MODEL TO ENSURE CLEAN INTELLECTUAL PROPERTY IN SOFTWARE DEVELOPMENT PROJECTS

by

Kamal M. Hassin

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Applied Science in Technology Innovation Management

Department of Systems and Computer Engineering,

Carleton University

Ottawa, Canada, K1S 5B6

August 2008

© Copyright 2008 Kamal M. Hassin
NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimées ou autrement reproduits sans son autorisation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.
ABSTRACT

This research examined how companies ensure implementation of clean intellectual property (IP) when their development projects incorporate open source code, third party code, and internal code. Various sources of data were analyzed to produce: a categorization of methods used to ensure clean IP, opportunities for entrepreneurs, and a model that defines clean IP building on an analogy with the clean room standards model. There were three key findings. First, the distinguishing characteristic between methods used to ensure clean IP is the stage of software development at which clean IP is addressed. Second, opportunities for entrepreneurs can be classified into two groups: functionality requirements and services to companies that develop software which incorporate open source code, third party code, and internal code. Third, the clean IP model developed in this research consists of four levels defined by six variables that can be used to clarify the cleanliness of software IP.
ACKNOWLEDGEMENTS

I am grateful to my thesis supervisor, Professor Tony Bailetti, who supervised this research and provided opportunities for me to succeed both inside and outside the university. Through his innovative guidance, unique encouragement, and firm commitment, a mere idea was transformed into a Master's thesis. Most of all, I am thankful to him for lessons gained throughout the development of this thesis which are not taught in the classroom. I will carry these with me throughout my career.

I would like to thank Peter Hoddinott and Mahshad Koohgoli for their help in planning the early stages of this research; Michael Weiss and Steven Muegge for their constructive suggestions and feedback throughout this research; Rowland Few for his patience in handling the "dirty work" of this project as well as his valuable business insights.

I am grateful to the Talent First Network and Nortel for generously funding this research.

I am thankful to my research colleagues and friends, Andrew Pullin and Monica Mora. Their friendship and constant feedback on numerous drafts and presentations helped me complete this research.

Finally, I thank my mother for setting high expectations and instilling her model work ethic and values in me which inspired me to pursue higher education and achieve my research goals. Her unwavering support was essential to completing this project.
# TABLE OF CONTENTS

1 INTRODUCTION...........................................................................................................1
   1.1 Objectives .............................................................................................................2
   1.2 Deliverables .........................................................................................................3
   1.3 Relevance ............................................................................................................3
   1.4 Contribution ........................................................................................................4
   1.5 Organization .......................................................................................................5

2 LITERATURE REVIEW .................................................................................................6
   2.1 Clean rooms .........................................................................................................6
      2.1.1 Clean room applications ...............................................................................7
      2.1.2 Clean room classifications ............................................................................8
   2.2 IP protection and innovation ................................................................................10
      2.2.1 Socio-economics and property rights ............................................................10
      2.2.2 IP protection and business strategy ...............................................................11
      2.2.3 Software IP rights and open source ...............................................................13
   2.3 Legal implications of lack of clean IP ..................................................................16
      2.3.1 Legal property rights ....................................................................................16
      2.3.2 Licensing regimes .........................................................................................18
   2.4 Open source license compliance ...........................................................................20
      2.4.1 License choice ..............................................................................................21
      2.4.2 License types .................................................................................................22
      2.4.3 Business implications ....................................................................................25
      2.4.4 Customer indemnification ............................................................................27
   2.5 Software design processes ....................................................................................28
   2.6 Lessons learned ...................................................................................................29

3 RESEARCH METHOD ................................................................................................32
   3.1 Unit of analysis ....................................................................................................32
   3.2 Analysis period ....................................................................................................33
   3.3 Sample selection ..................................................................................................33
   3.4 Method ................................................................................................................34
      3.4.1 Extract categories, opportunities, and clean IP model variables from the
3.4.2 Develop inventory of IP infringement cases .................................................. 35
3.4.3 Refine Version 1 of deliverables using cases .................................................. 36
3.4.4 Develop questionnaire ................................................................. 37
3.4.5 Sample selection ............................................................................. 37
3.4.6 Collect questionnaire responses .............................................................. 38
3.4.7 Refine Version 2 of deliverables using questionnaire responses .............. 38
3.4.8 Place software projects in clean IP model and produce observations .......... 39
3.4.9 Refine Version 3 of deliverables using three experts .............................. 39
3.4.10 Make observations about the placement of software projects .............. 40

4 RESULTS ............................................................................................................. 41

4.1 Version 1 of deliverables ................................................................. 42
4.1.1 Categorization of methods ................................................................. 42
4.1.2 Opportunities for entrepreneurs .......................................................... 43
4.1.3 Clean IP model ............................................................................. 44

4.2 Cases of IP infringement ................................................................. 47
4.2.1 Open source license infringement ......................................................... 48
4.2.2 Other IP infringement .................................................................. 50

4.3 Version 2 of deliverables ................................................................. 51
4.3.1 Categorization of methods ................................................................. 51
4.3.2 Opportunities for entrepreneurs .......................................................... 51
4.3.3 Clean IP model ............................................................................. 52

4.4 Version 3 of deliverables ................................................................. 54
4.4.1 Categorization of methods ................................................................. 54
4.4.2 Opportunities for entrepreneurs .......................................................... 55
4.4.3 Clean IP model ............................................................................. 56
4.4.4 Placement of software projects in clean IP model .............................. 58

4.5 Version 4 of deliverables ................................................................. 58
4.5.1 Categorization of methods ................................................................. 58
4.5.2 Opportunities for entrepreneurs .......................................................... 59
4.5.3 Clean IP model ............................................................................. 60

4.6 Observations ............................................................................. 63
4.6.1 Categorization of methods ................................................................. 63
LIST OF TABLES

Table 1. Clean room standard levels (adapted from Turpin (2004)) ...........................................9
Table 2. Summary of the private-collective model .................................................................14
Table 3. Comparison of software development property regimes ........................................18
Table 4. Comparison of open source licenses and closed source licenses ............................20
Table 5. Legal factors to consider when using open source components .............................22
Table 6. Tasks undertaken to manage license compliance .....................................................27
Table 7. Steps in research method .........................................................................................34
Table 8. Version 1: categorization of methods ........................................................................42
Table 9. Version 1: list of opportunities ..................................................................................44
Table 10. Version 1: clean IP model .......................................................................................46
Table 11. Literature articles used to produce Version 1 of the clean IP model ....................47
Table 12. Cases of open source license infringement ............................................................48
Table 13. Cases of other IP infringement ...............................................................................50
Table 14. Version 2: list of opportunities ...............................................................................52
Table 15. Version 2: clean IP model .......................................................................................53
Table 16. Cases used to produce Version 2 of the clean IP model ........................................54
Table 17. Version 3: categorization of methods .......................................................................55
Table 18. Version 3: list of opportunities ...............................................................................55
Table 19. Version 3: clean IP model .......................................................................................57
Table 20. Software projects placed in clean IP model ...........................................................58
Table 21. Version 4: categorization of methods .......................................................................59
Table 22. Version 4: list of opportunities ...............................................................................60
Table 23. Version 4: clean IP model .......................................................................................61
Table 24. Analysis of clean IP model using legal cases .............................................................73
LIST OF APPENDICES

APPENDIX A: INVENTORY OF IP INFRINGEMENT CASES ......................... 84
  A.1 Open source license infringement cases .................................................. 84
  A.2 Other IP infringement cases ................................................................. 93

APPENDIX B: QUESTIONNAIRE ................................................................. 100
1 INTRODUCTION

This chapter provides the objectives, deliverables, relevance, and contribution of this research, as well as the organization of this thesis.

Company and product value depend on the cleanliness of intellectual property (IP), not just the protection of IP as stated in frameworks developed by Gans, Hsu, and Stern (2002) and Teece (1988). For example, in a merger and acquisition situation, the value of a software company depends more on how clean the company’s IP is than the company’s ability to protect unclean IP.

For the purpose of this research, a company has clean IP if they can prove certainty of IP ownership in court by producing code pedigree (Black Duck Software, 2006), and ensuring license compliance and combined license compatibility (Copenhaver, 2006).

The term “clean room” is used to describe manufacturing processes that avoid injecting faults into the finished product via contamination in the atmosphere (Broadfoot & Hopcroft, 2005). Some of the control variables include air pressure, temperature, humidity, and number and size of particles per volume.
An analogy between a clean IP model and the clean room model was conceptualized. Both models use various dimensions to determine a specific level of cleanliness. The purpose of both models is to prevent the introduction of unwanted components into the final product. An unacceptable level of these components could render the final product defective. In the case of the clean room model, the unwanted component is the number of particles per volume which could damage electronic circuitry. In the clean IP model, the unwanted component is code of unknown origin and license which could dissuade potential buyers from purchasing the product. Both models define a level of standards for product development.

1.1 Objectives

The objectives of this research are to:

- Examine how companies ensure implementation of clean IP when their development projects incorporate open source code, third party code, and internal code
- Address two problems identified by entrepreneurs: (i) the inability to ensure clean IP in a timely manner reduces the value of software being sold, and (ii) the inability to ensure clean IP raises the likelihood of lawsuits resulting from IP infringement.
1.2 Deliverables

This research produces three outcomes:

- Categorization of the methods used by software development companies to ensure clean IP
- Identification of opportunities for entrepreneurs
- Model that defines clean IP based on an analogy with the clean room standards model

1.3 Relevance

This research is relevant to buyers and sellers of software, executives of software companies, entrepreneurs, and software support managers. This research is relevant to buyers and sellers of software because both groups require greater assurance that the IP of the products that anchor their transactions is clean. The clean IP model developed in this thesis can clarify the cleanliness of software IP. By understanding the level of clean IP at which their products fall, buyers and sellers can act to limit post-transaction liability and IP infringement.

This research is relevant to software company executives because the clean IP model developed provides them with greater confidence that they can sell their businesses or raise capital to grow them. Using the clean IP model can speed up the negotiation process in merger and acquisition or funding arrangements. The cases of IP infringement highlight the risk arising from lack of clean IP and costs of correcting infringing code.
This is an important hidden cost that can increase the total cost of ownership (TCO), but is not often recognized by executives of software companies (Davidson, 2006).

This research is relevant to entrepreneurs because it identifies opportunities that currently exist. Entrepreneurs providing products, tools, and services to companies that develop software that incorporates open source code, third party code, and internal code will be able to meet customer needs.

This research is relevant to software support managers because they can use the clean IP model to implement the method that best fits their desired level of clean IP. Choosing the correct method can accelerate software development and avoid costly delays and product re-engineering due to license compliance failure.

1.4 Contribution
This research makes two contributions. First, it anchors a clean IP model around literature and analysis of real data collected from a variety of sources. We find that there are various levels of clean IP depending on product market requirements. For example, some service providers refuse to purchase code that incorporates open source software while, on the other hand, universities promote this type of purchase. The clean IP model can be used as an industry standard to help grade the cleanliness of software IP. Action can then be initiated to achieve the desired level of clean IP for a product.
The second contribution is that this research identifies opportunities for entrepreneurs. These opportunities are anchored around functionality requirements identified when examining the various approaches used to deliver clean IP as well as opportunities to provide products, tools, and services to companies that develop software that incorporates open source code, third party code, and internal code. Entrepreneurs can act on these opportunities to add new features and develop new products and services.

1.5 Organization

The remainder of this thesis is organized into the following chapters:

- Chapter 2 is a review of the relevant literature and the lessons learned
- Chapter 3 explains the research method used to produce the deliverables
- Chapter 4 presents the results of this research
- Chapter 5 provides a discussion of results
- Chapter 6 presents conclusions, limitations of this research, and suggestions for future research
2 LITERATURE REVIEW

This chapter reviews the relevant literature. Section 2.1 reviews the literature on clean rooms. Section 2.2 reviews the literature on IP protection and innovation. Section 2.3 explains the legal implications of lack of clean IP. Section 2.4 reviews the literature on software license compliance. Section 2.5 reviews the literature on software design processes. Lastly, Section 2.6 presents the lessons learned from the literature review.

2.1 Clean rooms

A clean room is an enclosed environment in which pollutants in the atmosphere are controlled (Deforge, 2006). These rooms may house laboratories, manufacturing processes, or sensitive equipment which can be damaged by dust, airborne microbes, aerosol particles and chemical vapours. In many industries including electronics, pharmaceuticals, and biotechnology, clean rooms have well established best practices and parameters that are specific to development or manufacturing operations. Depending on the product, clean rooms can be set at different levels in order to achieve precise control of the environmental parameters. The clean room standards model provides a structure and set of established best practices that can be applied successfully to many different industries to prevent injecting faults into final products. This stream examines different clean room applications and classifications.
2.1.1 Clean room applications

The electronics and semiconductor industry is the largest application of clean room usage. This industry has a strict set of standards and processes to ensure that products manufactured in a clean room are not contaminated. Any particles that exist in the clean room must be non-conductive and inert. Other electronics clean room design parameters include machine size, speed of air movement, and product placement.

There is an increasing need for clean room technology in the pharmaceutical and biotechnology industry. This steadily growing sector represents the second largest consumer of clean room space (McIlvane, 2003). Its main focus is preventing contamination from airborne particles such as bacteria and viruses. As new products are developed, the presence new contaminants create the need to modify existing clean room levels and create new levels.

