STRATEGIC ANALYSIS FOR THE OPEN SOURCE INSTITUTE AT SIMON FRASER UNIVERSITY

by

Massoud Sarrafi
B. Eng., Iran University of Science and Technology, 1993

and

Cristian-Marius Oneata
GDBA, Simon Fraser University, 2003
M. Sc. Eng., Polytechnic University of Bucharest, 1996

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF BUSINESS ADMINISTRATION

Management of Technology Program

In the
Faculty
of
Business Administration

SIMON FRASER UNIVERSITY

Fall 2004

APPROVAL

Names: **Massoud Sarrafi**

Cristian-Marius Oneata

Degree: **Master of Business Administration**

Title of the Project: **Strategic Analysis for the Open Source Institute at Simon Fraser University**

Supervisory Committee:

Michael Brydon, Ph. D
Senior Supervisor
Assistant Professor, Faculty of Business Administration

Richard Smith, Ph. D
Second Reader
Associate Professor, School of Communications

Date Approved: Dec. 7/04

SIMON FRASER UNIVERSITY

Partial Copyright Licence

ABSTRACT

Over the last decades, Open Source Software (OSS) has become increasingly popular and moved into the mainstream software industry. Our project's goal is to investigate the need for an Open Source Institute (OSI) within Simon Fraser University (SFU). We performed a literature review and interviewed IT specialists from various local companies. We identified some barriers to the adoption of OSS, including lack of working knowledge of OSS, lack of multi-tier technical support, and legal concerns. Our analysis confirmed the need for this institute to promote OSS.

We recommend that the SFU-OSI should broker technical support for OSS, to increase the level of awareness by organizing events and incubator-type initiatives, to offer various levels of training, to create standards and frameworks for user-friendly and well-documented OSS, to mine the undocumented innovations embedded in mature OSS, and to work with governments to leverage OSS for the betterment of society.

ACKNOWLEDGEMENTS

We want to express our special gratitude to our project supervisor, Dr. Michael Brydon, for his guidance and support. We benefited greatly from his insights, and this thesis is better for his judicious and exigent comments. As well, we would like to thank Dr. Richard Smith, whose feedback and suggestions were most appreciated.

Dr. Robert Cameron, the initiator of the proposed Open Source Institute at Simon Fraser University, provided valuable input and ideas, for which we are grateful.

We also want to thank all the participants in this research: Dr. David Ascher, Darryl Braaten, Fred Bremmer, Steve Munford, Dmitry Samosseiko, Gurusamy Sarathy, Victor Sira, Alan Tromba, Heather Watts, Nigel Webster, and Frank Wong. And finally, our thanks go to Cristina Calboreanu for her help with editing and revising.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

ANSI	American National Standards Institute
Backend software	Any software performing either the final stage in a process, or a task not apparent to the user.
CAD	Computer-Aided Design
CAM	Computer-Aided Machining
Client / Server	The relationship between two computer programs in which one program, the client, makes a service request from other program, the server, which fulfils the request.
Client or Desktop	(Used interchangeable in the paper) a computer or program that can download files for manipulation, run applications, or request application-based services from a server. Usually it runs on small computers, with limited processing power.
CRM	Customer Relationship Management
ERP	Enterprise Resource Planning
Freeware	Software that is available for free, usually over the Internet
GUI	Graphical User Interface
HTML	Hyper Text Mark-up Language
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
OSS	Open Source Software
POSIX	Portable Operating System Interface
Public domain software	Software that is not protected under patent or copyright.
R&D	Research and Development

Server	A program (loosely a computer) that is integrated in a network and that fulfils service requests from other components of the network. Usually, it runs on powerful computers and/or mainframes, with redundancy and increased reliability.
Shareware	Copyrighted software that is available free of charge on a trial basis, usually with the condition that users pay a fee for continued use and support.
W3C	World Wide Web Consortium
XML	Extensible Mark-up Language

1 INTRODUCTION

The goals of this project are to ascertain the current level of awareness and understanding of Open Source Software (OSS) in the business community in the Greater Vancouver area, identify the gaps between this community's needs and its knowledge of OSS (e.g., perceived risks and barriers to adoption of OSS), and recommend strategies that will enable a proposed Open Source Institute at Simon Fraser University (SFU) to contribute to the community by promoting effective and efficient use of OSS.

OSS, defined as software for which the source code is freely available for viewing, modifying, redistribution, and derivative work, is a growing area of research, and one that presents many challenges that contradict current economic, technical and motivation models. Thousands of programmers worldwide are contributing for free to the development of complex software products based on millions of lines of code. Why are they willingly disclosing code? How were they able to organize themselves, and coordinate complex projects based on the contribution of so many volunteer programmers around the world? Could one build a business model based on OSS, and if so, would this contradict the underlying principles of the OSS movement? Are there any prospects for future developments and widespread adoption? What could be learned from the developing model of OSS?

To answer some of these questions, Dr. Robert Cameron, a professor from Simon Fraser University's Faculty of Computing Science, came up with the idea of creating an Institute for OSS studies within SFU. The envisioned goal of this Institute would be to create a framework that enables research and other academic activities around OSS (teaching, conferences, etc.), to the benefit of industry, students, and other stakeholders.

In the following chapters, we use ideas and frameworks from OSS-focused literature, as well as the results of our own empirical research, to validate our recommendations for this potential institute. Our approach is based on interviews with senior IT specialists from the industry. After introduction, the paper will continue with a literature review in which we define the concepts and frameworks required for a better understanding of OSS. A presentation of the methodology and the results of the interviews will also follow. The analysis chapter will explain the findings and will be followed by recommendations and conclusions.

1.1 Identifying the gaps

We began our research by trying to find answers to a broad question: What can the Institute for OSS at Simon Fraser University do to reduce the gap between what is currently supplied in the industry and the industry's current and future needs regarding OSS? More specifically, we were interested in learning:

1. What is the current level of awareness in the IT community regarding the OSS phenomenon?
2. What are the perceived risks of running OSS in organizations, compared with running proprietary software?
3. What are the perceived barriers to adoption of OSS?
4. What are the differences between the adoption of server-side solutions, compared with desktop-side solutions (see glossary section for definitions)?
5. What are the perceived prospects of OSS? How does the IT community see the infrastructure OSS applications (Apache, Linux, MySQL, etc.) in the following years? Will desktop OSS applications take off any time soon?
6. What could the Open Source Institute do to promote the use of OSS? What is needed in the industry?

To answer these questions, we conducted an extensive review of OSS-related literature, as well as interviews with senior Information Technology (IT) professionals working in a wide

array of organizations, from leading software companies to telecommunications companies and educational institutions. Our results are presented in the next few chapters in detail.

The starting point of our research is the identification of the current status of the OSS movement. This is presented in the literature review section and is corroborated throughout our discussions with the interviewees. The following step is to identify how the Open Source Institute can respond to industry needs. This step is based on the interviews conducted with managers and senior IT personnel from a variety of organizations. Our analysis involved both server-side applications and desktop-side applications. The next chapter deals with diffusion of OSS issues, as well as the role of government with respect to OSS. In the end we present our findings and recommendations.

2 LITERATURE REVIEW

2.1 What is Open Source Software?

During recent years we have witnessed unprecedented press coverage of open source software. Examples of recent press coverage include articles related to Microsoft's Shared Source Initiative (Microsoft, 2004), and the legal dispute between the SCO Group and IBM, in which SCO alleged that IBM contributed SCO's intellectual property to the code of the Linux operating system without authorization. In contrast to proprietary or closed source software, open source software has its source code freely available for programmers to view, modify, redistribute, and use to create derivative work. The nature of software as a product and the unique characteristics of OSS create economic policy issues that will be presented later.

