/10/syrian-electronic-army/

Fogel, K. (2005). *Producing open source software: How to run a successful free software project.* Sebastopol, CA: O'Reilly Media.

Free Software Foundation, Inc. (2012). The free software definition. Last accessed January 22, 2013 from https://www.gnu.org/philosophy/free-sw.html

Frischmann, B. M. (2012). *Infrastructure: The social value of shared resources*. New York, NY: Oxford Univesity Press.

Fuchs, C. (2013). Class and exploitation on the Internet. In Scholz, T. (ed.). (2013). *Digital labor: The Internet as playground and factory*. New York, NY: Routledge, pp. 211-224.

Fuchs, C. (2012). Critique of the political economy of web 2.0 surveillance. In Christian Fuchs, Boersma, Anders Albrechtslund and Marisol Sandoval (eds.), *Internet and surveillance: The challenges of Web 2.0 and social media*. New York, NY: Routledge, pp. 31-70.

Fuchs, C. (2011a, February 01). New Media, Web 2.0 and Surveillance. *Sociology Compass, 5(*2), 134-147.

Fuchs, C. (2011b). Web 2.0, prosumption, and surveillance. *Surveillance & Society* 8(3), pp. 288-309. Last accessed May 28, 2014 from http://library.queensu.ca/ojs/index.php/surveillance-and-society/article/view/4165

Fuchs, C. (2008). *Internet and society: Social theory in the information age.* New York, NY: Routledge.

Fydorenchyk, T. (2014, February 6). Software stacks market share: January 2014. *Jelastic*. Last accessed August 4, 2014 from http://blog.jelastic.com/2014/02/06/software-stacks-market-share-january-2014/

Gallagher, S. (2013, October 18). The navy's newest warship is powered by Linux. *Ars Technica.* Last accessed on May 30, 2014 from http://arstechnica.com/information-technology/2013/10/the-navys-newest-warship-is-powered-by-linux/

Garland, H. (1977). Design innovations in personal computers. *Computer*, 10(3), pp. 24-27. doi:10.1109/C-M.1977.217669

Gates, B. (1976). An open letter to hobbyists. In Robert Reiling (ed.), *Homebrew Computer Club Newsletter,* 2(1), pp. 2. Mountain View, CA: Hombrew Computer Club. Available from *Digibarn Computer Museum*, last accessed May 13, 2014 from http://www.digibarn.com/collections/newsletters/homebrew/V2_01/index.html

Gilbert, R.J. (1995). Networks, standards, and the use of market dominance: Microsoft. In John E. Kwoka Jr. and Lawrence J. White (eds.) (2004), *The Antitrust Revolution: Economics, Competition, and Policy.* New York, NY: Oxford University Press, pp. 409-429.

Gleick, J. (2011). *The information: A history, a theory, a flood.* New York, NY: Pantheon Books.

Gramsci, A. (1971). *Selections from the prison notebooks*. New York, NY: International Publishers.

Greene, T.C. (2001). Ballmer: "Linux is a Cancer." *The Register.* Last accessed March 8, 2013 from http://www.theregister.co.uk/2001/06/02/ballmer_linux_is_a_cancer/

Hamm, S. and Greene, J. (2004). The man who could have been Bill Gates. *Bloomberg Businessweek Magazine.* Last accessed May 14, 2014 from http://www.businessweek.com/stories/2004-10-24/the-man-who-could-have-been-bill-gates

Hardin, G. (1968). The Tragedy of the Commons. *Science*, 162 (3859), pp. 1243-1248.

Hardt, M., & Negri, A. (2005). *Multitude: War and democracy in the age of empire*. New York, NY: Penguin Books.

Hardt, M., & Negri, A. (2000). *Empire*. Cambridge, MA: Harvard University Press.

Harmon, A., & Markoff, J. (1998, November 3). Internal memo shows Microsoft executives' concern over free software. *The New York Times*. Last accessed May 21, 2014 from http://www.nytimes.com/library/tech/98/11/biztech/articles/03memo.html

Hayes, M. (1976, February 20). Regarding your letter of 3 February 1976 appearing in Homebrew Computer Club Newsletter vol. 2 no. 1. In Robert Reiling (ed.), *Homebrew Computer Club Newsletter,* 2(2), pp. 2. Mountain View, CA: Hombrew Computer Club. Available from *Digibarn Computer Museum*, last accessed May 13, 2014 from http://www.digibarn.com/collections/newsletters/homebrew/V2_02/homebrew_V2_02_p2.jpg

Hess, C., & Ostrom, E. (2007). Introduction: An overview of the knowledge commons. In Charlotte Hess & Elinor Ostrom (eds.), *Understanding knowledge as a commons: From theory to practice*. Cambridge, MA: MIT Press, pp. 3-26.