In another application of using a clean room approach, a study by Derwin (1989) describes a development method to greatly reduce legal risks involved in developing "functionally compatible high-technology products." The purpose of using a clean room approach is to provide defence to claims of copyright infringement or trade secret appropriation by allowing a defendant to prove that the second product, which was developed in a clean room environment, was created without access to the first. Many companies may use clean room procedures in product development, but such procedures differ from company to company. Furthermore, this study suggests that companies using
flawed procedures are likely to end up worse off than if no precautions had been taken (Derwin, 1989).

2.1.2 Clean room classifications

Clean rooms are rated by two main classification systems. In the 1960s, the US General Service Administration (GSA) in cooperation with the Institute of Environmental Sciences and Technology (IEST) developed the first set of clean room standards for people involved in government contract work. This system is known as a federal standard (US FED STD 209E) and has been modified during the 1980s and 1990s (Turpin, 2004). In 2001, this system was replaced by a series of international standards developed by the International Organization for Standardization (ISO) Technical Committee 209 (TC209), comprised of representatives from 19 different nations. This was done mainly because each country had its own clean room standard and it was difficult for countries who wanted to build clean rooms in different countries, requiring different standards. The currently used standard is called the ISO 14644-1, while ISO 14698-1 and -2 refer to clean room biocontamination standards (Deforge, 2006; Turpin, 2004).

To certify a clean room for a classification, samples of air are continuously monitored by a particle counter and the number and size of identified particles is evaluated. The ratings correspond to the maximum number and size of particles permitted per volume of air. For example, a Class 1000 clean room means the environment must not contain more than 1000 0.5µm particles per cubic meter per minute. Table 1 presents different clean room
levels from both the older US FED STD 209E standard and newer ISO 14644-1 standard.

Table 1. Clean room standard levels (adapted from Turpin (2004))

<table>
<thead>
<tr>
<th>ISO Class</th>
<th>US FED STD 209E Class</th>
<th>Maximum number of particles per m³ by µm size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.1µm</td>
</tr>
<tr>
<td>ISO 1</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>ISO 2</td>
<td>-</td>
<td>100</td>
</tr>
<tr>
<td>ISO 3</td>
<td>1</td>
<td>1,000</td>
</tr>
<tr>
<td>ISO 4</td>
<td>10</td>
<td>10,000</td>
</tr>
<tr>
<td>ISO 5</td>
<td>100</td>
<td>100,000</td>
</tr>
<tr>
<td>ISO 6</td>
<td>1000</td>
<td>1,000,000</td>
</tr>
<tr>
<td>ISO 7</td>
<td>1000000</td>
<td></td>
</tr>
<tr>
<td>ISO 8</td>
<td>10000000</td>
<td></td>
</tr>
<tr>
<td>ISO 9</td>
<td>Room air</td>
<td></td>
</tr>
</tbody>
</table>

In order to develop a clean room standard, different factors must be observed from a process. The assembly line from the hard disk drive industry is an example where a clean room standard was developed (Mrad, 1999). This study determined what factors characterize the assembly process in order to produce an acceptable set of measurements that can help product development teams present more manufacturing products. In particular, the assembly process depends on the product nature, market forecast, production geographical location, and other special constraints. This study followed progressive stages to acquire these factors in order to produce a clean room standard which never existed before in this specific setting.
2.2 IP protection and innovation

There is a connection between the literature of economics, management practice, and technology. This connection is highlighted to understand the importance of IP protection and how to protect a company's IP. Effective IP protection is necessary to create and capture value from innovation. However, the academic literature emphasizes IP protection over ensuring clean IP. This literature stream examines socio-economics and property rights, IP protection and business strategy, and software rights and open source.

2.2.1 Socio-economics and property rights

The literature stream of socio-economics and property rights provides an understanding of the importance of property rights and how they are used. Property rights are the basis of legal systems and have many business implications. Costello and Costello (2005) define property rights as social institutions that define or delimit the range of privileges that are granted to individuals (or entities) regarding specific assets. In a community, there are economic and social relationships among individuals and the state. Property rights often describe the relationships defining ownership of resources, rules for allocation of these resources, and rules for usage of these resources by state and community.

The value of goods in any transaction depends on the allowed rights of action over the goods and the degree to which these rights are enforced (Coase, 1960; Demsetz, 1964; Furubotn & Pejovich, 1972). By changing the property rights that accompany ownership
of resources, it is possible to maximize wealth from these resources. Different authors from economics have discussed how to effectively allocate property rights in accordance with externalities; however, strategies may vary between industries. Coase (1960) and Regan (1972) emphasize that transaction costs are relevant to assigning property rights. If changing the mix of property rights that accompany ownership increases the value of property, such a change will be desirable to maximize wealth (Demsetz, 1964).

From this literature stream, a few conclusions are developed by Furubotn & Pejovich (1972). These are summarized as follows:

- A company must balance transaction costs and the opportunity to operate all functions internally
- The state should create institutions which minimize transaction costs so that resources can be correctly and inexpensively allocated
- Property rights should initially be assigned to the players who can derive the most value from them
- Privately owned resources tend to be allocated to the highest valued uses

2.2.2 IP protection and business strategy

Gans, Hsu, and Stern (2002) study the determinants of commercialization strategy for start-up innovators. They examine whether the returns on innovation are earned through product market competition or through cooperation with established firms (through
licensing, alliances, or acquisition). Three hypotheses are developed: the relative returns to cooperation are increasing in

- Control over intellectual property rights
- Low transaction costs
- Sunk costs associated with product market entry

The strength of intellectual property rights (IPR) affects the absolute returns to innovation (regardless of commercialization strategy) by reducing the threat of expropriation (Gans et al., 2002). While expropriation may occur under competition or cooperation, the returns negotiated by the start-up with an incumbent reflect their ability to threaten competitive entry and the transaction costs of bargaining. These two factors ensure that the returns to cooperation are more sensitive to the strength of IPR than the returns to competition, and so an increase in the strength of IPR increases the relative returns to cooperation (Gans et al., 2002). However, there is no emphasis on the relationship between clean IP and strength of IPR.

Teece (1988) offers a framework that describes factors which determine who benefits from innovation: the first to market or those that follow. This framework identifies three constructs that decide who captures value from the introduction of an imitation:

- Appropriability regime
- Stage of technology life cycle relative to dominant design
- Ownership of complementary assets
The term appropriability regime refers to aspects of the commercial environment that govern an innovator’s ability to capture value from innovation. The most important dimensions of such a regime are the nature of the technology and the efficacy of IP protection. The property rights environment within which a firm operates can be classified according to the nature of the technology and the efficacy of the legal system to assign and protect IP. However, there is no emphasis on the relationship between clean IP and efficacy of IP protection.

2.2.3 Software IP rights and open source

Economists suggest factors that have increased open source interest and development in the software industry. These include among others (Lerner & Tirole, 2002):

- Rapid diffusion of open source software into the software market
- Significant capital investment in open source from major corporations
- New organizational structures regarding collaborative development processes

The emergence of open source has also created economic problems such as profit motivations for contributors and innovation, creating a hierarchical company organization, and diffusion of open source into environments with proprietary standards (Bonaccorsi & Rossi, 2003). Strong protection of IP is essential to the promotion and development of new applications and services, therefore creating a successful market. IP protection motivates creative individuals to innovate and seek economic rewards (Simon, 1996). In particular, the adoption of copyleft licenses is positively related to the strength of social and ideological motivations on the part of firms and to their degree of
involvement into the open source community (Bonaccorsi & Rossi, 2003). Other motivations for contributing to an open source project are further explored in the literature (Bonaccorsi & Rossi, 2003; Ghosh, 2003; Kogut & Metiu, 2001; O'Mahony, 2003; Rossi, 2004; Lerner & Tirole, 2002). However, the strength of IP rights is not absolute. Some economists argue that the greatest value is not created by the strongest and most restrictive IP protection (Parker & Van Alstyne, 2005). This argument is furthered by proponents of the open software movement, who argue for value created by peer review, reuse, and complementary investment (Parker & Van Alstyne, 2005; Raymond, 2000).

In organization science, two recognized models have been proposed to capture innovation from open source assets: private investment and collective action. The private-collective model combines these two previous innovation models, private investment and collective action (von Hippel & von Krogh, 2003). Table 2 summarizes these two models and explains how the compound private collective model addresses these single models.

<table>
<thead>
<tr>
<th>Single model</th>
<th>Main principles</th>
<th>Compound model improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private investment</td>
<td>Innovation will be supported by private investment and private returns can be appropriated from these investments (Demsetz, 1964)</td>
<td>Eliminates the assumption that free revealing of innovations developed with private funds will represent a loss of private profit for the innovator</td>
</tr>
<tr>
<td></td>
<td>To encourage private investment, IP protection mechanisms (patents, copyrights, trade secrets) are granted to innovators</td>
<td></td>
</tr>
<tr>
<td>Collective action</td>
<td>Contributors relinquish control of knowledge they have developed for a project and make it a common good</td>
<td>Eliminates the assumption that free riders will be able to obtain benefits from the</td>
</tr>
</tbody>
</table>
Emphasises the importance of recruiting and motivating participants to increase the attractiveness of contributing to the common pool

Open source software is an example of this innovation model because new knowledge is created from private funding and then released freely to the public domain. Programmers contribute freely to the common pool of the public good because they gain private benefits from doing so (O'Mahony, 2003).

Unless the correct strategy is in place, an innovator may create value but appropriate little of it. The emergence of open source has posed a unique challenge to researchers, developers, and practitioners: how can commercial firms appropriate returns from innovations that are created in the public domain (Dahlander, 2005; Kogut & Metiu, 2001)? Even if many important open source resources are used in a software project, the owner does not have to be restricted from capturing private returns. Based on empirical data, the framework proposed by Dahlander (2005) suggests that firms appropriate returns from open source software by relying mainly on selling consultancy hours. Depending on the type of software innovation and knowledge, traditional mechanisms of IP protection (patents, copyrights, trade secrets) are often costly and time-consuming to enforce. As a result, firms must balance these inefficiencies by putting greater emphasis on first-mover advantages and creating network externalities. This will create a community that can provide important inputs to the innovative process and help early market entrants acquire superior resources and capabilities (Dahlander, 2005). A complementary view promotes open source software as a production model that avoids
inefficiencies of strong IP regimes, yet allows continuous improvement and value creation (Kogut & Metiu, 2001). The property rights of various open source licenses are placed in the public domain. In some cases, it is also possible to capture returns by developing proprietary applications. Lastly, another option is to use dual licensing in which a project is released under both an open source license and commercial license. However, this poses many economical and legal risks to consider regarding strength of IP ownership. Dual licensing also requires the use of a license with a strong copyleft clause and all legal rights to distribute are managed by a single entity (Valimaki, 2003).

2.3 Legal implications of lack of clean IP

Legal property rights are used to create and capture value from technology innovation. Thus, law and technology are interdependent. Open source licensing and the open source development model can be a viable means of producing and disseminating software and other types of creative works. Software developers use law as a means of “regulating” how their code is handled in “cyberspace” (Lee, 1999). This literature stream examines legal property rights in open source projects and the principles of open source licensing regimes.

2.3.1 Legal property rights

The scope of legal property rights covers many areas of technology innovation including hardware designs, software source code, and programmer designs and inventions. Legal rights differ from the socio-economic property rights reviewed in Section 2.2.1 by
making third party adjudication and enforcement possible; however legal rights are not necessary or even sufficient for the existence of economic rights. Property rights institutions can range from formal arrangements (constitutional provisions, statutes, judicial rulings) to informal norms and customs regarding allocation and use of assets (Costello & Costello, 2005). Federal laws in Canada and the US provide three main ways of protecting proprietary software IP:

- Patents: exclusive temporary rights granted to protect inventions (Oz, 1998)
- Copyrights: exclusive rights to protect the expression of ideas (Rossi, 2004)
- Trade secrets: protection of a company’s confidential information against unauthorized use or disclosure (Davidson & Holloway, 2006)

Oz (1998) provides an extensive list of both pros and cons of software copyrights and patents. The reasons for using these IP mechanisms are extensive and must be carefully analyzed when dealing with any type of software.

In addition to diffusion of knowledge into society, the patenting of innovation creates incentives for innovation (Vemuri & Bertone, 2004). Individuals are motivated to patent their software innovations in order to achieve some sort of benefit, usually financial. On the other hand, the open source software community claims that patenting obstructs the development of software by imposing legal restrictions. The distinction between open source standards and patented technologies is not always clear. This is reflected in previous legal cases regarding the open source Unix Compress utility and LZW
algorithm, and W3C standards body and SOAP 1.2 specification (Vemuri & Bertone, 2004).

2.3.2 Licensing regimes

The use of open source software introduces different legal requirements into software projects. Licences are the most important institution in the governance structure of open source projects (Bonaccorsi & Rossi, 2003). The correct license must be chosen depending on the origins, composition, and intended purpose of the software project. The literature regarding open source license details is extensive. These are further explained in Section 2.4. To understand the domain of legal IP rights, the use of software development property regimes is a way to classify software assets and their according IP mechanisms (de Laat, 2005). Table 3 compares private regime and public regime.