It is important to note that OSS is quite distinct from "shareware", "freeware," and "public domain software" (see glossary). Shareware is usually distributed in binary format, and users cannot access the source code. Open source software is also different from freeware and public domain software because open source is in fact copyrighted to ensure the source code remains available to the users (Bretthauer, 2002, p. 3). In other words, an open source license gives the user the right to use, modify, and redistribute the software on condition that the licensee grants the same kind of rights when the derivative work is redistributed. This concept is also known as copy-left (Mustonen, 2002, p. 101).

2.2 History of OSS

Open source software or, more precisely, the idea of shared source code, is not a recent phenomenon. During the 1960s and 1970s, academic institutions and some corporate research centres collaborated on various software initiatives. The UNIX operating system and C

programming language were originally developed in the AT&T Bell Labs and were shared at virtually no charge with other research institutions that contributed to their further development. Cooperation in developing software was commonplace, encouraged by a culture of sharing in which software was considered secondary to hardware. But, in the early 1980s, AT&T started to enforce its intellectual property rights related to UNIX. This move disrupted the community of developers, some of which started to think of means of continuing cooperative development by mitigating the threat of litigation from the enforcers of intellectual property rights (Lerner & Tirole, 2002, p. 200-201).

One landmark in the development of the OSS movement was the establishment of the Free Software Foundation by Richard Stallman in the early 1980s. A pioneer of the open source software movement, Stallman worked until the early '80s for the Massachusetts Institute of Technology Artificial Intelligence (AI) Laboratory, where sharing source code was common. Due to changes in the legal environment of collaborative software development and after being refused the source code of a faulty printer driver by Xerox, Stallman decided to develop a free UNIX-compatible operating system, which he called GNU[1]. He resigned from the MIT AI Lab and devoted his time to developing and distributing GNU one piece at a time. The Free Software Foundation (FSF), the non-profit organization started by Stallman, charged a small fee to copy and mail out its software to those interested. The proceeds were then used to hire programmers to develop and debug the rest of the GNU operating system. By 1991, FSF had most pieces of the GNU operating system, including the C libraries, but it had not developed an operating system kernel. The kernel came from another open source movement advocate, Linus Torvalds, who had released the kernel of his UNIX-like operating system, "Linux". The Linux kernel and all the accompanying GNU software were then put together to create the most widely known open source software, "GNU/Linux" operating system (Bretthauer, 2002, p. 5).

[1] GNU = "GNU's Not Unix" it is a recursive acronym created by the Free Software Foundation to emphasize the work of creating a freely distributable replacement for UNIX.

Stallman developed a licensing agreement, called GNU General Public License (GNU GPL), to ensure users access the source code as it evolves. During the 1980s, GPL was the most widely used open source licensing agreement. The license requires all software that is distributed with open source software to have the same distribution terms and therefore is considered one of the most stringent open source licenses (Johnson, 2002, p. 639). However, the GPL licensing model contained a major issue that precluded many companies from using OSS and their proprietary software in the same product: all the proprietary software bundled with OSS components had to have the same licensing model as GPL software – the so-called "viral infection" (Lerner & Tirole, 2002, p. 203). Because it was very restrictive, a number of individuals and organizations started to develop other licensing agreements, to encourage more participation and contribution to the open source movement. Some of these agreements allowed bundling open source software with closed source software. One such agreement was developed by Debian, a company that distributes Linux operating system (Lerner & Tirole, 2002, p. 202). The Open Source Initiative (OSI), a non-profit organization dedicated to the promotion of open source software, used the "Debian Free Software Guidelines" to define what constitutes open source software. OSI explicitly states that:

> "The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software" (Open Source Initiative, 2004).

Table 1 compares the classic GPL license and the new Open Source Definition as defined by the Open Source Initiative.

Table 1 - Comparison between Open Source Licensing Models[2]

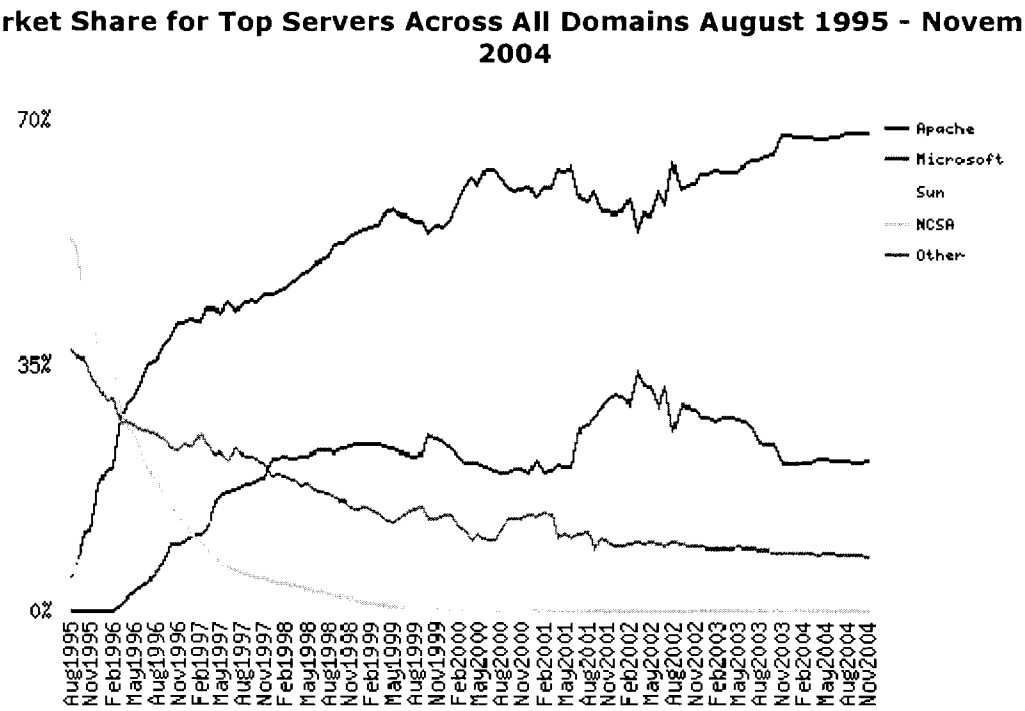
Characteristic	**General Public License (Free Software Foundation)**	**Open Source License**
Free redistribution	Software can be copied and given away (charge only for physical distribution)	Software can be sold or given away without paying royalties or other fees.
Source code	Should be distributed freely.	Must be freely available with the compiled form or from other sources.
Derived work	Derived work is allowed and has to be distributed within the same licensing terms.	Derived work is allowed and has to be distributed within the same licensing terms.
Restrictions on other software	Restrictive licensing model. Any software built or derived from using a part or the entire GPL licensed code has to be distributed as a whole under the GPL terms. Exception: the independent components that were not developed or derived from GPL licensed software.	More permissive licensing model. Other software bundled with OSS can use open or closed source licensing models.

The ability to view, collaborate, modify, and redistribute source code has allowed thousands of developers to contribute to the development of OSS. The contributions of this mass of talent have immensely improved the quality of open source software to the point that some open source software matches and even exceeds popular closed source software in terms of stability, reliability, inter-operability, and security. The combination of technical merits of some open source software and low acquisition costs has made certain open source software extremely popular. Sendmail (Mail Transfer Agent), Linux, Perl (Scripting language), Apache (Web server) and BIND (**B**erkeley **I**nternet **N**ame **D**aemon – a domain name server) are all examples of open

[2] Adapted from Free Software Foundation (1991) and Open Source Definition (2004)

source server software with significant success compared to closed source software. For instance, according to an October 2004 survey from Internet watchdog Netcraft.com (a survey of more than 55 million sites worldwide), the Apache web server dominates the web server market with almost 68 per cent of the global market, followed by closed source software such as Microsoft's Internet Information Services (IIS) with about 21 per cent and Sun's software with 3 per cent (Netcraft, 2004). Lerner and Tirole estimate that Sendmail handled about 75 per cent of all Internet traffic in 2000 (Lerner & Tirole, 2002, p. 212). On desktop applications, OpenOffice (Office productivity suit) and Mozilla-based web browsers are becoming increasingly more popular (Wheeler, 2004).