Jenkins, H. (2006). *Convergence culture: Where old and new media collide*. New York, NY: New York University Press.

Jher, & Birkinbine, B. (2012, March 19). Ecological media: Technology and the environment. Paper presented at the Inaugural History and Theory of New Media Unconference, held in the McKenzie Hall Collaboration Center, University of Oregon. Eugene, OR.

Kanellos, M., & Shankland, S. (1999, September 8). Red Hat stock surge creates billionaires. *CNet*. Last accessed April 23, 2014 from http://news.cnet.com/Red-Hat-stock-surge-creates-billionaires/2100-1001_3-205557.html

Kaste, M. (2004, September 15). Brazil switches from Microsoft to 'open source' software. *National Public Radio*. Last accessed May 30, 2014 from http://www.npr.org/templates/story/story.php?storyId=3919175

Kingstone, S. (2005). Brazil adopts open source software. *BBC News*. Last accessed May 30, 2014 from http://news.bbc.co.uk/2/hi/4602325.stm
Lai, E. (2007, October 29). Microsoft and open-source backers eye each other – warily. *ComputerWorld*.

Laishram, R. (2010, August 14). Oracle has killed OpenSolaris. *TechieBuzz*. Last accessed August 2, 2014 from http://techie-buzz.com/foss/oracle-has-killed-opensolaris.html

Lakhani, K.R., & Wolf, R.G. (2005). Why hackers do what they do: Understanding motivation and effort in free/open source software projects. In Joseph Feller, Brian Fitzgerald, Scott A. Hissam, and Karim R. Lakhani (eds.), *Perspectives on free and open source software*. Cambridge, MA: MIT Press.

Lazzarato, M. (1996). Immaterial labor. In Paul Virno & Michael Hardt (eds.), *Radical thought in Italy: A potential politics*. Minneapolis, MN: University of Minnesota Press.

Lessig, L. (2006). *Code: Version 2.0*. New York: Basic Books.

Lessig, L. (2001). *The future of ideas: The fate of the commons in a connected world*. New York, NY: Random House.

Levy, S. (1984). *Hackers: Heroes of the computer revolution*. Garden City, N.Y: Anchor Press/Doubleday.

Machlup, F. (1962). *The production and distribution of knowledge in the United States*. Princeton, NJ: Princeton University Press.

MariaDB Foundation. (2014). About the MariaDB Foundation. Last accessed August 2, 2014 from http://mariadb.org/en/foundation

Marx, K. (1906). *Capital, a critique of political economy*. New York, NY: Modern Library.

Marx, K. & Engels, F. (1845). *The German ideology.* Amherst, NY: Prometheus Books.

Maxwell, R. (2003). *Herbert Schiller*. Lanham, MD: Rowman & Littlefield.

McKercher, C., & Mosco, V. (2007) (eds.). *Knowledge workers in the information society*. Lanham, MD: Lexington Books.

Meehan, E.R. (2005). *Why TV is not our fault: Television programming, viewers, and who's really in control*. Lanham, MD: Rowman & Littlefield Publishers, Inc.

Meehan, E.R. (1999). Commodity, culture, common sense: Media research and paradigm dialogue. *Journal of Media Economics*, 12(2), pp. 149-163.

Meehan, E.R., Mosco, V., & Wasko, J. (1993). Rethinking political economy: Continuity and change. *Journal of Communication*, 43(4), pp. 347-358.

Microsoft Open Technologies. (2014a). Company web site – main page. Last accessed June 4, 2014 from http://msopentech.com/

Microsoft Open Technologies. (2014b). What we do. Last accessed June 4, 2014 from http://msopentech.com/what-we-do/

Moody, G. (2001). *Rebel code: The inside story of Linux and the open source revolution*. Cambridge, MA: Perseus Publishing.

Mosco, V. (2009). *The political economy of communication*. London: SAGE.

Mosco, V. (2006). Knowledge and media workers in the global economy: Antimonies of outsourcing. *Social Identities, 12,* 6, 771-790.