<table>
<thead>
<tr>
<th>Table 3.</th>
<th>Comparison of software development property regimes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project type</td>
<td>Private regime</td>
</tr>
<tr>
<td>Outsider access to firm's assets</td>
<td>Proprietary</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>IP protection mechanisms</td>
<td>Patents, copyrights, trade secrets</td>
</tr>
<tr>
<td>Purpose of IP protection mechanisms</td>
<td>Help software owners preserve market share, maximize returns, preserve IP rights (Horne, 2001)</td>
</tr>
</tbody>
</table>

There are two main methods used to maintain a high degree of separation between closed source software and an open source development project. The first way is by denying a licensee the discretion to take the licensee’s modifications to the open source code base
and turn that modification into a proprietary piece of software. A restrictive license not only ensures that the code base remains open, but it makes certain that any derivative works from that original source code base also remain open (Lee, 1999). The second way to maintain uniqueness of open source software is through the use of licensing provisions to prevent the combination of open source software with closed source software. However, a provision against mixing does not necessarily mean that a piece of open source code cannot be delivered on the same medium with a piece of closed source software, nor will it mean that open source software cannot be used along with closed source software on a user’s computer (Lee, 1999).

There are similarities and differences between open source licenses and traditional IP mechanisms. Open source licenses are based on copyright law. Copyright protections are granted to allow investors to capture returns on their investment while also providing an incentive for innovators to release their works so that the public can benefit from them (O’Mahony, 2003). Relationships between copyright owner and licensee as well as between owner and user are also governed. Under copyright law, copyright holders have exclusive rights pertaining to software which include the following (Arne & Yates, 2005; Lee, 1999):

- To reproduce the work
- To prepare derivative works based upon the original copyrighted work
- To distribute the work
The holder of these rights can either license these rights or sell them to a third party. Table 4 compares the main characteristics of open source licenses and closed source licenses.

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Open source license</th>
<th>Closed source license</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ownership of rights</td>
<td>Granted to licensee</td>
<td>Granted to licensor</td>
</tr>
<tr>
<td>Derivative works and distribution</td>
<td>Not allowed</td>
<td>May be allowed (Madison, 2005)</td>
</tr>
<tr>
<td>Enforceability</td>
<td>Enforceable against a user by copyright law, injunctions permitted for infringements</td>
<td>Enforceable against a user by contract law, does not injunctions not permitted for infringements</td>
</tr>
</tbody>
</table>

In the future, open source licenses will continue to play an increasing role in software projects. Their evolution over time is shown by releases of new versions and interpretations. The importance of IP rights and open source licenses also increases as the complexity of projects and the cost of coordinating those increases. Where significant costs have to be sunk to ensure success, more robust legal and social structures are required. Where no such costs are needed, production may succeed in more anarchic structures (McGowan, 2001).

### 2.4 Open source license compliance

The use of open source software is increasing throughout the software industry. Through the sharing of code, developers signal their talent, learn new skills and exchange ideas and knowledge in a common language (Bonaccorsi & Rossi, 2003). Development practices are also changing to reflect the use of open source. Software companies operate in a
"mixed-IP environment" in which projects may contain legacy software, proprietary components, internally developed software, and open source elements (Copenhaver, 2006). Even for developers and lawyers who regularly use open source, understanding licensing terms can be challenging. Furthermore, some open source licenses do not follow conventions of tight legal drafting, giving rise to some unnecessary uncertainty in the meaning of the licenses. Because open source licenses often use copyright law generally and the concept of derivative works specifically to dictate whether certain rights or obligations exist, a good understanding of copyright law and what a derivative work is and is not can be important (Arne & Yates, 2005). This stream of literature examines open source license choice and license types, business implications for software companies using open source software, and customer indemnification.

2.4.1 License choice

There are hundreds of open source licenses available to software developers. New licenses and versions of existing licenses are often released to accommodate the evolving issues in software development. They serve a wide range of purposes, yet share many common features, such as offering free software to users and minimizing restrictions on use and redistribution (Hecker, 2000; Lee 1999). The choice of an open source license depends on several factors on a larger scope such as:

- Type of product being offered by the company
- Business model in which the product will follow
- Type of customers who will buy the product
There are also legal factors to consider when using open source components. Table 5 lists brief descriptions of these legal factors (Hecker, 2000; Lerner & Tirole, 2005; Ruffin & Ebert, 2004).

**Table 5. Legal factors to consider when using open source components**

<table>
<thead>
<tr>
<th>Legal factor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction with other software</td>
<td>A company must ensure that adopted open source components can co-exist with other products. Development processes may need to be modified if existing products are linked to others governed by open source licenses.</td>
</tr>
<tr>
<td>Third party technology</td>
<td>Components of a product may be licensed by a third party that may not be compatible with open source licenses. This may require developers to seek permission for inclusion of third party code, or replace the code with open source code providing equivalent or similar functionality.</td>
</tr>
<tr>
<td>Future implications</td>
<td>Using specific open source licenses may result in unexpected legal conflicts in the future if correct due diligence is not performed. The risks of involuntary code disclosure according to restrictive open source licenses must be examined before an open source license is adopted.</td>
</tr>
</tbody>
</table>

There are various determinants for open source license choice. The decision is shaped not just by the preferences of the licensor itself, but also by that of the community of developers.

### 2.4.2 License types

While some open source licenses ensure openness of source code and the ability to freely redistribute licensed software, there is a wide spectrum of available licenses and interpretations. At both ends, there is strong support from the open source community as to which side software licensing should reside. On one end of the scale are those who
believe that all software should be free and that IP protection is a barrier to software
development. Led by Richard Stallman, supporters of these ideas believes that only the
"freeing" of source code can lead to freedom of both users and developers (Lee, 1999).
They tend to favour more restrictive licenses such as the GPL. On the other end of the
scale are those who interpret open source in a more utilitarian manner. They tend to focus
on the qualitative virtues of open source software and embrace the commercialization of
the open source movement (Lee, 1999). Some notable people in favour of permissive
licenses include Linux guru Linus Torvalds, Open Source Initiative co-founder Eric
Raymond, publisher Tim O'Reilly, and Perl inventor Larry Wall. In between these two
extremes are a myriad of groups with their own interpretation of open source licenses.
However, it is useful to classify the commonly used open source licenses into these two
groups, restrictive and permissive, in order to understand their implications regarding
compliance and composition with other software (Lerner & Tirole, 2005).

The GPL garners a lot of attention because it is the most commonly used open source
license and also governs the Linux operating system. Its general provisions place
restrictions on copying, distribution, and modification. The copyleft clause maintains a
strict barrier between open code and closed code. Copyleft grants the right to use,
modify, and distribute the program on the condition that the licensee also grants similar
rights over the modifications he has made (Mustonen, 2003). This allows free access to a
program's source code, but prevents it from becoming private IP. Projects governed by
copyleft licenses are usually geared towards end-users and projects that software
developed in a corporate setting.
Combining GPL governed code with closed source programs means that the IP rights of the closed source software may be overruled by the GPL, a license that weakens IP rights (Arne & Yates, 2005; Lee, 1999). As a result, some software companies developing proprietary products forbid GPL code from being inserted in programs released under licences that do not allow the access to the source code. The GPL is often called a “viral” license due to its ability to “infect” projects that include its code. Thus, developers who want to incorporate GPL governed code into their software projects should understand the risks associated with interoperability, as well as whether the GPL imposes restrictions on the developer’s proprietary code as a collective work of open source software (Arne & Yates, 2005). However, the use of mixed license models implies that companies are not entirely worried about the viral nature of copyleft because they know how to separate the legal implications of different licensing schemes.

In contrast, permissive (less restrictive) licenses usually govern projects geared towards developers, designed to run on commercial operating systems, and geared towards the Internet (Lerner & Tirole, 2005). Examples of open source licenses in this category include the BSD, MIT, and Apache License. They allow licensees make changes to the original source code and keep it private. Furthermore, open source and closed source software can be combined without restriction on releasing source code. The main stipulation of these licenses is to give credit to the original copyright holder(s) of the software and to those who contributed to the source code. Less focus is applied to this category of licenses from a legal point of view due to the relaxed restrictions and lack of legal conflicts.
There are hundreds of other open source licenses that can be classified in between these two opposite categories. Choosing an appropriate open source license requires analysis from different technical and legal perspectives. Even after a license is chosen, compliance and compatibility assurance is an ongoing process throughout the life cycle of a program.

2.4.3 Business implications

The benefits of using mature open source software and its user/support community are numerous. A few of these benefits include (Ruffin & Ebert, 2004):

- Independence from a specific supplier
- Long-term support at affordable costs
- Low cost of ownership
- Reduction in development time and costs

There are also risks associated with the use of open source software in a company environment. The use of open source licenses increases the complexity of projects and cost of coordination because significant costs have to be sunk to ensure success, and more robust legal and social structures are required. If companies do not perform extensive due diligence during or after software development, this may result in lack of clean IP. This is important because open source license compliance and compatibility determine the strength of IP rights and degree of IP cleanliness throughout the product life cycle.
Lack of clean IP can negatively impact a product and company both legally and financially. This is relevant to companies that highly value or rely on software they have developed or acquired, considering an investment in, or acquisition of software assets (Davidson, 2006). Uncertainty over clean IP infringement can result in (Black Duck Software, 2006; Copenhaver, 2006; Davidson, 2006):

- Security risks throughout a company’s software assets
- Risks threatening successful closure of merger and acquisition, or funding arrangements
- Product re-engineering and remediation which increase product time to market
- Diminish or destroy software IP value and overall company value
- Reduced ability to create partnerships from negative reputation and fear of being tainted
- Litigation that can last for years and waste company resources

Consequently, there is increasing concern among software companies regarding license compliance. These companies want to ensure that any company in which they invest or acquire has clean IP. They also want to protect the integrity of their own assets, operations, and business plans against others (Davidson, 2006). In order to do this, license compliance must be implemented and maintained during the entire software development process and product life cycle (Black Duck Software, 2006; Copenhaver, 2006; Ruffin & Ebert, 2004). Table 1 describes different tasks undertaken to manage license compliance.
Table 6. Tasks undertaken to manage license compliance

<table>
<thead>
<tr>
<th>Task</th>
<th>Actions undertaken to accomplish task</th>
</tr>
</thead>
</table>
| Introduce explicit checks in decision-making processes | • Understand obligations of licensed materials  
|                                                  | • Manage obligations and compatibility requirements                                                    |
|                                                  | • Document checks to educate other developers on potential risks and liability                          |
| Qualify any open source code introduced into the project | • Maintain control over what is in the code base  
|                                                  | • Screen introduction of external components                                                            |
|                                                  | • Conduct manual and automated code reviews                                                              |
| Disseminate technical, managerial, and legal education throughout the company | • Maintain a centralized knowledge of all external components  
|                                                  | • Provide information on software distribution, previous users, previous distributors, and possible liabilities |

Addressing the problems and issues of software development is not just a matter of introducing an effective tool and environment (Fuggetta, 2000). Ensuring license compliance is a company wide initiative involving technical, legal, and management cooperation. Understanding the possible customer liability can help to mitigate legal conflicts.

2.4.4 Customer indemnification

Lack of clean IP affects parties in business relationships with a company. Strategic partners want assurances that companies they ally with are compliant with their legal obligations. Investors also want confidence that company practices are credible and there are no hidden secrets that will affect shareholder value (Black Duck Software, 2006; Copenhaver, 2006). Indemnification affects scale for small companies and entrepreneurs who have low margins. They cannot always provide indemnity and therefore miss out on business opportunities. As a result of recent legal cases, customers are demanding more extensive IP warranties and indemnifications.
For proprietary software, the software vendor has the responsibility to handle IP infringement lawsuits. However, it is difficult for a company to grant warranties or indemnification to customers if its software product cannot guarantee clean IP (McLaughlin, 2005). In open source projects, the responsibility is shifted to the user. A 2005 study by market research firm International Data Corporation found that four percent of US companies "have been faced with the threat of vendor legal action over IP infringement; the figure rises to eight percent at firms with more than US $750 million in annual revenues (McLaughlin, 2005). Without the use of methods to ensure license compliance and clean IP, this number is likely to grow.

Davidson (2006) describes the benefits of sophisticated, automated tools that are specially engineered and marketed for use in the open source due diligence process. The main features of Black Duck and Palamida are highlighted to show how companies can save time and expenses by identifying questionable content within large amounts of code if they run automated tools periodically during software development.

2.5 Software design processes

A software process is a set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product (Fuggetta, 2000). Throughout the life cycle of a software project, processes may be used to maintain product architecture, security, documentation, quality control, and other functions. There is an underlying assumption of a correlation between
the quality of a process and the quality of the developed software (Fuggetta, 2000; Broadfoot & Hopcroft, 2005). The same can be noted from ensuring clean IP. In light of the consequences arising from lack of clean IP, developers recognize that error removal is an error prone, inefficient activity that consumes resources better allocated to getting to the code right the first time (Linger, 1994). The use of formal methods applied in this field is one way to create a clean IP framework. Organizational change may be costly and disruptive in any area of software development and the potential benefits that occur in the future are largely uncertain (Broadfoot & Hopcroft, 2005). However, formal processes may be necessary to avoid legal implications created by lack of clean IP.

An important trend in software development is the increasing importance of managing software license compliance in mixed development environments (Copenhaver, 2006; Davidson, 2006). As product development incorporates open source code, third party code, and internal code, new techniques for ensuring quality assurance as well as clean IP must be introduced to avoid IP infringement.

### 2.6 Lessons learned

The following lessons were learned from reviewing the literature.

1. The academic literature emphasizes effective IP protection over clean IP to create and capture value from innovation.
2. The company literature emphasizes multiple approaches to deliver clean IP, however the reasons for software development companies choosing specific approaches are unclear.