Figure 1 - Apache Web Server Domination[3]



[3] © Netcraft, 2004 – by permission for fair use.

2.3 Comparison between Open and Closed Source Software

To better understand the following chapters, a comparison between open and closed source software is required. The OSS licensing model (freedom of use, allowing for modifications and redistributions) is in obvious contrast to closed source software (restricting the use to licensed users and forbidding modifications and redistributions). The following attributes are also different:

1. **Control**: In the case of OSS, consumers and developers have equal access, although there is usually a core group that centralizes, maintains and creates new releases. The leaders of the Open Source projects emerge informally and are recognized by the OSS community (Lerner & Tirole, 2002, p. 221). The average user is not usually in a position to influence the direction of future developments, but users can contribute code, participate in discussions and customize the software to fit their needs. In the case of proprietary software, the developing companies are in control. They create new versions and receive royalties and fees. The customers, depending upon their size and relative power, are sometimes able to influence the direction of software development, but in most cases do not. For instance, when an important customer requires some particular software features, it can use contractual leverage to determine the vendor to supply the additional features.
2. **Development model:** OSS is developed by heterogeneous groups of unpaid or paid developers who act within decentralized networks and benefit from the variety and diversity of members' backgrounds and ideas. The informal characteristic of the network is instrumental in organically allocating the best resources to solve the relevant issues. Members volunteer to solve those issues they think they can solve, and the best solution is usually chosen through a defined mechanism. However, less-than-interesting features (such as help menus, documentation, etc.) are usually left behind. Proprietary software is developed within the organization by a limited number of programmers, organized in groups and with specific goals to attain. They benefit from standard market research and are told what features the software should have. Most of the features important for users (help menus, documentation, etc.) are planned and then developed.

Management is formal and the programmers receive some form of compensation (money, dividends, stock). The software companies are interested in conserving the revenue model and preserving their competitive advantage.

3. **Innovation:** In the case of OSS, large groups of developers create and test software simultaneously. Because of the large number of programmers, the odds of finding better, more innovative solutions are higher. The innovation is "distributed" (Kogut & Metiu, 2001, p. 248). In the case of proprietary software, the innovative process is usually driven by R&D (see glossary) departments. Enforcing property rights helps obtaining sufficient resources to fund R&D activities. The innovative capacity is limited to smaller groups of programmers inside or outside companies.
4. **Acquisition and expansion costs:** In the case of OSS, there are low or no initial costs, as well as no additional costs for additional users (e-Cology, 2003, p. 7). Upfront licensing costs could be very high for proprietary software, and the licensing model is usually linked with the number of users or connections.
5. **Total cost of ownership (TCO):** It is difficult to compare total cost of ownership for OSS and proprietary software. There are examples of contextual advantages on both sides. More research is needed in this area, for conclusive results.
6. **Longevity:** OSS exists as long as it serves a useful purpose (R-smart group, 2004, p. 4). However, there is a non-negligible risk of the OSS project being abandoned, in which case the consumer is stuck with an obsolete and/or unsupported solution. This risk is considerably smaller for mature projects. Proprietary software is viable as long as the developer exists and does not discontinue its products. If the developer goes out of business or is acquired by other company, the customer is left with unsupported software for which the source code is unavailable.

2.4 Benefits of OSS

OSS is controversial and most studies have been conducted by stakeholders with conflicting interests. However, there are two comprehensive government-funded empirical studies on usage, benefits, the current and the future state of OSS. The first study was funded by

the European Commission and was conducted by researchers from the University of Maastricht in The Netherlands. This study is called "Free/Libre and Open Source Software" and its results were published as the "FLOSS Report". The FLOSS report is based on extensive interviews with IT decision makers in 1,452 organizations (each employing over 100 people), in Germany, UK and Sweden. 395 of these companies used Open Source Software in one form or another (Wichmann, 2002, p. 11). The second study was funded by the government of Canada and was conducted by the e-Cology Corporation of Toronto, Ontario. This report is known as "Open Source Software in Canada" or the "canFLOSS" report. CanFLOSS is based on 180 questionnaires, 19 in-depth interviews, an industry review, and an extensive literature review (e-Cology, 2003, p. 10). As we will see in the next two sub-sections the results of these reports are congruent.

2.4.1 Social Benefits

OSS offers an alternative to proprietary solutions, therefore creating competition and stimulating innovation. As a result, society benefits from better solutions and reduced costs (e-Cology, 2003, p. 21), increasing the social welfare. Using OSS in the public sector prevents lock-in situations, in which the organization has to increase spending to be able to continue to use the proprietary software from certain developers (mandatory upgrades, discontinued support for older versions, etc.). In the canFLOSS study, 88 per cent of respondents strongly agreed and 9 per cent agreed that the government procurement policy should include the option of open source solutions. OSS could be seen as "a form of market correction" (e-Cology, 2003, p. 65).

For developing countries, the use of OSS is a means of obtaining good value by overcoming legal and economic barriers of proprietary software (e-Cology, 2003, p. 44). Local solutions based on OSS architecture could be developed and used to the benefit of their societies. This translates into strategic benefits for those countries.

2.4.2 Technical and Monetary Benefits

Monetary benefits are believed to be the key business drivers for the adoption of OSS. The European FLOSS report found "Independence from pricing and licensing policies of big software companies" to be the strongest reason for using OSS (Wichmann, 2002, p. 29). This finding is consistent with the Canadian FLOSS project, which reported "cost reduction and vendor independence" as the major business drivers for using OSS in Canada (e-Cology, 2003, p. 21).

The primary benefits of using OSS are a combination of monetary and technical merits. The majority (over 90 per cent) of respondents in the Canadian FLOSS study expressed "agreement" or "strong agreement" that OSS is a good fit for the current IT infrastructure (e-Cology, 2003, p. 21). The survey found "Greater security" to be the third greatest benefit. CanFLOSS found "Cost reduction" and "Greater flexibility" to be the main benefits of using OSS in Canada, followed by "Greater security", "Improved productivity," and "Improved competitiveness" (e-Cology, 2003, p. 22). The European FLOSS report found "Higher stability and access protection" to be the top benefit of using OSS, followed by "Direct cost savings", "Indirect cost savings," and "Open and modifiable source code" (Wichmann, 2002, p. 47).

2.5 Community Motivation

There are a number of ways to explain what motivates a considerable number of individuals scattered all over the world to collaborate and develop open source software. The phenomenon is not fully understood, but a number of attempts have been made to clarify the motivational factors.

First, it is recognized that in the proprietary software industry, as in any economic domain, the main motivator is profit maximization, which is usually achieved by securing intellectual property rights (Mustonen, 2002, p. 103). So, what are the motivators for the OSS

developer? Researchers and analysts have found that open source developers derive at least two benefits from their work. First, they improve their long-term career prospects by signalling their skills, networking with their peers and raising their profile in the software development arena. Second, it is believed they enjoy some ego-gratification[4] as a result of the peer recognition they receive (Lerner & Tirole, 2002, p. 213).

A second insight into community motivation comes from examining the types of applications that the open source community tends to develop. It is important to note that although the open source community has built industrial-strength complex backend software (see glossary), it has failed to develop desktop applications of similar quality. For instance, OpenOffice, the OSS competitor of MS Office, is still in its infancy, though according to the OpenOffice.org Web site about 16 million downloads were recorded from their site. There is evidence that open source developers are motivated to develop what is useful in their work (Johnson, 2002, p. 639). Hence the success of Sendmail, Apache, Linux, CVS, Perl, and similar open source software that is widely used by system administrators, web-masters, and programmers, mainly in infrastructure applications.