Mundie, C. (2001, May 3). Speech transcript – Craig Mundie, The New York University Stern School of Business. *Microsoft.com*. Last accessed June 2, 2014 from http://www.microsoft.com/en-us/news/exec/craig/05-03sharedsource.aspx?navIndex=2

Murdock, G. & Golding, P. (1973). For a political economy of mass communications. *Socialist Register*, 10, 205-234.

MySQL. (2008). Sun to acquire MySQL. Press Release. Last accessed August 4, 2014 from http://www.mysql.com/news-and-events/sun-to-acquire-mysql.html

Nairn, A. (2002). *Engines that move markets: Technology investing from railroads to the internet and beyond.* New York, NY: John Wiley & Sons.

Neeson, J.M. (1993). *Commoners: Common right, enclosure and social change in England, 1700-1820*. New York, NY: Cambridge University Press.

Neilson, B., & Rossiter, N. (January 01, 2008). Precarity as a Political Concept, or, Fordism as Exception. *Theory, Culture and Society, 25*, 7-8.

O'Reilly, T. (2005). What is Web 2.0: Design patterns and business models for the next generation of software. *O'Reilly.com*. Last accessed May 28, 2014 from http://oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=all

OpenOffice Community Council. (2010, October 14). Community council log 201014. Last accessed August 4, 2014 from http://wiki.openoffice.org/wiki/Community_Council_Log_20101014

Open Source Ecology. (2014). Web site. Last accessed August 5, 2014 from http://opensourceecology.org/

Open Source Initiative. (2007, October 12). OSI approves Microsoft license submissions. *Opensource.org*. Last accessed June 2, 2014 from http://opensource.org/node/207

Oracle Corporation. (2013). Annual report: Form 10-k. Last accessed August 2, 2014 from http://oracle.q4cdn.com/e8a91886-38c1-427e-a6c5-a32536c1fa4e.pdf

Ostrom, E. (2005). *Understanding institutional diversity*. Princeton: Princeton University Press.

Ostrom, E. (1990). *Governing the commons: The evolution of institutions for collective action*. Cambridge: Cambridge University Press.

Page, W.H., & Lopatka, J.E. (2007). *The Microsoft case: Antitrust, high technology, and consumer welfare*. Chicago, IL: The University of Chicago Press.

Pang, A.S. & Marinaccio, W. (2000). The Xerox PARC visit. Included as part of the *Making the Macintosh: Technology and Culture in Silicon Valley* project, an ongoing project available online. Last accessed May 15, 2014 from http://www-sul.stanford.edu/mac/parc.html

Polanyi, K. (2001). *The great transformation: The political and economic origins of our time*. Boston, MA: Beacon Press.

Pollack, A. (1990, March 24). Most of Xerox's suit against Apple barred. *The New York Times*. Last accessed May 15, 2014 from http://www.nytimes.com/1990/03/24/business/most-of-xerox-s-suit-against-apple-barred.html

Rahemipour, J. (2010, October 31). [native-lang] Every end is a new beginning. Retrieved from The Mail Archive, last accessed March 6, 2013 from http://www.mail-archive.com/dev@native-lang.openoffice.org/msg04865.html

Raymond, E.S. (2004). Halloween X: Follow the money. *The Halloween Documents*. Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween10.html

Raymond, E.S. (2002a). Halloween VII: Survey says. *The Halloween Documents.* Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween7.html

Raymond, E.S. (2002b). Halloween VIII: Doing the damage-control dance. *The Halloween Documents*. Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween8.html

Raymond, E.S. (2000). *The cathedral and the bazaar* [online version]. Last accessed January 23, 2013 from http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html

Raymond, E.S. (1998a). Open source software: A (new?) development methodology. *The Halloween Documents: Halloween Document I (Version 1.17)*. Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween1.html

Raymond, E.S. (1998b). Linux OS competitive analysis: The next Java VM? *The Halloween Documents: Halloween Document II (Version 1.7)*. Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween2.html

Raymond, E.S. (1998c). Microsoft's reaction to the 'Halloween Memorandum.' *The Halloween Documents: Halloween Document III (Version 1.6)*. Last accessed May 31, 2014 from http://www.catb.org/esr/halloween/halloween3.html

Red Hat, Inc. (2014). The Fedora Project wiki. Last accessed May 9, 2014 from http://fedoraproject.org/wiki/Fedora_Project_Wiki.