3. The traditional methods of protecting proprietary software IP are not sufficient for protecting IP in software development projects that incorporate open source code, third party code, and internal code.

4. The clean room standards model provides a structure and set of established best practices that can be applied successfully to many different industries to prevent injecting faults into final products.

5. It is difficult for a company to grant warranties or indemnification to its customers if its software product cannot guarantee clean IP.

6. The incorporation of open source code, third party code, and internal code in a software project requires the appropriate licensing regime to avoid legal risk.

7. Open source license compliance and compatibility determine the strength of IP rights and degree of IP cleanliness throughout the product life cycle.

8. The use of open source licenses increases the complexity of projects and cost of coordination because significant costs have to be sunk to ensure success, and more robust legal and social structures are required.
9. As product development incorporates open source code, third party code, and internal code, new techniques for ensuring quality assurance as well as clean IP must be introduced to avoid IP infringement.
3 RESEARCH METHOD

This chapter describes the research method. Section 3.1 describes the unit of analysis for this research. Section 3.2 describes the analysis period. Section 3.3 describes the sample selection. Section 3.4 describes the research method.

3.1 Unit of analysis

Three sources of data were used in this research: literature, Internet websites, and questionnaire responses. Each data source has a different unit of analysis.

The unit of analysis for the literature is a journal article. This data source is used to build Version 1 of the categorization of methods used to ensure clean IP, build the initial list of opportunities for entrepreneurs, and build the initial clean IP model.

The unit of analysis for the Internet websites is a single case of IP infringement. This data source is used to refine the categorization of methods used to ensure clean IP, refine the list of opportunities for entrepreneurs, and refine and analyze the clean IP model.

The unit of analysis for the questionnaire responses is a software development project that uses methods to ensure clean IP when integrating open source code, third party code,
and internal code. This data source is used to refine the categorization of methods used to ensure clean IP, refine the list of opportunities for entrepreneurs, and refine the clean IP model.

3.2 Analysis period

The period of analysis for the cases is from January 1, 2000 to April 16, 2008. This period was chosen because the majority of open source litigation cases occurred during these years. The period of analysis for the software development projects relates to projects started after January 1, 2007 and completed by April 16, 2008.

3.3 Sample selection

The sample is comprised of software development projects that incorporate open source code, third party code, and internal code into a product and use methods to ensure clean IP. In total, a sample size of nine software development projects was drawn from the population for analysis.

The sample of software projects was identified in two steps: (i) distribution of questionnaire in person, and (ii) distribution of questionnaires electronically. First, a convenience sample was acquired using an April 16, 2008 lecture at Carleton University. The lecture focused on the topic of corporate software development. Questionnaires were distributed to all attendees who:
- Participated in a software development project that started after January 1, 2007 and was completed prior to April 16, 2008
- Participated in the post-mortem meeting of the project
- Volunteered to participate in this study

Second, with the help of one experienced telecommunications manager, software support managers who met the above criteria were asked by email to complete the questionnaire. Those who agreed to participate were sent a questionnaire to complete either electronically or manually.

### 3.4 Method

Table 7 outlines the steps undertaken in this research.

<table>
<thead>
<tr>
<th>Step</th>
<th>Dominant activity</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Extract categories of methods, opportunities for entrepreneurs, and variables for clean IP model from literature</td>
<td>Version 1 of categorization of methods, list of opportunities, and clean IP model</td>
</tr>
<tr>
<td>2</td>
<td>Develop inventory of IP infringement cases</td>
<td>Descriptions of legal cases to be used to analyze complete clean IP model</td>
</tr>
<tr>
<td>3</td>
<td>Refine Version 1 of the categorization of methods and list of opportunities, and extract variables for clean IP model from cases</td>
<td>Version 2 of categorization of methods, list of opportunities, and clean IP model</td>
</tr>
<tr>
<td>4</td>
<td>Develop questionnaire based on literature and cases</td>
<td>Questionnaire to acquire details about software development projects</td>
</tr>
<tr>
<td>5</td>
<td>Select sample of software support managers, distribute questionnaire</td>
<td>A list of software support managers to complete questionnaires</td>
</tr>
<tr>
<td>6</td>
<td>Collect completed questionnaires</td>
<td>Completed questionnaires</td>
</tr>
<tr>
<td>7</td>
<td>Refine Version 2 of the categorization of methods, list of opportunities, clean IP model using</td>
<td>Version 3 of categorization of methods, list of opportunities, and clean IP model</td>
</tr>
<tr>
<td>questionnaire responses</td>
<td>Table showing software projects placed at different clean IP levels</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>8 Place software projects from questionnaire responses in clean IP model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 Refine Version 3 of the categorization of methods, list of opportunities, clean IP model by consulting with three experts in software innovation and IP</td>
<td>Version 4 of categorization of methods, list of opportunities, and clean IP model of clean IP</td>
<td></td>
</tr>
<tr>
<td>10 Make observations about each deliverable, placement of software projects and legal cases in clean IP model</td>
<td>Observations about each deliverable, placement of software projects and legal cases in the clean IP model</td>
<td></td>
</tr>
</tbody>
</table>

### 3.4.1 Extract categories, opportunities, and clean IP model variables from the literature

Version 1 of the categorization of methods and list of opportunities was developed from the information provided in Sections 2.3 and 2.4 in Chapter 2. Variables for the clean IP model were extracted from the literature by searching for factors that contribute to increasing or decreasing the level of clean IP in a software project. Four dimensions were identified for each of four levels. “Level one” was defined as one where a project has a minimal level of clean IP. “Level four” was defined as one where a project has clean IP. Descriptions were then developed for each distinguishing dimension selected. The intent of producing Version 1 of the deliverables is to better understand what is known from reading the literature. Version 1 will serve to benchmark the thesis’ contribution.

### 3.4.2 Develop inventory of IP infringement cases

An inventory of cases was developed describing IP infringement. This was done in two steps: (i) identify cases of IP infringement, and (ii) identify individual case details. In the
first step, the following keywords were entered into the Google search engine to find cases of IP infringement:

- Software license infringement
- Open source infringement
- GPL violations

In the second step, using the Google utility, keyword searches were performed again. For example, the keyword search "IBM v SCO" produced the Groklaw website. This was done to obtain additional detail on each case of IP infringement in order to analyze the clean IP model.

3.4.3 Refine Version 1 of deliverables using cases

Cases developed were examined to determine the nature of the conflicts around which the case is anchored. Indicators of these conflicts include:

- Injunctions
- Lawsuits
- Remediation costs
- Settlements
- Other legal consequences

The cases were further examined to fully understand the case origins, legal parties, conflict details, duration, and results. They were then organized by the type of issue producing the conflict. The categorization of methods and list of opportunities were
refined using these cases. Variables extracted from the analysis of these cases were used to add new variables to the existing clean IP model or refine those which were identified from the literature review. This step produced Version 2 of the deliverables.

3.4.4 Develop questionnaire
In order to further refine the categorization of methods, list of opportunities, and clean IP model, a questionnaire was developed and used to collect information from software support managers. Questions were developed based on the literature review and cases of IP infringement. Some questions are close-ended. These were used to determine characteristics of the software development project such as number of developers, type of software developed, and type of organization. Some questions are open-ended. These were used to explore the details of each method, features that are desirable in new methods, and attractive opportunities for entrepreneurs. Questionnaire respondents were also asked to add new variables to the clean IP model or modify the existing clean IP levels.

3.4.5 Sample selection
Two steps were used to identify the sample of software projects: (i) distribution of questionnaire in person and (ii) distribution of questionnaires electronically. In the first step, a convenience sample was acquired using an April 16, 2008 lecture at Carleton University. The lecture focused on the topic of corporate software development. Questionnaires were distributed to all attendees who:
• Participated in a software development project that started after January 1, 2007 and was completed prior to April 16, 2008
• Participated in the post-mortem meeting of the project
• Volunteered to participate in the study

In the second step, an experienced telecommunications manager identified software support managers who met the above criteria. The researcher contacted these managers by email and asked them to complete the questionnaire. Those who agreed to participate were sent a questionnaire to complete either via email or regular mail.

3.4.6 Collect questionnaire responses

The completed questionnaires were collected either through email or regular mail. Short follow-up interviews were done in person or over the phone. The purpose of the interview was to clarify and expand the responses provided.

3.4.7 Refine Version 2 of deliverables using questionnaire responses

Questionnaire responses were used to refine the categorization of methods, opportunities for entrepreneurs, and clean IP model.

From the questionnaire responses, characteristics of each software development project were identified. The types of characteristics were added to the existing categories
developed from the literature review and cases. This created an array of dimensions for a typology of methods used to ensure clean IP (Bailey, 1994). New array cells in the typology were created where necessary. A count of the number of software projects in each cell of each typology was produced. Functional reduction of the typology was performed to eliminate null cells (Bailey, 1994).

Respondents were asked to identify new product features in existing methods to ensure clean IP and provide descriptions of these features. These were added to the opportunities developed from reading the literature review and developing the cases. Version 3 of the deliverables included what was known from the literature review, cases of IP infringement, and questionnaire responses.

3.4.8 Place software projects in clean IP model and produce observations

The questionnaire respondents were asked to place their projects into the clean IP model. These responses were presented in a table showing the number of projects that fall into each level.

3.4.9 Refine Version 3 of deliverables using three experts

Three experts in software innovation and IP examined Version 3 of the categorization of methods, list of opportunities for entrepreneurs, and clean IP model. The experts included a senior software engineer, software entrepreneur, and experienced telecommunications
manager. Version 4 of these three deliverables incorporates the feedback received from the three experts.

3.4.10 Make observations about the placement of software projects

Observations were made about the distribution among the levels and difficulty in placing the projects in the clean IP model. Observations were also made about how the methods used to ensure clean IP were categorized. The list of opportunities for entrepreneurs was analyzed to make observations about the most attractive opportunity, based on the perception from the sample of software support managers. Finally, the legal cases were placed in the clean IP model in order to determine the reasons that prevent the companies in each case from having clean IP.
4 RESULTS

This chapter describes the results produced by this research. Section 4.1 presents Version 1 of the deliverables which is based on data extracted from the literature review. Section 4.2 analyzes the cases of IP infringement. Section 4.3 presents Version 2 of the deliverables which is based on data extracted from the literature review and cases. Section 4.4 presents Version 3 of the deliverables which is based on data extracted from the literature review, cases, and questionnaire responses. Section 4.5 presents Version 4 of the deliverables which is based on data extracted from the literature review, cases, questionnaire responses, and consultations with three experts. Section 4.6 presents observations about the final deliverables.

The outcomes of this research include:

1. Categorization of the methods used by software development companies to ensure clean IP

2. Identification of opportunities for entrepreneurs

3. Model that defines clean IP based on an analogy with the clean room standards model

This section provides four versions of the preceding deliverables. Each version uses a new source of data to make cumulative revisions the categorization of methods,
opportunities for entrepreneurs, and clean IP model. The following sources of data were used:

- Version 1: literature
- Version 2: legal cases of IP infringement
- Version 3: questionnaire responses
- Version 4: consultations with experts

4.1 Version 1 of deliverables

Version 1 of the categorization of methods, opportunities for entrepreneurs, and clean IP model is based on data extracted from the literature review.

4.1.1 Categorization of methods

Based on the literature review and analysis of promotional material from suppliers of methods used to ensure clean IP, five stages at which clean IP is addressed were identified. The main distinguishing characteristic was the stage of product development at which clean IP is addressed. Table 8 provides Version 1 of the categorization of methods used to ensure clean IP with the source of data that supports each category.

<table>
<thead>
<tr>
<th>Stage at which clean IP is addressed</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Literature</td>
</tr>
<tr>
<td>1 Never</td>
<td>Black Duck Software (2006)</td>
</tr>
<tr>
<td>2 During product development with periodic monitoring</td>
<td>Black Duck Software (2008a)</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>After product development by internal organization</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>After product development by external organization</td>
</tr>
</tbody>
</table>

The literature includes the articles from academic, management, legal journals and conference proceedings. Promotional material includes product data sheets, white papers, and presentation slide decks made available online by suppliers: Black Duck Software, Protecode, Palamida, Hewlett-Packard (HP), Eric Raymond, Alex Aiken (Stanford University), Mib Software, and Open Source Risk Management (OSRM).