A third source of community motivation is offered by the proponents of open source who claim altruistic values drive the open source movement. Due to its nature, OSS is equally and freely accessible by all people, poor as well as rich, small companies and multi-nationals, third world countries and industrialized nations. Politics and economic sanctions cannot stop less favoured nations from using open source software. Furthermore, provisions five and six of the official open source definition explicitly prohibit discrimination against any people, group or field of endeavour (Open Source Initiative, 2004). However, Lerner and Tirole discount altruism as a

[4] The term "ego-gratification" refers to the fact that people want to be perceived by their peers as being knowledgeable, influential, and prominent. Within large communities of developers, the ability to resolve software issues confers people this central role.

driver of open source movement. They argue that while OSS is accessible by the poor, many beneficiaries of OSS are rich people or Fortune 500 companies (Lerner & Tirole, 2002, p. 198).

2.6 Various Economic Perspectives on OSS

Many economic researchers have studied the IT industry extensively in the last decade, giving particular attention to its software development component. Names such as Lerner, Tirole, Fichman, Kemerer, Christensen, are only a few from a large group of academics that have closely studied software development and OSS. Open Source Software poses a different set of issues with respect to business models than proprietary software, because of the different characteristics (such as licensing models, modular architecture, etc.) and development models. In this chapter we present some general economic issues around the IT industry, with the purpose of preparing the foundation of the following analysis.

2.6.1 The Provision of Public Goods and OSS

In economic theory, a good is said to be a "public good" if it is "non-rival" and "non-excludable". "Non-rival" means that consumption of the good by one party does not preclude another party from using it, so the amount of good available for consumption does not vary with the number of consumers of it (Baye, 2002, p. 513). Since the marginal cost of duplicating software is almost zero and running a particular application on one machine does not stop others from running a separate copy of it on a different machine, software (both proprietary and open source) is by nature a non-rival good. "Non-excludable" means that no one is precluded from using that good, so anyone can consume it once it is available. Here, a clear distinction between proprietary and open source software is apparent. Traditionally, closed source software companies have tried to make software an excludable good by enforcing intellectual property rights. These copyright agreements have been designed to allow licensees to access a binary copy of the program and prohibit anyone except the copyright holder from accessing the source code.

Conversely, the licensing schemes used in open source software have been designed to ensure that the public can freely access the source code.

It is common knowledge in the economist community that the market provides public goods in inefficient quantities. People have few incentives to purchase a public good because it is freely supplied, so they prefer others to pay for the goods (or to produce them) – the so-called "free riding" problem (Baye, 2002, p. 513). Thus, public goods tend to be undersupplied because too few want to produce them. This is known as "market failure". When instances of market failure arise, governments often intervene to correct the market. The role of governments will be detailed later.

The proposed Open Source Institute within SFU could have a major role in correcting these instances of market failure for the public good that is OSS. As a component of a public institution, together with the governments (local and federal), the institute may fill in the gaps left by the market. Our null hypothesis is that there are instances of market failure in the OSS space that could be addressed by the Institute. Through our research, we intend to determine whether market failure really exists and, if it does, whether there is a potential role for a university-affiliated Institute to fill the gaps.

2.6.2 Risks of Investments in Information Systems

One of the initial objectives of the research is to identify the perceived risks involved in running OSS. Our interests are twofold: to identify specific risks and to compare them with the risks of running new proprietary software in general. Before presenting the findings from our research, it is proper to review the current understanding regarding the risks involved in IT projects.

Generally, the main risks involved in managing information systems are (Clemons, 1991, p. 31):

1. **Financial risk:** the risk of not affording a suitable technical solution because of a failed cost-benefit analysis;
2. **Technical risk:** the technology cannot be properly managed and used because it is simply not available, or the knowledge and support system are missing;
3. **Project risk:** the company cannot do it because of the complexity, lack and technical or human resources, lack of skills or the organizational culture;
4. **Functionality risk:** even if the project is completed, the functionality of the system does not satisfy users' requirements, so the benefits are lower than expected;
5. **Systemic risk:** the environment is so dynamic that the system, even if implemented successfully, becomes obsolete because of change in the external medium.

The software-specific risks could be included in one of the above broad categories of risks. In the following sections, we present an analysis of the perceived risks of OSS compared with proprietary software.

2.6.3 Standards and Competitive Strategies

The development of standards was one of the key factors that contributed to the success of the industrial revolution and the commoditization of many goods and services. Without standards, automobiles would not be affordable for so many people, only a select few would be able to afford the cost of an airplane ticket, and household appliances would be out of the reach of the majority of people. In the software industry, standards are vital to ensure not only compatibility between diverse solutions and/or components, but also to the economic success or failure of the company. Many of the OSS solutions that are currently mature and widespread were developed around open standards (as presented in chapter 2.2). Before discussing the research subjects' perceptions regarding standards and how they think OSS could be leveraged into competitive strategies, a presentation of current ideas around standards and business strategies in the IT industry is necessary. In the following paragraphs, we present three types of approaches

with respect to creating strategies around standards: ***promoting proprietary standards to the industry standard level***, ***adopting established strategies*** (in the case of companies unable to impose their own standards), and ***developing strategies around OSS***.

Firstly, a company wishes for its technology to become the industry standard. This would allow the company to extract rents[5] and become a leader of the industry. This is the case of Microsoft and Intel – the so-called "Wintel" standard (Hill, 1997, p. 8), whose technologies, Windows operating system based on Intel microprocessors, define industry standards. In the case of Wintel, a virtuous cycle was created when a larger installed base led to greater software availability over time, which in turn created increased value for the hardware. This was a self-reinforcing cycle that eventually created customer lock-in in the market, and the technology became a "de facto" standard. The economists framed this phenomenon as "increasing returns" (Hill, 1997, p. 9). Other companies, such as IBM with its PC architecture, were able to promote their technology to the industry standard level, but were unable to appropriate the benefits.

To attain this privileged position the following strategies are instrumental (Hill, 1997, p. 18):

1. Aggressive initial licensing for a wide initial distribution and building the market expectation;
2. Strategic alliances, especially with complementary product vendors, but also competitors. Again, bear in mind Microsoft's aggressive strategy of increasing the value of its operating system by creating partnerships with other software developers and increasing the degree of software integration;
3. Product differentiation by increasing the supply of complementary products. An example is Adobe's strategy of giving away its "PDF (Portable Document Format)" file format reader, with a view to attaining the critical mass necessary to become the industry standard (in Adobe's case, the goal was to be able to obtain rents from its "PDF" writer);

[5] Supernormal profits

4. Aggressive positioning using penetration prices and wide distribution for an initially accelerated adoption. For instance, educational licenses at discounted prices are instrumental in accelerating adoption.

The key factors that impact the above strategies are raising the barriers to imitation through patent and copyright protection and availability of the complementary products (Hill, 1997, p. 19-21). Some commercial software companies, such as Microsoft, Adobe, and Oracle, were able to actively pursue the above-presented strategies and enforce copyright protection, and obtain market domination. In the OSS domain, however, the so-called "copyleft licensing" model cannot create barriers to imitation. Thus, other strategies have to be used to create a viable business model for companies.

Secondly, other companies in the industry, which were not able to promote their technology to the level of industry standards, use copyright protection to maintain a competitive advantage for their products. By excluding users from using their product unless they pay for the license and forbidding the modification and redistribution, they control the customers and could be competitive. Their best competitive stance is based on customer solutions (creating a wide variety of products and services that satisfy most of the customers' needs) or lower costs/ differentiated products – having the cheapest solutions with a lower level of features or having high featured products for which higher prices could be charged (Hax & Wilde, 1999, p. 12).