Red Hat, Inc. (2013). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2012). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2011). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2010). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2009). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2008). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2007). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2006a). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2006b). *Red Hat trademark guidelines.* Last accessed May 1, 2014 from http://www.redhat.com/f/pdf/corp/RH-3573_284204_TM_Gd.pdf

Red Hat, Inc. (2005). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2004). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2003). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2002). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2001). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Red Hat, Inc. (2000). Form 10-K. *Annual report*. Last accessed May 1, 2014 from http://investors.redhat.com/sec.cfm?DocType=Annual&Year=&FormatFilter=

Reimer, J. (2005, December 14). Total share: 30 years of personal computer market share. *Ars Technica*. Last accessed May 19, 2014 from http://arstechnica.com/features/2005/12/total-share/6/

Rossiter, N., & Zehle, S. (2013). Acts of translation: Organized networks as algorithmic technologies of the common. In Trebor Scholz (ed.), *Digital labor: The Internet as playground and factory*. New York, NY: Routledge, pp. 225-239.

Salmon, F. & Stokes, J. (2010, December 27). Algorithms take control of Wall Street. *Wired.* Last accessed March 5, 2014 from http://www.wired.com/magazine/2010/12/ff_ai_flashtrading/all/

Saunders, M., & Morrison, G. (2014). The big switch: How Munich switched 15,000 PCs from Windows to Linux. *Linux Voice*. Last accessed May 30, 2014 from http://www.linuxvoice.com/the-big-switch/

Sayers, S. (2007). The concept of labor: Marx and his critics. *Science & Society,* 71(4), pp. 431-454.

Schiller, D. (2007). *How to think about information*. Urbana, IL: University of Illinois Press.

Schiller, D. (1999). *Digital capitalism: Networking the global market system*. Cambridge, MA: MIT Press.

Scholz, T. (ed.). (2013). *Digital labor: The Internet as playground and factory*. New York, NY: Routledge.

Scott, J. (1990). *A matter of record.* Cambridge, MA: Polity Press.

Smith, M. (2011). *The secrets of Station X: How the Bletchley Park codebreakers helped win the war.* London: Biteback Publishers.

Smythe, D.W. (1981). *Dependency road: Communications, capitalism, consciousness, and Canada*. Norwood, NJ: Ablex Publishing Corp.

Smythe, D.W. (1960). On the political economy of communications. *Journalism & Mass Communication Quarterly,* 37(4), pp.563-572.

Söderberg, J. (2008). *Hacking capitalism: The free and open source software movement*. New York, NY: Routledge.

Stallman, R. (2012). Why open source misses the point of free software. Last retrieved January 22, 2013 from https://www.gnu.org/philosophy/open-source-misses-the-point.html

Stallman, R.M. (2002). *Free software, free society: Selected essays of Richard M. Stallman.* Boston, MA: GNU Press.

Streeter, T. (2011). *The net effect: Romanticism, capitalism, and the internet*. New York, NY: New York University Press.

Taibbi, M. (2013, April 25). Everything is rigged: The biggest price-fixing scandal ever. *Rolling Stone*. Last accessed March 5, 2014 from http://www.rollingstone.com/politics/news/everything-is-rigged-the-biggest-financial-scandal-yet-20130425

Tapscott, D., & Williams, A. D. (2006). *Wikinomics: How mass collaboration changes everything*. New York, NY: Portfolio.

TechInsider.org. (2013). Joint development agreement between IBM and Microsoft. Last accessed July 3, 2014 from http://tech-insider.org/os2/microsoft.html

Terranova, T. (2004). *Network culture: Politics for the information age*. Ann Arbor, MI: Pluto Press.

Terranova, T. (2000). Free labor: Producing culture for the digital economy. *Social Text*, 63, 18(2), pp. 33-58.

The History of Computing Project. (2014). Microsoft company 15 September 1975. Last accessed May 15, 2014 from http://www.thocp.net/companies/microsoft/microsoft_company.htm

The Linux Foundation. (2013). *Linux kernel development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it*. Last accessed August 4, 2014 from http://www.linuxfoundation.org/publications/linux-foundation/who-writes-linux-2013

The Linux Foundation. (2012). *Linux kernel development: How fast it is going, who is doing it, what they are doing, and who is sponsoring it*. Last accessed January 24, 2013 from http://www.linuxfoundation.org/publications/linux-foundation

Thompson, E.P. (1971). The moral economy of the English crowd in the eighteenth century. *Past and Present*, 50, pp. 76-136.