4.1.2 Opportunities for entrepreneurs

Based on the literature review and analysis of promotional material from suppliers of methods to ensure clean IP, two different types of opportunities were identified: (i) functional requirements for tools to ensure clean IP, and (ii) services to companies that develop software by integrating open source code, third party code, and internal code. Five opportunities for entrepreneurs were identified. Table 9 provides short descriptions of the five opportunities and source of data organized around the two categories identified above.
Table 9. Version 1: list of opportunities

<table>
<thead>
<tr>
<th>Opportunity type</th>
<th>Description</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Literature</td>
</tr>
<tr>
<td>requirements</td>
<td>collaboration and program forking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Clean IP tool embedded in programming environment that relies on private</td>
<td>Black Duck Software (2008a)</td>
</tr>
<tr>
<td></td>
<td>code database capability, as opposed to external open source databases</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Automated clean IP tool that guarantees detection of unclean IP within a</td>
<td>Black Duck Software (2006)</td>
</tr>
<tr>
<td></td>
<td>certain time</td>
<td>Copenhagen (2006)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Davidson (2006)</td>
</tr>
<tr>
<td>Clean IP services</td>
<td>4. Implementation of a formalized process for introduction of code into</td>
<td>Black Duck Software (2008b)</td>
</tr>
<tr>
<td></td>
<td>software development projects to improve to code approval workflows</td>
<td>Palamida (2008b)</td>
</tr>
<tr>
<td></td>
<td>5. Clean IP product built on a collaborative platform that provides real-time</td>
<td>IBM (2008)</td>
</tr>
<tr>
<td></td>
<td>access to code from CVS and improve efficiency of code approval process</td>
<td></td>
</tr>
</tbody>
</table>

4.1.3 Clean IP model

Based on the literature review, three variables were extracted and categorized into different clean IP levels. Table 10 presents Version 1 of the clean IP model and definitions of each variable. Level one in the model shown in Table 10 represents the lowest standard of clean IP in which companies have no measures in place to achieve clean IP. Level four represents the highest standard of clean IP. At Level Four,
companies are certain they can prove IP ownership in court by producing code pedigree, and ensure license compliance and combined license compatibility.
<table>
<thead>
<tr>
<th>Clean IP level</th>
<th>Clean IP variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A. Preventative measures: what is done during product development to prevent legal conflicts</td>
</tr>
<tr>
<td>1</td>
<td>Policies for using third party and open source software exist</td>
</tr>
<tr>
<td>2</td>
<td>Policies for using third party and open source software are enforced</td>
</tr>
<tr>
<td>3</td>
<td>Company-wide standards exist and periodic monitoring of the product development process searching for infringements exist</td>
</tr>
<tr>
<td>4</td>
<td>Company-wide standards exist and periodic monitoring of the product development process searching for infringements exist</td>
</tr>
<tr>
<td></td>
<td>B. Tools: use of tools to determine code pedigree and license compliance</td>
</tr>
<tr>
<td>1</td>
<td>Manual searching for code segments, internally developed scanning tools</td>
</tr>
<tr>
<td>2</td>
<td>Automatic pedigree generation during product development</td>
</tr>
<tr>
<td>3</td>
<td>Automatic pedigree analysis, license compliance, and license compatibility of components of whole product</td>
</tr>
<tr>
<td>4</td>
<td>Pedigree analysis, license compliance, and license compatibility of components of whole product</td>
</tr>
<tr>
<td></td>
<td>C. Customer indemnification: company's efforts to reduce legal risk for customers</td>
</tr>
<tr>
<td>1</td>
<td>Maintenance contracts and specific usage guidelines for product</td>
</tr>
<tr>
<td>2</td>
<td>Replacement of infringing code by company</td>
</tr>
<tr>
<td>3</td>
<td>Limit indemnification to a fixed amount</td>
</tr>
<tr>
<td>4</td>
<td>Full indemnification for damages</td>
</tr>
</tbody>
</table>
Table 11 provides a list of articles from the literature review used to develop each clean IP level.

Table 11. Literature articles used to produce Version 1 of the clean IP model

<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Literature article</th>
</tr>
</thead>
</table>

4.2 Cases of IP infringement

Cases refer to IP infringement disputes between two parties that are resolved either before, during, or after a legal trial. Google searches for information on cases of IP infringement produced the following websites:

- Groklaw (http://www.groklaw.net)
- GPL Violations (http://gpl-violations.org)
- Software Freedom Law Center (SFLC) (http://www.softwarefreedom.org)
- Open Source Law Blog (http://www.oslawblog.com)
- Reg Developer (http://www.regdeveloper.co.uk)
- CNET News (http://www.news.com)
- ZDNet UK News (http://news.zdnet.co.uk/software)
A total of 25 cases were found and classified into two major categories: (i) open source license infringement, and (ii) other IP infringement. Table 12 lists the cases of open source license infringement and presents the conflicts and results of these cases. Table 13 lists the cases of other IP infringement and presents the conflicts and results of these cases.

### 4.2.1 Open source license infringement

Table 12 presents cases of open source license infringement and the parties involved. For each case, the type of open source conflict central to the case is identified as well as the court ruling. Appendix A.1 provides the following additional information for each case: detailed case summary, date, location, ruling/result, and unique implications.

<table>
<thead>
<tr>
<th>Case</th>
<th>Parties</th>
<th>Open source conflict(s)</th>
<th>Ruling/result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Welte (GPL Violations) v. Fortinet</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Court granted injunction, release of product source code by Fortinet</td>
</tr>
<tr>
<td>2</td>
<td>Welte (GPL Violations) v. Sitecom</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Court granted injunction, release of product source code and reimbursement of legal costs by Sitecom for trial</td>
</tr>
<tr>
<td>3</td>
<td>Welte (GPL Violations) v. D-Link</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Court granted injunction; release of product source code, reimbursement legal costs and product re-engineering costs by D-Link</td>
</tr>
<tr>
<td>4</td>
<td>Progress Software Corp v. MySQL</td>
<td>GPL license compliance (did not release source code)</td>
<td>Out of court settlement, details not disclosed</td>
</tr>
<tr>
<td>Case</td>
<td>License Compliance Details</td>
<td>Outcome/Resolution</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Welte (GPL Violations with FSF France) &amp; BusyBox v. Iliad</td>
<td>GPL license compliance (did not release source code for product containing Linux kernel and netfilter/iptables)</td>
<td>Pending result</td>
<td></td>
</tr>
<tr>
<td>Hemel (GPL Violations) v. Cisco</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Release of product source code by Cisco</td>
<td></td>
</tr>
<tr>
<td>Welte (GPL Violations) v. Skype</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code for product containing Linux kernel)</td>
<td>Both parties mutually agreed to minor changes in sales procedures, full details not disclosed</td>
<td></td>
</tr>
<tr>
<td>FSF v. Cisco (Linksys)</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Voluntary release of product source code by Cisco</td>
<td></td>
</tr>
<tr>
<td>SFLC (BusyBox) v. Verizon</td>
<td>GPL license compliance (did not release source code, did not release copy of GPL with code)</td>
<td>Out of court settlement: Verizon will publish source code, appoint open source compliance officer, notify customers of software rights</td>
<td></td>
</tr>
<tr>
<td>SFLC (BusyBox) v. Xterasys</td>
<td>GPL license compliance (see above case)</td>
<td>Out of court settlement; release of source code and financial compensation by Xterasys</td>
<td></td>
</tr>
<tr>
<td>SFLC (BusyBox) v. High-Gain Antennas</td>
<td>GPL license compliance (see above case)</td>
<td>Out of court settlement; release of source code and financial compensation by High-Gain Antennas</td>
<td></td>
</tr>
<tr>
<td>SFLC (BusyBox) v. Monsoon Multimedia</td>
<td>GPL license compliance (see above case)</td>
<td>Out of court settlement; release of source code and financial compensation by Monsoon Multimedia</td>
<td></td>
</tr>
<tr>
<td>Jacobsen v. Katzer</td>
<td>Artistic License compliance (copyright infringement), court interpretation of open source license under contract law</td>
<td>Court denied injunction since claim fell under contract law, ruling was overturned and deemed copyright infringement</td>
<td></td>
</tr>
<tr>
<td>Palmer et al. v. Sony BMG Music</td>
<td>LGPL license compliance (did not release source code by Sony)</td>
<td>Release of product source code by Sony</td>
<td></td>
</tr>
</tbody>
</table>
4.2.2 Other IP infringement

Table 13 identifies the cases of other IP infringement found and the parties involved. For each case, the type of conflict central to the case is identified as well as the court ruling. Appendix A.2 provides the following additional information for each case: detailed case summary, date, location, ruling/result, and unique implications.

<table>
<thead>
<tr>
<th>Case</th>
<th>Parties</th>
<th>Type of conflict</th>
<th>Ruling/result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SCO Group v. IBM</td>
<td>IP ownership (Unix copyright infringement)</td>
<td>Loss by SCO Group; Novell is legal owner of Unix, not SCO Group</td>
</tr>
<tr>
<td>2</td>
<td>SCO Group v. Novell</td>
<td>IP ownership (Unix copyright infringement)</td>
<td>Loss by SCO Group; Novell is legal owner of rights to code</td>
</tr>
<tr>
<td>3</td>
<td>Computer Associates v. Rocket Software</td>
<td>Misappropriation of trade secrets by engineers who work for Rocket Software</td>
<td>Pending result</td>
</tr>
<tr>
<td>4</td>
<td>ConnectU v. Facebook</td>
<td>Copyright infringement by founder of Facebook</td>
<td>Out of court settlement</td>
</tr>
<tr>
<td>5</td>
<td>Intel v. Shenzhen Donjin Communication Technology (SDCT)</td>
<td>Copyright infringement of SDCT technology</td>
<td>Out of court settlement</td>
</tr>
<tr>
<td>7</td>
<td>IP Innovation LLC &amp; Technology Licensing Corporation v. Red Hat &amp; Novell</td>
<td>Patent infringement by the top Linux sellers’ software products</td>
<td>Pending result, seeking damages and injunction</td>
</tr>
<tr>
<td>8</td>
<td>Network Appliance</td>
<td>Patent infringement when</td>
<td>Pending result, seeking</td>
</tr>
</tbody>
</table>

Table 13. Cases of other IP infringement
<table>
<thead>
<tr>
<th></th>
<th>(NetApp) v. Sun</th>
<th>Sun distributed as OS software containing NetApp IP</th>
<th>damages and injunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Alcatel-Lucent v. Microsoft</td>
<td>Patent infringement on MP3-related technologies</td>
<td>Alcatel-Lucent was awarded $1.52 billion in damages</td>
</tr>
<tr>
<td>10</td>
<td>Veritas v. Microsoft</td>
<td>Copyright infringement and misappropriation of trade secrets by Microsoft when it developed and incorporated Logical Volume Manager (LVM) into its operating system products</td>
<td>Pending result</td>
</tr>
</tbody>
</table>

4.3 **Version 2 of deliverables**

Version 2 of the categorization of methods, opportunities for entrepreneurs, and clean IP model is based on data extracted from the literature review and cases of IP infringement. Data from the cases was used to revise Version 1 of the deliverables.

4.3.1 **Categorization of methods**

The cases of IP infringement did not contribute revisions to Version 1 of the categorization of methods. Version 2 of the categorization of methods is identical to Version 1 as shown in Table 8.

4.3.2 **Opportunities for entrepreneurs**

Based on the cases of IP infringement, no new opportunities were identified; however, Opportunity 3 was modified using information drawn from two cases. Table 14 provides
a revised list of descriptions for the three opportunities along with the cases used to make the modification.

Table 14. Version 2: list of opportunities

<table>
<thead>
<tr>
<th>Opportunity type</th>
<th>Description</th>
<th>Case of IP infringement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional requirements</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1. Clean IP tool</td>
<td>Clean IP tool embedded in programming environment that supports real-time</td>
<td>FSF v. Cisco (Linksys)</td>
</tr>
<tr>
<td></td>
<td>collaboration and program forking</td>
<td></td>
</tr>
<tr>
<td>2. Clean IP tool</td>
<td>Clean IP tool embedded in programming environment that relies on private code</td>
<td>Progress Software Corp v. MySQL</td>
</tr>
<tr>
<td></td>
<td>database capability, as opposed to external open source databases</td>
<td></td>
</tr>
<tr>
<td>3. Automated clean IP</td>
<td>Automated clean IP tool that guarantees detection of unclean IP within a</td>
<td></td>
</tr>
<tr>
<td>tool</td>
<td>certain time</td>
<td></td>
</tr>
<tr>
<td>Clean IP services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Implementation of a</td>
<td>Implementation of a formalized process for introduction of code into software</td>
<td></td>
</tr>
<tr>
<td></td>
<td>formalized process for introduction of code into software development projects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to improve code approval workflows</td>
<td></td>
</tr>
<tr>
<td>5. Clean IP product</td>
<td>Clean IP product built on a collaborative platform that provides real-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>access to code from CVS and improve efficiency of code approval process</td>
<td></td>
</tr>
</tbody>
</table>

4.3.3 Clean IP model

Based on the cases of IP infringement, use of the three existing clean IP variables was strengthened and one new variable, code library, was added to the clean IP model. Table 15 presents Version 2 of the clean IP model with four variables and definitions of each variable. The levels are defined as in Version 1.
<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Clean IP level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>A. Preventative measures:</strong> what is done during product development to prevent legal conflicts</td>
<td>Policies for using third party and open source software exist</td>
</tr>
<tr>
<td><strong>B. Tools:</strong> use of tools to determine code pedigree and license compliance</td>
<td>Manual searching for code segments, internally developed scanning tools</td>
</tr>
<tr>
<td><strong>C. Customer indemnification:</strong> company's efforts to reduce legal risk for customers</td>
<td>Maintenance contracts and specific usage guidelines for product</td>
</tr>
<tr>
<td><strong>D. Code library:</strong> completeness of library and search capability</td>
<td>Local libraries exist</td>
</tr>
</tbody>
</table>
Table 16 provides a list of cases of IP infringement used to develop each clean IP level.