Thirdly, some recently emerged strategies built around OSS allow companies to compete in the industry. Even if the founders of most OSS projects did not think about immediate monetary profits and building business models, some companies have recently embraced business models built around OSS. The OSS strategies are (Konig, 2004):

1. **Optimisation:** Because of the modularised design, one layer of the software stack[6] is "comformable" (i.e. Linux), allowing other modules to be optimised.

[6] Software is made up of components with diverse roles. Those components have basic roles (for instance the operating system, drivers etc) or specialized roles (for instance a spreadsheet application). In computer jargon, the components are seen as a hierarchy, often called stack.

The business model exploits the optimised modules. (Christensen, Anthony & Roth, 2004, p. 19);

2. **Dual Licensing:** A strategy in which a company offers free use of its software with some limitations (under a GPL-like license) and at the same time a commercial distribution with a bigger set of features. The company benefits from the advantages of OSS, such as better development and bug fixes, faster adoption, as well as the advantages of commercial software (licensing fees). For instance, MySQL AB's business model of offering its database core module free under a GPL-type license and also supplying a commercial version with improved tools and with documentation and support. The commercial version is bought by important customers such as Sabre (the first airline ticketing system in the world, whose over 100 servers run MySQL, with plans for another 200 in the following years), Nokia, Siemens, etc.;
3. **Consulting:** Consulting for implementation and other IT services;
4. **Subscription:** Business model embraced by many OSS companies: assuring support and maintenance as well as updates. (Red Hat, SuSE distributors);
5. **Patronage:** It is a competitive strategy in which a company contributes to an OSS project to drive a standard to widespread adoption, to crack existing markets, to increase the selling of complementary products or to commoditize an existing layer of the software stack. (i.e., IBM supports Linux for complementarity reasons, to increase hardware sales, and also as a means of competing with Microsoft's and Sun's server operating systems);
6. **Hosted:** Renting a software application instead of buying it. The trend is towards hosted applications, benefiting from the expansion of the Internet infrastructure and grid networks;
7. **Embedded:** OSS is increasingly used in devices and appliances such as TiVO; the main advantage is that it is free and can be customized by the company to fit its purpose.

2.6.4 Adoption and Paradigm Shift

Tim O'Reilly, a well-known advocate of open source software, sees the OSS movement as the expression of three long-term trends (O'Reilly, 2004):

1. Commoditization of software, being pushed by standards and Internet and communication systems;
2. Network-enabled collaboration: the explosion of the Internet in the last decade has made collaboration in OSS development easier;
3. Software customisability and software as a service: users rent software customized for their particular needs.

The software industry has had to adapt and respond to the ever-increasing competition from mature OSS solutions available at far cheaper prices than proprietary software. There are areas where commercial software companies cannot charge their usual prices anymore because of mature OSS alternatives: Apache web server versus proprietary software, Linux as an alternative to Unix and Windows systems are examples. O'Reilly calls this phenomenon commoditization of software (e-Cology, 2003, p. 6). He has compared this commoditization trend with the era when IBM introduced the standardized architecture of the PC (1981) and allowed others to use the design, a major shift in the industry's practice that opened up the era of computer clones and cheap affordable home computers.

Network-enabled collaboration has further fostered the initial development of OSS. Virtual Internet communities that collaborate on OSS projects are increasingly numerous: SourceForge.com recently counted about 63,000 ongoing OSS projects (SourceForge.com, 2004). Sites such as SourceForge.com act as portals for OSS virtual communities, whose members exchange opinions, files, and software components, and use different mechanisms to choose, from a variety of solutions, those solutions that will be inserted in the core code.

The current trend of outsourcing IT capabilities to hosting companies or renting software applications from software service suppliers is encouraged by the explosion of web-based OSS

solutions such as Linux, Apache, etc., dynamic languages such as Perl, Python, or PHP, and multi-tier architectures. Vendors like IBM or HP refer to this trend as "computing on demand" or "pervasive computing" (O'Reilly, 2004).

O'Reilly frames the trends as a "Paradigm Shift", and he predicts that these changes are happening and will continue to happen in the near future. As radical changes occur in science and technology, it is not uncommon for OSS to represent a major impetus behind a new paradigm shift. O'Reilly suggests that a new field of scientific and economic inquiry is created, that deserves to be pursued.

2.6.5 Diffusion of New Technologies

Diffusion of new technologies is a major area of study for many researchers. We use the framework to comment later about OSS prospects. According to Rogers (Rogers, 1983), five attributes of innovation influence the rate of adoption of new technologies, and these are:

1. **Relative advantage:** technical superiority against the old technology (in terms of costs, functionality, image etc.);
2. **Compatibility:** the extent to which the new technology is compatible with the current systems, norms and values;
3. **Complexity:** the difficulty of learning the new solution;
4. **Trialability:** the easiness of trying the new solution;
5. **Observability:** whether the results can easily be communicated to other interested parties.

While these attributes partially explain the diffusion of technology, for the software industry they are insufficient. They do not explain the increasing return to adoption phenomenon that is the characteristic of industries with network effects, where the benefits of adoption are increasing with the size of the community of adopters. For the software industry, economists have identified three factors that explain the increasing returns to adoption: learning by using, positive

network externalities, and technological interrelatedness (Fichman & Kemerer, 1993, p. 10). Learning by using means that the community of adopters is expanding as the users and vendors learn and accumulate experience regarding the software. Technological interrelatedness means that a technology becomes worthwhile as a whole when there is a large base of comparable products, while positive network externalities appears when the value of a product is increasing as the number of users increase.

Fichman and Kemerer identified another four economic factors that are affecting the adoption of technology: ***prior technology drag***, ***the irreversibility of investments***, ***sponsorship***, and ***expectations***. A large and mature installed base could create fewer incentives to adopt a new technology, and also, adoption requires investments that are irreversible and that can generate resistance to change. The existence of clear and powerful sponsors is helping the adoption speed while high expectations could act as a catalyst of the adoption process (Fichman & Kemerer, 1993, p. 11).

Fichman and Kemerer have identified some diffusion factors that influence the new technologies' speed of adoption in general. Do these factors apply to OSS? What else could influence the diffusion? In the following chapters the results of investigation and interviews will give a better perspective about the future of OSS.

2.6.6 Resistance to Change

The rapid pace of scientific and economic evolution has made change a constant of modern life. Resistance to change is a major problem in many organizations and is manifested at every level within organizations. The implementation of new software systems and solutions is almost always confronted with resistance to change, and OSS is not different from proprietary software in this regard.

At the organizational level the factors that contribute to resistance to change are power and politics, differences in departmental orientation, "mechanistic structures" (tall, centralized organizational structures[7]) and the organizational culture (George & Jones, 2002, p. 650). Because change involves benefits for some people, functions or departments at the expense of others, it is likely that those that do not benefit will resist it and will engage in political actions. Moreover, differences in the opinions and interests of various departments could create organizational inertia because time and effort have to be spent to convince reluctant departments to align to change. Tall, centralized organizational structures that emphasize standardisation of behaviours also have difficulties adapting to change because they are not flexible enough. Finally, organizational culture plays a heavy role in accepting change, especially when change disrupts widely accepted values and norms.

There are factors that could contribute to resistance to change at the group level, such as group norms, cohesiveness, and groupthink (George & Jones, 2002, p. 651). A high level of group cohesiveness and the attractiveness of the group to its members could create resistance if change threatens to disturb the group as a whole. Groupthink, a faulty decision making pattern in excessively cohesive groups (George & Jones, 2002, p. 651), is also regarded as a potential factor.