Thompson, E.P. (1966). *The making of the English working class*. New York, NY: Vintage Books.

Tiemann, M. (2007). Who is behind 'Shared Source' misinformation campaign? *Open Source Initiative.* Last accessed May 28, 2014 from http://opensource.org/node/225

Top500.org. (2014). Operating system family/Linux. *Top500.org*. Last accessed May 30, 2014 from http://www.top500.org/statistics/details/osfam/1#.U4i_InKfoxA

Tramontano, M., & Trevisan, N. (2003). A dimensão digital de Solonópole, Brasil. *SIGraDi: Proceedings from the 7th Iberoamerican Congress of Digital Graphics*, pp. 74-77. Rosario, Argentina. Last accessed May 30, 2014 from http://cumincades.scix.net/data/works/att/sigradi2003_060.content.pdf

Tu, J.I. (2012, December 7). Goldman Sachs: Microsoft has gone from 97 percent market share of compute [sic] market to 20 percent. *The Seattle Times*. Last accessed May 12, 2014 from http://seattletimes.com/html/microsoftpri0/2019853243_goldman_sachs_microsoft_os_has_gone_from_more_than.html

United States Mission to European Union. (2009, October 27). Oracle concerned over EU investigation of Sun merger. *WikiLeaks*. WikiLeaks cable: 09BRUSSELS1455. Last accessed August 2, 2014 from http://wikileaks.org/cable/2009/10/09BRUSSELS1455.html

United States vs. Microsoft (2002). Final Judgement. Retrieved from the United States Department of Justice. Last accessed June 2, 2014 from http://www.justice.gov/atr/cases/f200400/200457.htm

United States vs. Microsoft. (2000). Conclusions of Law. Retrieved from the United States Department of Justice. Last accessed May 31, 2014 from http://www.justice.gov/atr/cases/f218600/218633.htm

United States vs. Microsoft. 84 F. Supp. 2D 9. (1999). Retrieved from Westlaw Campus database.

Vaughn-Nichols, S.J. (2014, March 28). Red Hat reveals CentOS plans. *CDNet*. Last accessed May 1, 2014 from http://www.zdnet.com/red-hat-reveals-centos-plans-7000027812/

Von Hippel, E. (2005). *Democratizing innovation*. Cambridge, MA: The MIT Press.

Ward, J. (2013). Apple lore: The creation of the Macintosh. *Vectronic's Apple World*. Last accessed May 15, 2014 from http://vectronicsappleworld.com/macintosh/creation.html

Webb, E.J., Campbell, D.T., Schwartz, R.D., Sechrest, L., & Grove, J.B. (1981). *Non-reactive measures in the social sciences, 2nd ed.* Boston, MA: Houghton Mifflin Company.

Weber, S. (2004). *The success of open source*. Cambridge, MA: Harvard University Press.

Whitney, L. (2009, December 14). Oracle pledges to play well with MySQL. *Cnet*. Last accessed August 2, 2014 from http://news.cnet.com/8301-1001_3-10414686-92.html

Wilcox, J. (2001, March 13). Jackson exits Microsoft discrimination case. *Cnet.* Last accessed June 2, 2014 from http://news.cnet.com/Jackson-exits-Microsoft-discrimination-case/2100-1001_3-254049.html

Williams, R. (1975). *Television: Technology and cultural form*. New York, NY: Schocken Books.

Williams, S. (2002). *Free as in freedom: Richard Stallman's crusade for free software*. Sebastopol, CA: O'Reilly.

Wu, T. (2010). *The master switch: The rise and fall of information empires*. New York, NY: Alfred A. Knopf.

Young, R., & Rohm, W.G. (1999). *Under the radar: How Red Hat changed the software business – and took Microsoft by surprise*. Scotsdale, AZ: The Coriolis Group.

Zittrain, J. (2008). *The future of the Internet and how to stop it*. New Haven, CT: Yale University Press.

Zuse, Konrad, Bauer, F. L., Zemanek, H., McKenna, P., & Ross, J. A. (2010). *The computer: My life*. New York, NY: Springer-Verlag New York Inc.