Table 16. Cases used to produce Version 2 of the clean IP model

<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Preventative measures</td>
<td>Welte v. Fortinet</td>
</tr>
<tr>
<td></td>
<td>Welte v. Sitecom</td>
</tr>
<tr>
<td></td>
<td>Welte v. D-Link</td>
</tr>
<tr>
<td></td>
<td>FSF v. Cisco (Linksys)</td>
</tr>
<tr>
<td>2 Tools</td>
<td>Welte v. Skype</td>
</tr>
<tr>
<td>3 Customer indemnification</td>
<td>IBM v. SCO Group</td>
</tr>
<tr>
<td></td>
<td>SCO Group v. Novell</td>
</tr>
<tr>
<td>4 Code library</td>
<td>Progress Software Corp v. MySQL</td>
</tr>
</tbody>
</table>

4.4 Version 3 of deliverables

Version 3 of the categorization of methods, opportunities for entrepreneurs, and clean IP model is based on data extracted from the literature review, cases of IP infringement, and questionnaire responses. Data from nine questionnaires was used to revise Version 2 of the deliverables. Appendix B: provides a copy of the questionnaire used in this research.

4.4.1 Categorization of methods

Based on the questionnaire responses, no new categories were extracted, but two new methods to ensure clean IP were identified. Table 17 provides Version 3 of the categorization of methods used to ensure clean IP with the source of data that supports each category.
Table 17. Version 3: categorization of methods

<table>
<thead>
<tr>
<th>Stage at which clean IP is addressed</th>
<th>Data source</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Literature</td>
<td>Promotional material</td>
<td>Questionnaires</td>
</tr>
<tr>
<td>1 Never</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>
| 2 During product development with periodic monitoring
  • Manual inspection
| 3 During product development with real-time monitoring | Protecode (2008) |              | No           |
| 5 After product development by external organization |              | Open Source Risk Management (2008) | No           |

4.4.2 Opportunities for entrepreneurs

Based on the questionnaire responses, six new opportunities were identified. Table 18 provides a revised list of descriptions for the 11 opportunities.

Table 18. Version 3: list of opportunities

<table>
<thead>
<tr>
<th>Opportunity type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Functional requirements | 1. Tool embedded in programming environment that supports real-time collaboration and program forking  
2. Tool embedded in programming environment that relies on private code database capability, as opposed to external open source databases  
3. Automated clean IP tool that guarantees detection of unclean IP within a certain time |
4. Ability to tag in-coming code with information about origin of code so that post-development analysis is more efficient

5. Automated identification of relevant open source licensing terms that are relevant to binary files as a result of include statements in header files and static/dynamic links used in source code

6. Tool to definitively establish origin and timeline of code

7. Tool that ties identified code back to specific built binaries and permit reporting and instrumentation of the resulting binaries with data that permits code identification

8. Tool that integrates with a database of licenses between company and IP vendors to permit management of dependencies such that (i) company could see at release whether the necessary licenses have been obtained prior to release, or (ii) so that company can see which prior projects are affected if company decides not to renew a licensing-in contract with an IP vendor

9. Implementation of a formalized process for introduction of code into software development projects to improve to code approval workflows

10. Clean IP product built on a collaborative platform that provides real-time access to code from concurrent versions system (CVS) and improve efficiency of code approval process

4.4.3 Clean IP model

Based on the questionnaire responses, one new variable was extracted and added to the clean IP model. Table 19 presents Version 3 of the clean IP model with five variables and definitions of each variable. The levels are defined the same as in Versions 1 and 2.
<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Clean IP level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Preventative measures: what is done during product development to prevent legal conflicts</td>
<td>Policies for using third party and open source software exist</td>
<td>Policies for using third party and open source software are enforced</td>
<td>Policies for using third party and open source software exist</td>
<td>Policies for using third party and open source software exist and periodic monitoring of the product development process for infringements exist</td>
<td>Policies for using third party and open source software exist and periodic monitoring of the product development process for infringements exist</td>
</tr>
<tr>
<td>B. Tools: use of tools to determine code pedigree and license compliance</td>
<td>Manual searching for code segments, internally developed scanning tools</td>
<td>Automatic pedigree generation during product development</td>
<td>Automatic pedigree analysis during product development</td>
<td>Automatic pedigree analysis, license compliance, and license compatibility of components of whole product</td>
<td>Automatic pedigree analysis, license compliance, and license compatibility of components of whole product</td>
</tr>
<tr>
<td>C. Customer indemnification: company's efforts to reduce legal risk for customers</td>
<td>Maintenance contracts and specific usage guidelines for product</td>
<td>Replacement of infringing code by company</td>
<td>Limit indemnification to a fixed amount</td>
<td>Full indemnification for damages</td>
<td>Full indemnification for damages</td>
</tr>
<tr>
<td>D. Code library: completeness of library and search capability</td>
<td>Local libraries exist</td>
<td>Central library with string search tools</td>
<td>Central library with fuzzy (pattern) matching against internal database exists</td>
<td>Central library with fuzzy (pattern) matching against internal database exists</td>
<td>Central library with fuzzy (pattern) matching against internal database exists</td>
</tr>
<tr>
<td>E. Outsourcing agreement: agreement between company and outsourcers regarding use of open source in components</td>
<td>Company relies on outsourcer assurance for license compliance</td>
<td>Company relies on outsourcer assurance for license compliance</td>
<td>Company relies on outsourcer assurance for license compliance</td>
<td>Company relies on outsourcer assurance for license compliance</td>
<td>Company relies on outsourcer assurance for license compliance</td>
</tr>
</tbody>
</table>
4.4.4 Placement of software projects in clean IP model

Seven of the nine questionnaire respondents placed their software projects at the different clean IP levels as shown in Table 20. The remaining questionnaire respondents from the sample were not able to do this assessment.

<table>
<thead>
<tr>
<th>Clean IP level</th>
<th>Number of software projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

4.5 Version 4 of deliverables

Version 4 of the categorization of methods, opportunities for entrepreneurs, and clean IP model is based on data extracted from the literature review, cases of IP infringement, questionnaire responses, and consultation with three experts in software innovation and IP. Data from the consultation with three experts was used to revise Version 3 of the deliverables.

4.5.1 Categorization of methods

Based on the consultation with three experts, one new category was extracted. The main distinguishing characteristic of the methods was the stage at which clean IP is addressed in a software development project. Table 21 provides Version 4 of the categorization of methods used to ensure clean IP with the source of data that supports each category. It is
important to note that the categories in Table 21 are not mutually exclusive. A software project may address clean IP at multiple stages throughout product development.

Table 21. Version 4: categorization of methods

<table>
<thead>
<tr>
<th>Stage at which clean IP is addressed</th>
<th>Data source</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Literature</td>
</tr>
<tr>
<td>1 Never</td>
<td>Black Duck Software (2007)</td>
</tr>
<tr>
<td>2 Project planning before product development</td>
<td>Golden (2007)</td>
</tr>
<tr>
<td>4 During product development with real-time monitoring</td>
<td>Protecode (2008)</td>
</tr>
<tr>
<td>6 After product development by external organization</td>
<td>Open Source Risk Management (2008)</td>
</tr>
</tbody>
</table>

4.5.2 Opportunities for entrepreneurs

Based on the consultation with three experts, one new opportunity was identified. Table 22 provides a revised list of descriptions for the 11 opportunities.
<table>
<thead>
<tr>
<th>Opportunity type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional requirements</td>
<td>1. Tool embedded in programming environment that supports real-time collaboration and program forking</td>
</tr>
<tr>
<td></td>
<td>2. Tool embedded in programming environment that relies on private code database capability, as opposed to external open source databases</td>
</tr>
<tr>
<td></td>
<td>3. Automated clean IP tool that guarantees detection of unclean IP within a certain time</td>
</tr>
<tr>
<td></td>
<td>4. Ability to tag in-coming code with information about origin of code so that post-development analysis is more efficient</td>
</tr>
<tr>
<td></td>
<td>5. Automated identification of relevant open source licensing terms that are relevant to binary files as a result of include statements in header files and static/dynamic links used in source code</td>
</tr>
<tr>
<td></td>
<td>6. Tool to definitively establish origin and timeline of code</td>
</tr>
<tr>
<td></td>
<td>7. Tool that ties identified code back to specific built binaries and permit reporting and instrumentation of the resulting binaries with data that permits code identification</td>
</tr>
<tr>
<td></td>
<td>8. Tool that integrates with a database of licenses between company and IP vendors to permit management of dependencies so that (i) company could see at release whether the necessary licenses have been obtained prior to release, or (ii) company could see which prior projects are affected if company decides not to renew a licensing-in contract with an IP vendor</td>
</tr>
<tr>
<td>Clean IP services</td>
<td>9. Implementation of a formalized process for introduction of code into software development projects to improve to code approval workflows</td>
</tr>
<tr>
<td></td>
<td>10. Clean IP product built on a collaborative platform that provides real-time access to code from concurrent versions system (CVS) and improve efficiency of code approval process</td>
</tr>
<tr>
<td></td>
<td>11. Operationalize the clean IP process by implementing a distributed service throughout the supply chain and parallel to quality assurance methods, as opposed to having a “checklist at a single door” every time a supplier contributes IP</td>
</tr>
</tbody>
</table>

4.5.3 **Clean IP model**

Based on the consultation with three experts, one new variable was added to the clean IP model. Table 23 presents Version 4 of the clean IP model with six variables and their definitions. The levels are defined as in Versions 1, 2, and 3.
Table 23. Version 4: clean IP model

<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Clean IP level 1</th>
<th>Clean IP level 2</th>
<th>Clean IP level 3</th>
<th>Clean IP level 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Preventative measures:</strong> what is done before and during product development to prevent legal conflicts</td>
<td>Policies for using third party and open source software exist</td>
<td>Policies for using third party and open source software are enforced</td>
<td>Company-wide standards exist and periodic monitoring of product development for infringements exist</td>
<td>Real-time access to documentation of code pedigree and licenses, real-time monitoring for infringements of company-wide standards</td>
</tr>
<tr>
<td><strong>B. License compliance tools:</strong> use of tools to determine code pedigree and license compliance</td>
<td>Manual searching for code strings, internally developed scanning tools</td>
<td>Bill of materials generation during product development</td>
<td>Automatic pedigree generation and analysis, license compliance checking</td>
<td>Pedigree analysis and bill of materials used to ensure license compatibility of components in whole product</td>
</tr>
<tr>
<td><strong>C. Customer indemnification:</strong> company’s efforts to reduce legal risk for customers</td>
<td>Maintenance contracts and specific usage guidelines for product</td>
<td>Limited indemnification to a fixed amount</td>
<td>Replacement of infringing code by company and limited indemnification</td>
<td>Full indemnification for damages</td>
</tr>
<tr>
<td><strong>D. Clean code library:</strong> completeness of pre-approved components library, search capability, associated metadata</td>
<td>Local libraries exist</td>
<td>Central library with search capability to a certain date or product versions</td>
<td>Central library capable of searching all externally developed modules</td>
<td>Central library with accompanying license and pedigree metadata for each component</td>
</tr>
<tr>
<td><strong>E. Outsourcing agreement:</strong> agreement between company and outsourcers regarding use of open source components</td>
<td>Company relies on internal management or legal department for clean IP approval</td>
<td>All incoming code is scanned continuously to ensure clean IP before used internally</td>
<td>Recognized third party provides clean IP reports periodically or at project milestones</td>
<td>Company requires license compliance reports and outsourcer assurance for clean IP</td>
</tr>
<tr>
<td><strong>F. Access to legacy code:</strong> organization of existing code and searching capability so that developers can access components</td>
<td>Manual string searches in Linux and Windows</td>
<td>String search capability in libraries and licenses</td>
<td>Limited internal knowledge database handling basic snippet and pattern matching</td>
<td>Internal knowledge database including all legacy code, handling complex pattern matching and dependency analysis</td>
</tr>
</tbody>
</table>
4.6 Observations

The following observations were made about each of the deliverables.

4.6.1 Categorization of methods

This research reveals that the distinguishing characteristic between methods used to ensure clean IP is the stage of software development at which clean IP is addressed. Clean IP is often addressed after the software is developed using a scanning tool by an organization inside the company.

Academic projects or open source tools to ensure clean are not widely used in commercial software development because of uncertain or incomplete results. With the increasing amount of open source code, third party code, and internal code, it is difficult for non-commercial suppliers of clean IP methods to develop methods that are recognized in the software development community.

4.6.2 Opportunities for entrepreneurs

Opportunities for entrepreneurs were classified into two groups: (i) functionality requirements, and (ii) services to companies that develop software which incorporate open source code, third party code, and internal code. Most opportunities identified in the literature focus on technical functionality and less on providing services to users. The
questionnaires and consultation with three experts identified opportunities anchored around services that ensure clean IP. These services could then be integrated into existing company processes.

4.6.3 Clean IP model

The clean IP model developed in this research consists of four levels defined by six variables. Clean IP level one contains the most projects from the sample of software development projects. Five of the seven projects placed in the model were located between levels one and two. This indicates that most software projects (i) do not prioritize ensuring clean IP in order to reach a higher level, or (ii) use very minimal methods to ensure clean IP. Placing these projects in the model proved difficult because questionnaire respondents could not accurately assess the level of clean IP in their software project. This was mainly due to their lack of knowledge of clean IP.
5 DISCUSSION

This chapter discusses the results of this research. Section 5.1 discusses the categorization of methods. Section 5.2 discusses the opportunities for entrepreneurs. Section 5.3 discusses the clean IP model. Section 5.4 discusses the legal cases of IP infringement. Section 5.5 uses cases of IP infringement to analyze the clean IP model.