At the individual level, uncertainty and insecurity, selective perception and retention, and habit are factors identified as responsible for resistance to change (George & Jones, 2002, p. 652). Learning new skills and the prospect of lay-offs could create powerful organizational inertia, turnover, and absenteeism that could sometimes thwart the change process. Habit, the preference for familiar actions and events, is another common factor responsible for the fight against change.

[7] Tall, centralized organizational structures are organizational structures with many management layers, in which decisions are made at the top and are sent to lower hierarchical levels for execution.

OSS implementations have to overcome these common problems of change. Aside from the common themes around the resistance-to-change phenomenon, OSS has to overcome the competitive forces of the commercial software companies, which see their business model attacked in its fundamentals. We have also included the adoption theme in our research questions about the prospects of OSS, since academic research around adoption issues can provide valuable information for the proposed institute.

3 METHODOLOGY

The first pillar of our approach is in-depth study of OSS-related literature, as well as a comprehensive review of broader IT issues. Our research included books, Internet sites, as well as scholarly articles from recognized academic journals. Our sources ranged from Canadian-based to global sources, mainly from the United States and the European Union. A second pillar included qualitative interviews in the Greater Vancouver area.

In order to achieve the goals of our research, we designed an interview guide based on open-ended questions. These questions provided us with a wealth of information by allowing the interviewees to elaborate on the topic and encouraging them to offer full-fledged analyses. We targeted senior IT professionals from a variety of companies, from proprietary software vendors to distributors of open source software, as well as companies that use IT as an enabler for their activities. We were interested in discussing with advocates as well as detractors of OSS, to obtain as complete and accurate an understanding of the phenomenon as possible.

For consistency, we used the same interview guide throughout the entire series of interviews, making only small changes and/or asking supplementary questions depending upon particular situations. From an initial list of 24 global and local companies, we were able to interview 12 people.

The main goals we sought to accomplish during the interviews were:

1. To assess the general level of awareness regarding Open Source Software within the Greater Vancouver business community;
2. To obtain expert opinion about the perceived level of support and expertise in the industry regarding Open Source Software;

3. To obtain qualitative information about the perceived risks involved in adopting Open Source Software and the perceived barriers to adoption.
4. To see how these specialists see the future of OSS;
5. To get the interviewees' opinions regarding the prospective Institute for Open Source Software: what do they think about the initiative, how do they see the relationship between this prospective unit and the local business community;
6. To obtain comparisons between open and closed source software in terms of quality of support, reliability, security, inter-operability, and features and functionality.

The interviews were conducted during the month of October 2004. The interview guide is available in the Appendix. The following table depicts the interviewees' profiles and the companies that employ them.

Table 2 – List of Interviewees

Interviewee	Profile	Company	Company Domain
1.	Senior IT Specialist, Contractor	Self-employed	Software contracts with Business Objects, IT Global Company
2.	IT Manager	Sophos	IT, Open Source, Global Company
3.	Chief Technologist, more than 12 years OSS experience, board member of Python Software Foundation, close relationships with Apache and Mozilla foundations	ActiveState	IT, Open Source, Global Company
4.	Director, IT Products	Business Objects	IT, Global Company
5.	President – North America	Sophos	IT, Open Source, Global Company

Interviewee	Profile	Company	Company Domain
6.	Senior IT Developer	Sophos	IT, Open Source, Global Company
7.	Chief Information Officer	QLT	Biotechnology, Global Company
8.	Senior IT Developer, (Perl), Release manager for Perl versions 5.005 and 5.6	Sophos	IT, Open Source, Global Company
9.	IT Manager	UBC, Faculty of Graduate Studies	Education
10.	Senior System Administrator	UBC, IT Services	Education
11.	Owner	ADA Computers	IT, small local company
12.	Operations Manager	Telus	Information and Communication Technology

The above interviewees fill senior positions in their respective companies. They are seasoned IT professionals with at least 7 years' IT experience and various degrees of expertise regarding OSS. They cover a wide spectrum, from novice users of OSS and good proprietary software expertise to expert developers of OSS applications and tools, with more than a decade of experience.

4 ANALYSIS OF INTERVIEWS

In this chapter we analyze our interviews in search for ideas as to what is needed to promote the use of Open Source Software in the local business community. In particular we synthesize our interviews from two different perspectives. First, we look into the ***Local business community's knowledge of OSS***. Next, we analyze the ***Risks and Barriers to adoption of OSS***. Finally, we look into the ***Prospects of OSS*** and the ***Recommendations*** the interviewees made for the Open Source Institute at SFU (OSI).

4.1 Local Business Community's Knowledge of OSS

We interviewed a number of IT professionals from different companies to understand how they perceive OSS. As we expected the awareness of OSS movement is high. However, the level of understanding of OSS as a technology, as a software development process, as a social movement and as a business opportunity varies significantly from one individual to another. Specifically we observed that three kinds of knowledge are required to make an educated decision regarding the use of OSS: ***Technical Knowledge***, ***Legal Knowledge***, and ***Strategic IT Knowledge***.

4.1.1 Technical Knowledge

Significant technical knowledge is required in order to effectively implement and support any software in a business environment. But implementing OSS needs deeper technical knowledge compared to proprietary software. OSS usually requires technical experts to learn to use command line utilities, the location of various configuration files (which is slightly different on every UNIX/Linux distribution), location of run time libraries, environment variables, and much more. What makes this process more challenging is that OSS documentation is usually

written by technically savvy programmers who are not trained to produce easy to read and follow documents for the less technically sophisticated audience. Furthermore, there is no single authoritative source for documentation of most OSS.

OSS tends to be less user-friendly than proprietary software. For example, to install an OSS such as Apache, system administrators have to use various command line utilities to configure the environment and the configuration files, compile the source code, install the binary, and set up the installed package for operation. Simply searching the Internet can be a very time consuming and expensive task. All of the above make the learning curve for OSS much steeper.

Installing and supporting proprietary software, on the other hand, tends to require less technical sophistication and better documentation makes the learning process easier. Proprietary software is much better written, and it has better designed user interfaces and professionally written documentation. For example, a novice system administrator can easily click through the installation process of Microsoft IIS web server to install it and then use its graphical management interface to set up a site, with almost no need to learn about the underlying operating system.

The need for a second type of technical knowledge stems from the opportunities that OSS provides for young companies. Free access to the source code of Open Source Software, some of which is distributed under relaxed licensing agreements, creates a wonderful opportunity for small start-up companies. An interviewee noted during the interview that she sees big opportunities for small software companies to benefit from the distribution and derivative work of OSS. These firms do not have the resources to develop multi-million line applications from scratch; however, if they have a sufficient understanding of OSS, they can easily create their own derivative work. Considering that under some licensing agreements derivative work can successfully be turned into some form of protected intellectual property, the potential for early

stage firms and entrepreneurs alike is significant. According to a manager from a company that has been very successful with this approach,

> "[His company], which now provides 160 –170 jobs in BC, wouldn't be here today if it wasn't for OSS. We wouldn't have built [our product] if we weren't good at consuming open source components and including them in proprietary software,"

However, in-depth understanding of programming languages, as well as open source software development practices, open source culture, and dedication to the study of a particular open source project is required before such an opportunity can be seized. Unfortunately, the above efforts create a substantial barrier to entry for the individual programmers. Therefore, the number of these highly skilled experts is very limited, compared to the potential that it creates for the local economy. A senior IT developer from a software company said during the interview that he had a hard time finding expert developers of OSS in Vancouver. He ascertained that good developers, who understand multiple OSS and also how to integrate it, are very rare. More training should definitely help increase the programmer base.

4.1.2 Legal Knowledge

Businesses do not operate in a vacuum. Any decision they make, whether regarding licensing an intellectual property or building their own intellectual property, impacts and is monitored by other players in the industry. Inevitably, executives need to be alert so that their firms do not infringe the IP rights of other firms while making sure that they capitalize on the IP created by their own company.

When it comes to Open Source Software, we discovered that the abundance of open source licensing schemes, the incomprehensibility of legal jargon to the average IT executive, and the ambiguities of owning derivative work has created substantial confusion for executives and technical experts. One interviewee, referring to the risks of running OSS and the viral infection issue – see chapter 2.2, said:

> "I see two risks from a business perspective. One is that some OSS has a certain type of license [...] that if you put inside your software you can lose the IP ownership of your software, so you have to understand your license model. Second, [...] if your business model depends on your software being proprietary and close, then you have to be careful how you use OSS. [...] How you integrate the pieces of proprietary and open software to build a business model around it [and at the same time] use it [the software] under the term of the license is a challenge."

Another CIO from a biotechnology company went even further:

> "In general, OSS has to pass the test of intellectual property and patent infringing lawsuits. [...] Companies (public or private) do not want to be held accountable if a programmer inserts some intellectual property code of another company into an OS software package they are using. Instead, companies want to be able to point the finger at another legal entity that supplied them the software to prevent possible legal repercussions. A primary example of this is the patent infringement lawsuits made by SCO against IBM, Red Hat and Novell, and the outbreak of OSS users being sued. Until OSS sees more software companies coding under the OSS label, which one would hope has a proper software development life cycle, then there will be a perceived risk of using OSS over proprietary software."

In other words, firms look for two things. First, that they do not infringe IP rights of others by consuming OSS or by creating derivative work from existing OSS. Second, they fear that, due to the intricacies of open source licensing schemes, the interdependence of some open source and closed software (most notably, Linux and commercial UNIX) or because of a viral license, they may lose the ownership of the intellectual property that they create.

4.1.3 Strategic IT Knowledge

Tim O'Reilly explains the open source movement as a "Paradigm Shift", one that creates new opportunities and poses new challenges for all business (O'Reilly, 2004). For some organizations to survive and excel in the new paradigm it is crucial that their executives understand how OSS can be leveraged to give them a competitive advantage. Specifically, Open Source Software can create three advantages. First, by implementing various OSS based on open standards, firms can avoid vendor lock-in. Second, as mentioned in section 2.4.2, cost cutting is one of the most important reasons for using OSS – indeed, it is the top reason in Canada and the

second top reason according to the European FLOSS report. Therefore, cutting the cost of IT infrastructure can create a cost advantage for the companies. Third and perhaps most important as demonstrated in section 2.2, given that there are a number of well-established OSS with substantial market share, technology companies can quickly build proprietary solutions either based on or around OSS for an already developed market. Oracle9i Real Application Cluster (RAC) for Linux clusters is an example of porting commercial software to a platform with significant market share in the server market. Another example is ActiveState/Sophos's PureMessage, which uses SpamAssassin, an OSS SPAM filter, to enable SPAM filtering on Sendmail Mail Transfer Agents.

After interviewing a number of individuals from the local business community, we feel that most senior IT personnel and executives do not fully appreciate the opportunities and the threats of this new paradigm. This is consistent with the findings of the federally funded "Open Source Software in Canada" study which concluded that senior management need to be educated to appreciate that open source is strategic not just for Information and Communication Technology (ICT) industry but also for others (e-Cology, 2003, p. 65).

4.2 Risks and Barriers to Adoption of OSS

Since we believe that the primary mission of the prospective SFU Open Source Institute is to promote the use of Open Source Software in the local business community, our primary goal in the interviews was to find out what is stopping businesses from using OSS. Although many reasons were given as the barriers to the adoption of OSS, we can split these reasons into three broad categories, ***Technical Reasons***, ***Nature of Open Source Software***, and ***Business Reasons***.

4.2.1 Technical Reasons

As explained in section 4.1.1, implementing and supporting open source solutions requires a high level of technical competence. A novice system administrator who may not be

comfortable using the typical text-based download-configure-build-install process of installing or upgrading an open source application such as Apache web server can easily step through a Graphical User Interface (GUI) and install or upgrade Microsoft IIS. Therefore, we cannot expect more system administrators to promote open source in their organizations unless they are reasonably comfortable with the installation, configuration, and maintenance process of open source software. As we will see in the next few pages, beside plain technical skills, there are a number of other technical barriers to wide scale adoption of OSS.

4.2.1.1 Lack of Social Skills in Dealing with Online Communities

Programmers and system administrators need a broader skill set when working with OSS. For programmers, contributing to an open source project involves more than producing well-written code. Since the software development process is highly dynamic and transparent in the open source community, developers need to have a broad set of social skills as well as technical skills. Referring to this, an experienced Chief Technologist said:

> "Some developers find [OSS communities] a very threatening world. In the open source world nobody knows how senior you are supposed to be: [...] you are looking to your fellows and you are looking to your contributions and your abilities to communicate to be a social animal, as opposed to being just a developer. So open source is as much about socio-dynamics as it is about code. If you can't convince people that you have the right approach, your code will not be inserted. It is a very different way of interacting with peers than having a senior developer or a manager to tell you what to do."

For system administrators, maintaining an open source application is not limited to making sure that it runs smoothly on a machine. Superior social and decision-making skills, such as effectively monitoring the status of open source projects, knowing which online resources to use to stay updated, knowing how to network with other open source users and what can reasonably be expected from the community, and knowing how and what to contribute to the community, deciding when to upgrade to a new version, when to apply a bug fix, and how to proactively maintain the installed application to make sure it performs at its best, are required to

effectively and efficiently maintain open source solutions. The lack of technical and social skills makes in-house support of OSS more challenging.

4.2.1.2 Inadequate Support

Inadequate support was perhaps the most cited barrier to adoption of open source software in our interviews. Analysis of our interviews reveals three reasons for this perceived lack of support. First, as explained earlier, there seems to be a shortage of people with the right combination of technical and social skills to provide in-house maintenance for open source products. Second, once set-up, mature OSS run smoothly and require minimal effort for daily operations – therefore, most firms choose to reduce their IT personnel costs by having fewer system administrators on payroll. Because they have fewer people onboard, occasionally they need to hire external consultants and they prefer to have a bigger pool of consultants when negotiating deals. Third, and perhaps most importantly, firms who are not in the business of supporting a particular application prefer to pay another company that is highly specialized in it instead of supporting it in-house. For example, a firm such as Unilever, which is not in the business of developing, distributing, or maintaining Linux, would pay a company such as IBM for high-end support of their Linux systems (Weiss, 2003).

Historically, proprietary technology companies have provided a multi-tier support channel including:

1. Limited free support for basic installation and maintenance
2. Support through Value Added Resellers (VAR) for more complex installations
3. Around-the-clock support contracts with the technology company for mission-critical systems.

Depending on their needs, user companies have used these free or paid services to maintain their systems. So even though most IT managers have no doubts regarding the maturity of much open source software, and they are well aware of free support by the open source

community, they still need a multi-tier support channel to meet their requirements and budget before they can implement OSS with peace of mind. For example, their need is more than Red Hat Linux support through Red Hat: they need well-respected local companies which act as "VARs" and "Solution Providers" to fill in the gap between free community support and Red Hat's high-end support. One interviewee commented:

> "As soon as you go away from these mainstream things [mainstream OSS applications such as Red Hat Linux or Apache], [...], even if you do have access to the source and in theory you can get one of your staff to start working around source code, I think that it is a huge waste of time for any IT organization to start doing it; you could pull out a few exceptions such as Google that has the customisation, but in your average IT organization you don't want people to really spend time trying to fix issues [...] and it takes a lot of effort to learn what is going on inside those packages..."