5.1 Categorization of methods

The different sources of data provide complementary information on the methods companies use to provide clean IP. For example, the literature suggests that tools developed by Palamida and Open Source Risk Management (OSRM) are widely used tools. However, these methods were not found in the questionnaire responses or consultations with experts. The questionnaire responses suggested that manual processes and search scripts were used to ensure clean IP, but these methods were not found in the literature. The categorization of methods consists of three groups: (i) methods that were found only in the literature, (ii) methods that were found only in specific software companies, and (iii) methods that are documented in the literature and also used by software companies.

Most methods used to ensure clean IP address the issue of clean IP after product development using a tool internally to scan code. Leading companies in the clean IP space such as Black Duck Software and Palamida are developing products that can be
integrated into the software development life cycle. These tools validate code pedigree and license compliance so that license obligations and conflicts are recognized earlier in the development process. Implementing methods to ensure clean IP earlier the software development process is beneficial to companies that partner with many suppliers. These methods can be implemented by individual suppliers, resulting in clean IP being delivered to the integrator. By reducing the number of major suppliers and pushing the responsibility of ensuring clean IP further down the supply chain, integrators can make the clean IP process more efficient. Component integrators can also create clean IP policies for these suppliers to avoid unclean IP becoming embedded in products and therefore reduce product time to market. In the past, embedding unclean IP components has resulted in the involuntary release of product source code which diminishes company and product value. An example of this is shown in the notorious FSF v. Cisco case.

5.2 Opportunities for entrepreneurs
This research identifies two types of opportunities for entrepreneurs: (i) functionality requirements, and (ii) services to companies that develop software that incorporates open source code, third party code, and internal code. The latter type of opportunities supports clean IP as a formalized process in software development which did not exist before. The literature focuses mainly on functional opportunities while the questionnaire responses and expert interviews believe clean IP has more opportunity as a package of services that are integrated into existing processes.
Other software support processes may include maintaining security, code libraries, and product architecture. However, clean IP is currently viewed as a post-development process which can be expensive and time-consuming. New opportunities are also created by combining these two types of opportunities into a complete method that validates code and licenses during the development process as well as scans for compliance and compatibility requirements periodically after each build.

5.3 Clean IP model

High quality data for the clean IP model was obtained from the consultation with three experts. This model could be extended further if new methods used to ensure clean IP can fill the opportunities for entrepreneurs identified in this research. As the methods evolve, new clean IP concerns may arise that do not exist today.

Using the clean IP model may be difficult to encompass all variables relating to clean IP. For example, one expert from the consultation interviews suggested that the ability to provide remediation was important to assess the level of clean IP in a software project. Remediation refers to (i) the replacement or re-engineering of non-compliant code, or (ii) changing the list of approved components or licenses in a bill of materials, so that all code is license compliant and clean IP. Placing this variable in the clean IP model is difficult because it is contingent on several factors that may change throughout a software development project. Some of these include:

- Sensitivity of code
• Usage model for code
• Level of exposure
• Nature of policies

If the ability to remediate infringing code can be assessed, there is also the question of how to handle the infringement. This variable might be accommodated better with a different theoretical model.

5.4 Cases of IP infringement

The initial purpose of research cases of IP infringement was to determine the methods used by the parties involved as in the cases to ensure clean IP. These methods could then be classified into different categories based on characteristics of the cases. However, the cases of IP infringement did not discuss the use of methods to ensure clean IP. Instead, they revealed the source of conflict and details about the conflict.

This section discusses the link between cases and literature, and the results from the two groups of legal cases: (i) open source license infringement, and (ii) other IP infringement. It presents the similarities and differences between the two groups, as well as the types and roots of conflicts.
5.4.1 Link to economics and property rights literature

The cases of IP infringement have a direct link to the economics and property rights literature stream as shown in Section 2.2. Gans et al. (2002) and Teece (1988) state that company and product value depend on the protection of IP rights. The cases suggest that IP ownership must be certain before companies can capture value. In the cases of IP infringement, injunctions are granted against the infringing party to prevent them from capturing profits from unclean IP. This often results in the release of proprietary source code by the infringing party and they must find a different way to capture value. The value of goods (IP) depends on the allowed rights over the goods and the degree to which these rights are enforced (Coase, 1960; Demsetz, 1964; Furubotn & Pejovich, 1972). IP infringement cases show that courts are willing to enforce the IP rights over software.

The involvement of license enforcement organizations such as GPL Violations, Free Software Foundation (FSF), and Software Freedom Law Center (SFLC) play an important role regarding software license holders and the open source community. Property rights maintain economic and social relationships among individuals in the community (Costello & Costello, 2005). These groups preserve the economic and property rights of license holders by enforcing open source licenses in an area of law that lacks case law. This creates greater confidence in open source licenses by the community. Knowing IP protection will be enforced motivates community individuals to innovate and contribute (Simon, 1996). Developers will contribute to the open source community as well as capture private returns if there is incentive. The role of enforcement organizations
ensures that the balance of property rights is maintained between developers and the software community.

5.4.2 Open source license infringement

From the 25 cases that were collected, 15 cases involved open source license infringement. Among these 15 cases, 13 cases are GPL infringements along with single cases involving the LGPL and Artistic License infringement. This imbalance is largely due to the fact that the GPL is the most widely used open source license and has strict conditions in terms of derivative works and distribution.

Copyleft is often the major issue in these cases. The case of Progress Software Corp v. MySQL clarified the terms static linking, dynamic linking, and derivative work. Major conflicts also arose regarding use of the Linux kernel (Welte & BusyBox v. Iliad, Welte v. Skype) which are also the subject of ongoing copyright infringement trials.

The main cause of open source license infringement is the lack of correct methods used to ensure clean IP. Most of these conflicts result from lack of detection of open source code, third party code, and internal code embedded in software products. The level of cleanliness of a company's IP is debatable. It depends on many factors including the purpose of the software product and nature of users. Some companies believe their code is license compliant, but there is no standard or scale to measure this. As a result,
companies have different interpretations of what clean IP means and the processes/organizations needed to achieve an “acceptable” level. Some products are acquired through company merger or acquisition (M&A) deals. This chain of M&A can be lengthy and have legal repercussions for each company in the chain (FSF v. Cisco).

Organizations such as GPL Violations, Free Software Foundation (FSF), and Software Freedom Law Center (SFLC) actively seek companies that violate the GPL in both Europe and the US. This only adds to the number of cases. They pursue legal action against infringing companies which result in either court cases or out-of-court settlements.

The lack of case law in this area does not help license interpretation and enforceability by courts. However, ambiguities and other legal obstacles in open source licenses are being revised by the open source community. As a result, new versions are being developed to overcome problems in the software industry. With combined efforts from the open source community, license enforcement organizations, and the legal system, software developers will greater understand the legal implications of license compliance and clean IP.

5.4.3 Other IP infringement

The remaining ten cases of the 25 cases collected involve open source vendors and distributors as parties. Other IP mechanisms such as copyrights, patents, and trade secrets
are the subject of infringement cases. Use of the open source Linux kernel in proprietary software was a source of conflict for some major cases (SCO Group v. IBM, SCO Group v. Novell) while patent infringement by open source products was central to others (FireStar Software v. Red Hat (JBoss), Network Appliance (NetApp) v. Sun).

Researchers estimate that currently approximately 20,000 software-related patents are awarded each year in the US (de Laat, 2004). In addition, the number of opportunistic patent trolls who aggressively seek infringements is increasing (IP Innovation LLC & Technology Licensing Corporation v. Red Hat & Novell). This combination results in more litigation and financial repercussions for software companies who are often forced to settle out of court. Other cases are based on circumstantial evidence (ConnectU v. Facebook) so trials can last for years in order to discover rightful IP ownership and infringement. In fear of future litigation, indemnification is becoming more important. Companies such as Black Duck Software and Palamida are providing products and services to help companies ensure clean IP. These efforts will likely increase in order to reduce the amount of license infringement litigation in the future.

5.5 Analysis of clean IP model using cases

The legal cases of IP infringement from Section 4.2 were used to analyze the clean IP model. The details of each case were analyzed to determine the reasons that prevent the companies in each case from having clean IP. These reasons are represented in Table 24 as the limiting variable. For example, the case of Welte v. Sitecom is located at level 2
because the IP policies for using third party software and open source were not enforced (clean IP variable A), allowing GPL code to enter the software product. In this case, the GPL code was non-compliant with company policies which resulted in IP infringement. This limiting variable prevents the case from being placed at level 3.

**Table 24. Analysis of clean IP model using legal cases**

<table>
<thead>
<tr>
<th>Level</th>
<th>Case</th>
<th>Limiting variable from clean IP model</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Progress Software Corp v. MySQL</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>Welte v. Skype</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>FireStar Software v. Red Hat (JBoss)</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>IP Innovation LLC &amp; Technology Licensing Corporation v. Red Hat &amp; Novell</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>Network Appliance (NetApp) v. Sun</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>Veritas v. Microsoft</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>Welte v. Sitecom</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Welte (with FSF France) &amp; BusyBox v. Iliad</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SFLC (BusyBox) v. Verizon</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SFLC (BusyBox) v. Xterasys</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SFLC (BusyBox) v. High-Gain Antennas</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>SFLC (BusyBox) v. Monsoon Multimedia</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>Welte (GPL Violations) v. Fortinet</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Welte (GPL Violations) v. D-Link</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>Hemel (GPL Violations) v. Cisco</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>FSF v. Cisco (Linksys)</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td>FSF v. OpenTV</td>
<td>A, C</td>
</tr>
<tr>
<td></td>
<td>Jacobsen v. Katzer</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Alcatel-Lucent v. Microsoft</td>
<td>C</td>
</tr>
</tbody>
</table>

The following cases were not included in Table 24 because there are other factors unrelated to clean IP at the roots of these infringements:

- SCO Group v. IBM
- SCO Group v. Novell
- Computer Associates v. Rocket Software
- ConnectU v. Facebook
- Intel v. Shenzhen Donjin Communication Technology (SDCT)

Some of the possible conflicts in these cases leading to IP infringement are anchored around: definition of IP ownership, theft of IP, method of product distribution. Most of the cases in Table 24 are located at levels 1-3. They are often contingent on one variable that prevents them from being located at a higher level. Therefore, small changes to clean IP variables may raise the clean IP level.
6 CONCLUSIONS, LIMITATIONS, AND FUTURE RESEARCH

This chapter is organized into three sections. Section 6.1 provides the conclusions for this research. Section 6.2 describes the limitations of this research. Section 6.3 provides suggestions for future research in this domain.

6.1 Conclusions

Company and product value depends on the cleanliness of IP, not just the protection of IP. This research examines how companies ensure implementation of clean IP when their development projects incorporate open source code, third party code, and internal code. The research led to the following conclusions:

The distinguishing characteristic between methods is the stage of software development at which clean IP is addressed. By addressing clean IP earlier in the software development process, companies can avoid time-consuming and costly remediation after software development.

Opportunities for entrepreneurs can be classified into two groups: (i) functionality requirements, and (ii) services that allow software companies to integrate a formal clean IP process along with other software support processes.
The clean IP model developed in this research consists of four levels defined by six variables: preventative measures, license compliance tools, customer indemnification, clean code library, outsourcing agreement, and access to legacy code.

6.2 Limitations

This section outlines two limitations of this research. First, this research is focused on a sample of software development projects that were completed in a specific period of time. The definition of clean IP changes over time so the results of this research may also change over time. For example, the clean IP model may have new variables developed in the future that are not important today.

Second, the sample of software projects selected in the research was obtained from a single geographic location. Therefore, the scope of the clean IP model produced may not be completely applicable in other countries where IP laws differ from those in North America. There may be difficulty in generalizing the results across the entire software industry based on this sample.

6.3 Future research

This section suggests four opportunities for future research to further define clean IP and its business implications. First, anchor the clean IP model levels around actual software
development projects. Real data can be obtained from projects which are analyzed from start to completion. This data could be used to test the clean IP model.

Second, use a different theoretical abstraction to define clean IP such as:

- Capability Maturity Model (CMM) or Capability Maturity Model Integration (CMMI): framework for continuous process improvement, more explicit than the ISO standard in defining plan of action
- Team Software Process (TSP): framework for managing, tracking and reporting the team’s progress
- ISO 9001: framework specifying minimal acceptable quality level for software processes

Third, use different criteria for questionnaire participants to determine how clean IP methods vary across product or service diversity. The definition of clean IP may depend on the nature of product or service. For example, the definition of clean IP used by a shrink-wrap software provider may differ from the definition used by an open source software provider.

Fourth, examine how changes in product licensing for software development projects (i.e. proprietary to open source, proprietary to hybrid model) changes a company’s methods to ensure clean IP.
REFERENCES


APPENDICES

APPENDIX A: INVENTORY OF IP INFRINGEMENT CASES

A.1 Open source license infringement cases

Welte v. Fortinet

Party types: non-profit license enforcement, security software company

Conflict: GPL infringement

Summary: did not release full source code, did not release copy of GPL with code

Date: ruling April 2005

Location of court: Germany

Ruling, result: injunction, full release of source code

Unique implications: German courts will enforce GPL

Welte v. Sitecom

Party types: non-profit license enforcement, wireless router provider

Conflict: GPL infringement

Summary: did not release full source code, did not release copy of GPL with code

Date: ruling May 2004
Location of court: Germany

Ruling, result: injunction, full release of source code

Welte v. D-Link

Party types: non-profit license enforcement, network solutions provider

Conflict: GPL infringement

Summary: did not release full source code, did not release copy of GPL with code

Date: ruling September 2006

Location of court: Germany

Ruling, result: full release of source code, reimburse legal costs and re-engineering costs

Progress Software Corp v. MySQL

Party types: software vendor, open source database vendor

Conflict: GPL infringement
Summary: distributed a database product that linked directly to MySQL's code which had originally been released under the GPL; did not distribute source code for the database product

Date: ruling November 2002

Location of court: Boston, United States

Ruling, result: out of court settlement

Unique implications: interpretation and evaluation of "derivative work"

Welte (w/ FSF France) & BusyBox v. Iliad

Party types: non-profit license enforcement, DSL equipment provider

Conflict: GPL infringement

Summary: device called Freebox uses various GPL licensed software, including the Linux kernel and netfilter/iptables, but product source code was not released

Date: filed November 2007

Location of court: France

Ruling, result: pending

Unique implications: first case by GPL Violations in foreign court
Hemel v. Cisco

Party types: non-profit license enforcement, telecommunications company

Conflict: GPL infringement

Summary: did not release full source code, did not release copy of GPL with code

Date: early 2007

Location of court: United States

Ruling, result: full release of source code

Unique implications: product was reverse-engineered to discover infringement

Welte v. Skype

Party types: non-profit license enforcement, software development company

Conflict: GPL infringement

Summary: VoIP handset incorporates GPL-covered Linux kernel in its firmware; did not release full source code, did not release copy of GPL with code

Date: July 2007

Location of court: Germany

Ruling, result: details not released
Unique implications: organizations can be held liable for GPL violations even if they are simply distributing a product and do not manufacture it

FSF v. Cisco (Linksys)

Party types: non-profit license enforcement, telecommunications company

Conflict: GPL infringement

Summary: product was acquired through acquisition (Broadcom, Linksys), but unknowingly contained GPL code; did not release full source code

Date: filed 2003

Location of court: Boston, United States

Ruling, result: full release of source code

Unique implications: router can reportedly be re-engineered from $60 functionality to $600 functionality by modifying available source code

SFLC (BusyBox) v. Verizon

Party types: non-profit license enforcement, telecommunications company

Conflict: GPL infringement
Summary: Verizon wireless routers contain BusyBox software (Unix utilities) which contains GPL code, did not release copy of GPL with code

Date: March 2008

Location of court: United States

Ruling, result: Out of court settlement: Verizon will publish source code, appoint open source compliance officer, notify customers of software rights

SFLC (BusyBox) v. Xterasys

Party types: non-profit license enforcement, networking solutions provider

Conflict: GPL infringement

Summary: Xterasys networking products include BusyBox software (Unix utilities) which contains GPL code, did not release copy of GPL with code

Date: December 2007

Location of court: New York, United States

Ruling, result: out of court settlement; full release of source code, financial compensation to plaintiffs
SFLC (BusyBox) v. High-Gain Antennas

Party types: non-profit license enforcement, network hardware provider

Conflict: GPL infringement

Summary: HG-hardware includes BusyBox software (Unix utilities) which contains GPL code, did not release copy of GPL with code

Date: March 2008

Location of court: New York, United States

Ruling, result: out of court settlement; release of source code, financial compensation to plaintiffs

SFLC (BusyBox) v. Monsoon Multimedia

Party types: non-profit license enforcement, digital video software company

Conflict: GPL infringement

Summary: Monsoon Multimedia product includes BusyBox software (Unix utilities) which contains GPL code, did not release copy of GPL with code

Date: October 2007

Location of court: United States
Ruling, result: out of court settlement; full release of source code, financial compensation to plaintiffs

Jacobsen v. Katzer

Party types: scientist, private company (Kamind Associates)

Conflict: Artistic License infringement, copyright infringement

Summary: Katzer violated copyrights on JMRI Project decoder definition files by reproducing and redistributing versions of the software without including attribution required by open source license utilized by the JMRI Project

Date: August 2007, appeal decision August 2008

Location of court: San Francisco, United States

Ruling, result: injunction was denied since claim fell under contract law, not copyright law which does not provide injunctions; ruling was overturned and deemed copyright infringement

Unique implications: original ruling stated that open source license is a contract; attribution requirement was a condition of the contract, rather than a restriction on the scope of the license

Palmer et al v. Sony BMG Music Entertainment et al.
Party types: class action, music production label

Conflict: LGPL infringement

Summary: Sony Rootkit contained pieces of code identical to LAME, an open source mp3-encoder, licensed under the LGPL, did not release full source code

Date: December 2005

Location of court: New York, United States

Ruling, result: full release of source code

FSF v. OpenTV

Party types: non-profit license enforcement, interactive television software provider

Conflict: GPL infringement

Summary: FSF alleged that OpenTV was violating the GPL by failing to deliver source code when shipping a set-top box containing Linux

Date: February 2007

Location of court: California, United States

Ruling, result: court awarded $65 000 to FSF for damages and legal costs
A.2 Other IP infringement cases

SCO Group v. IBM

Party types: tech/consulting services provider, software vendor

Conflict: IP ownership (Unix copyright)

Summary: SCO claimed that IBM had, without authorization, contributed SCO’s IP to code base of OS Linux operating system

Date: August 2007

Location of court: Utah, United States

Ruling, result: in favour of IBM; Novell owns rights to code, not SCO

Unique implications: raises indemnification concerns for all software companies

SCO Group v. Novell

Party types: tech/consulting services, network operating systems company

Conflict: IP ownership (Unix copyright)

Summary: SCO claimed that they were owners of Unix, and that Novell owed them money for using Unix in their proprietary products

Date: August 2007
Location of court: Utah, United States

Ruling, result: Novell is owner of Unix, not SCO

Unique implications: ruling influences future Unix litigation

Computer Associates v. Rocket Software

Party types: software companies

Conflict: misappropriation of trade secrets

Summary: engineers stole confidential source code and used it at another company; CA asking for $200 million

Date: filed August 2007

Location of court: United States

Ruling, result: pending

ConnectU v. Facebook

Party types: software companies

Conflict: copyright infringement
Summary: ConnectU claims its founders hired Zuckerberg to write code for Harvard Connection, a social-networking site for Harvard; Zuckerberg instead stole their idea and technology and launched Facebook

Date: June 2008

Location of court: Boston, United States

Ruling, result: out of court settlement

Intel v. Shenzhen Donjin Communication Technology

Party types: semiconductor manufacturers

Conflict: copyright infringement

Summary: alleged copyright infringements on technology used for touch-pad telephone systems; settled a three-year legal battle that sued for $7.96 million

Date: May 2007

Location of court: United States

Ruling, result: out of court settlement

Unique implications: Shenzhen countersued Intel in 2005, alleging that Intel was operating as a monopoly
FireStar Software v. Red Hat (and JBoss)

Party types: enterprise software company, open source middleware software company

Conflict: patent infringement

Summary: FireStar Software claims that JBoss’ Hibernate 3.0 infringes on its patent for linking relational databases with object-oriented software

Date: June 2008

Location of court: Texas, United States

Ruling, result: out of court settlement

Unique implications: first time a patent infringement lawsuit has been filed against an open source distributor; Red Hat distributors, customers, and anyone else who uses Red Hat products are protected by settlement

IP Innovation LLC & Technology Licensing Corporation v. Red Hat & Novell

Party types: patent troll, open source vendors

Conflict: patent infringement
Summary: the top Linux sellers' software products infringe patent: "user interface with multiple workspaces for sharing display system objects," and two identically named patents

Date: filed October 2007

Location of court: Texas, United States

Ruling, result: pending (seeking damages and a permanent injunction prohibiting any further infringement)

Unique implications: first patent infringement lawsuit by patent trolls against FOSS vendors

Network Appliance (NetApp) v. Sun

Party types: network storage and data management company, open source vendor

Conflict: patent infringement

Summary: Sun infringed on seven NetApp patents for the Write Anywhere File Layout (WAFL) and RAID; NetApp claims that Sun infringed its patents when it released its Zettabyte File System (ZFS) in 2005 and allowed it to be distributed as OS software in 2007

Date: filed September 2007

Location of court: Texas, United States
Ruling, result: pending (seeking damages and injunction)

Unique implications: first patent lawsuit by a commercial competitor against a FOSS vendor

Alcatel-Lucent v. Microsoft

Party types: telecommunications provider, computer software provider

Conflict: patent infringement

Summary: Alcatel-Lucent claimed that Microsoft violated the '457 and '080 audio coding patents related to MPEG video technology, speech technology, internet technology, and other technologies. Alcatel-Lucent claimed that Microsoft's Windows Media Player infringed these patents by virtue of its MP3 capabilities

Date: February 2007

Location of court: California, United States

Ruling, result: jury awarded Alcatel-Lucent $1.52 billion in damages

Veritas v. Microsoft

Party types: storage management software provider, computer software provider
Conflict: copyright infringement, misappropriation of trade secrets

Summary: Copyright infringement and misappropriation of trade secrets by Microsoft when it developed and incorporated Logical Volume Manager (LVM) into its operating system products

Date: filed April 2008

Location of court: Washington, United States

Ruling, result: pending

Unique implications: Court observed only 54 lines of code, or 0.03% of a code base of almost 160,000 lines, had been identified as having been infringed. The Court noted that even where a relatively small quantity of code is copied, a finding of substantial similarity can still be made if the copied code is sufficiently important to the operation of the new program or gives the new program distinctive features or makes it more desirable.
APPENDIX B: QUESTIONNAIRE

Ensuring clean intellectual property in software projects

Researcher
Kamal Hassin
Master’s of Applied Science – Technology Innovation Management (TIM) Program
Department of Systems and Computer Engineering, Carleton University
khassin@connect.carleton.ca
Tel
Cell

Research supervisor
Dr. Tony Bailetti
Associate professor
Department of Systems and Computer Engineering, Carleton University
Bailetti@sce.carleton.ca
Tel 613-520-2600, ext 8398

Thank you for volunteering to participate in this research study. This questionnaire is part of my research leading to the completion of my thesis in the TIM program. The objective of this thesis is to examine how companies ensure clean intellectual property in development projects that incorporate third party software.

The data collected from this questionnaire will be used to:
• Develop a layered definition of clean intellectual property building on the clean room standards model
• Develop a categorization of methods used by software companies to ensure clean intellectual property
• Identify opportunities for entrepreneurs to profit from ensuring clean intellectual property

To participate in this study you must have:
• Participated in a software development project that started after January 1, 2007 and was completed prior to April 16, 2008
• Participated in the post-mortem meeting of this project

We ask that you:
1. Answer six questions
2. Examine a model that defines four levels of clean intellectual property shown as Table 1
3. Make suggestions on how to improve the model shown in Table 1

If you are willing to participate in a short follow-up interview and receive a copy of my thesis, please provide an email address at the bottom of page 2. The time and effort you take to participate in this research study is greatly appreciated. Thank you for your collaboration.

Sincerely,

Kamal Hassin
The following questions do not require disclosure of sensitive data. Please answer to the best of your knowledge:

Indicate the type of software project your answers are anchored around

Number of developers: ______________________

Type of software developed (e.g. custom, shrink wrapped, or other type): ______________________

Organization (centralized or distributed development): ______________________

The software project was completed on: Month ______ Year ______

1) What method was used to ensure that the intellectual property included in the software product was clean?

2) What software tools were used to ensure that the intellectual property included in the software product was clean?

3) What are the main features of the tool identified in 2) that attracted your company to use it?

4) What are the main features of the tool identified in 2) that would attract you to use it in another project?

5) What additional features would you like to see in the tool used to ensure clean intellectual property?

6) In which of the four levels in the model shown as Table 1 would you place your software project?

If I can contact you to follow up your answers, please print your name and email address.

Name: ______________________________________________________________

Email address: _______________________________________________________

<table>
<thead>
<tr>
<th>Clean IP variable</th>
<th>Clean IP level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Preventative measures:</strong></td>
<td>Policies for using ...</td>
<td>Policies for using third party and open source software exist</td>
<td>Policies for using third party and open source software are enforced</td>
<td>Company-wide standards exist and periodic monitoring of the product development process searching for infringements exist</td>
<td>Real-time access to documentation of code pedigree and licenses, real-time monitoring of infringements to company-wide standards</td>
</tr>
<tr>
<td>what is done during product ...</td>
<td>Local libraries exist</td>
<td>Central library with string search tools exist</td>
<td>Central library with fuzzy (pattern) matching against internal database exists</td>
<td>Central library capable of searching internally and externally developed modules</td>
<td></td>
</tr>
<tr>
<td>development to prevent legal ...</td>
<td>Manual searching for ...</td>
<td>Automatic pedigree generation during product development</td>
<td>Automatic pedigree analysis and license compliance checking</td>
<td>Pedigree analysis, license compliance, and license compatibility of components of whole product</td>
<td></td>
</tr>
<tr>
<td>conflicts</td>
<td>Maintenance contracts and specific usage guidelines for product</td>
<td>Replacement of infringing code by company</td>
<td>Limit indemnification to a fixed amount</td>
<td>Full indemnification for damages</td>
<td></td>
</tr>
<tr>
<td><strong>B. Code library:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>completeness of library and search capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C. Tools:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>use of tools to determine code pedigree and license compliance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>D. Customer indemnification:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>company’s efforts to reduce legal risk for customers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please:

1. Add new variables to those shown as A, B, C and D in the clean IP model

2. Add, delete, or modify the words used in the cells to define clean IP levels 1, 2, 3, and 4 using variables A, B, C and D