4.2.1.3 Previous Technology Drag

Another barrier to the adoption of OSS is the existence of legacy applications built on proprietary platforms. This is one of the factors that influence the diffusion of innovation according to Fichman & Kemerer (see section 2.6.5). The presence of proprietary systems slows down or even stops altogether the adoption of open source, in four ways.

First, many firms have invested in in-house applications built on proprietary platforms, or written in proprietary languages. Rewriting these applications from scratch for the sake of using open source platforms makes little economic sense. Second, most proprietary software is bundled so that it paves the way for more proprietary software from the same vendor. For example, if an organization running Sendmail as its Mail Transfer Agent (MTA) needs a calendaring and collaboration tool, it may adopt Microsoft Exchange. By adopting Microsoft Exchange, which is also an MTA, a new question is raised: shall we continue to run Sendmail as our MTA? Furthermore, the best client to access E-mail, Calendar, and Public Folders on an Exchange server is MS Outlook. Interestingly enough, MS Outlook itself is bundled with the MS Office productivity suite, which runs best on an MS Windows operating system, and so on... It is easy to

see how adopting one proprietary solution leads to a single vendor's domination of servers and desktops. Third, unlike open source software, which is by and large based on open standards, proprietary systems usually either use proprietary protocols or modified open standards. Given the omnipresence of certain proprietary systems, this lack of adherence to open standards makes it difficult to integrate open source systems into existing proprietary systems. A fourth and perhaps related barrier to the adoption of OSS is the network effect of existing proprietary software. For example, Microsoft Word and Adobe Acrobat formats are the most widely used formats for exchanging text documents across the Internet. To be able to effectively communicate with the rest of the world, one needs to use software that can read documents in these formats and produce documents that can be opened by this proprietary software. In many cases, it follows that nothing works better than software produced by these vendors, hence they become increasingly more popular.

4.2.1.4 Lack of Viable Open Source Applications for Certain Domains

Businesses use software to automate tasks, solve complex problems, increase productivity, cut costs, and facilitate communication. To achieve these goals, they use a wide variety of applications. Unless most of these applications are available either in the form of an open source product or at least can run on an open source platform, OSS will not be widely adopted.

Our interviews confirmed the availability of OSS for IT infrastructure as explained in section 2.2. But, with the exception of some Mozilla-based web browsers and of the OpenOffice productivity suite, there is a limited supply of viable open source desktop applications. This shortage is even greater in highly specialized areas other than IT, such as business applications (ERP, CRM[8], etc.), financial modelling, accounting, and CAD/CAM[9]. Some such applications

[8] See Glossary.
[9] See Glossary.

can run on open source platforms such as Linux or interact with other open source software such as MySQL, Apache, etc., but their number is very limited. An experienced Chief Technologist commented about the lack of applications in very specialized domains:

> "The more non-technical [IT related] or more vertical a domain, the harder it will be to come up with viable OSS applications. Something that is vertical but very technical is e-commerce. Lots and lots of people do on-line sales. There are viable open source e-commerce systems. [...] As soon as you get into legal domains, [...] the risks involving picking up an open source accounting package is pretty high. You might save some money, but it could not be fiscally prudent to take that risk. [...] It depends on who else in the world needs to do that sort of task, how much of a community they are [...], how technical they are."

4.2.2 Nature of Open Source Software

Open source software is different from proprietary software in one fundamental way: there is usually no legal entity that owns, manages, and takes responsibility for them. Most OSS is developed by a group of geographically dispersed programmers without the financial and legal support of any legal entity. The lack of ownership by a legal entity creates a few barriers for the adoption of OSS.

4.2.2.1 Lack of Accountability

Where would you go if a particular application failed? Or whom would you hold liable if a malfunction occurred? The answers to these questions are straightforward if you are dealing with proprietary software. The legal ownership of the software by a firm provides a clear and centralized accountability for the firm behind the software. In theory, you can expect the software vendor to rectify the problem in a reasonable time at no or little cost. Should the software vendor fail to make the best effort in a reasonable time frame to support the software, you have the option to take legal action against the vendor. Referring to accountability, one interviewee has pointed out:

> "Take into consideration our Spam prevention. We'd checked out some Open Source solutions but elected to go with a commercial vendor. This isn't necessarily because the other solutions, including home-grown and Open Source

> didn't suit our needs. The constant pressure is to move in the direction of commercial software - apparently under the guise that if they don't deliver we can hold them accountable/responsible - but the latter part has yet to occur that I'm familiar with."

When you are dealing with OSS, the answers to the above questions are a lot more ambiguous. The absence of legal ownership of OSS leads to a lack of accountability, or at best to a decentralized accountability without clear distribution. No single entity is completely responsible for the software: some of the responsibility is shifted to the system administrators and IT managers and the rest lies with the open source community, which cannot be held liable, coerced, or even expected to support the software. They will only do so, if they perceive it as important. If the issue is seen as important and if there are enough people affected by the problem, the motivation to fix the problem is higher, but there is no intrinsic guarantee.

Consider for example the following scenario: a security bug is found in a particular proprietary software and a similar bug is found in a competing Open Source Software, for example a web server. The administrator of the proprietary software is not expected to do much until the software vendor officially releases a patch. Once the patch is released, all that needs to be done is to use GUI to go through the installation step by step. Whether the patch is quality assured is up to the software vendor. Furthermore, in case the installation fails there is one authoritative source to go to or to blame it on. However, the administrator of the OSS faces far more responsibility. First, given that (s)he has access to the source code, (s)he might be expected by colleagues or management to fix the bug, a capability that the average system administrator lacks. Second, it is likely that some independent open source programmers release a bug fix before the "Core Project Developers" formally approve one of them or release their own bug fix. The decision to apply an unapproved patch or to wait for an official patch lies on the shoulders of the technical expert. Third, regardless of which one he chooses, no outside legal entity can be held liable, either for the bug or for a faulty patch. Fourth, if the installation, which usually requires sufficient technical competence, fails, the system administrator is at the mercy of the

open source community, however helpful. Referring to the cooperativeness of the OSS communities, one of our interviewees said:

> "A good example is SNORT: it is a open source product that's been out there [for which] there is a lot of support [...] I used a source called Internet Security Guru [...] I have some issues, I email the group and, generally, in 4-6 hours I have an email back."

4.2.2.2 Inability to Influence Direction of Projects

The lack of ownership gives rise to another problem. Governments and large corporations often need specific features and functionality. When dealing with commercial enterprises, these large institutions have considerable financial and contractual leverage that they can use to shape the future direction of the software they choose to implement. But, since open source software is usually developed and maintained by a community of volunteers who usually have motives other than short-term monetary gains, it is hard for large institutions to influence the direction of the projects. These organizations see this as a downside of open source software and stay away from OSS. One interviewee commented:

> "They perceive they have less influence on open source project as users than they might have as customers of a company because they don't have financial leverage on the project."

4.2.2.3 Risk of Abandoned Projects

The lack of legal ownership by a single entity makes adopting OSS more risky than proprietary software. Some of the interviewees mentioned that corporations fear that an open source project, regardless of how promising it is, may be abandoned by its leader or core group. One interviewee explained:

> "There is an inherent risk of OSS projects regarding their stability; the risk is bigger for small OSS projects which can disappear while companies are left with unsupported software."

Another interviewee confirmed the above concern, but in his opinion the risk of project abandonment is affected by the size of the community and the length of the history of the project:

> "As a rule, it [abandonment] doesn't happen for high-profile projects. It only happens on projects that got started by one person or a small group of people that then lose interest and go on to do other things.... If the software has been around long enough for people to have come to depend on it, there will always be someone to keep it alive, perhaps not in active development, but at least just in maintenance mode.

4.2.3 Business Reasons

More often than not, businesses choose a particular technology for reasons other than its technical merits. As we will see in subsequent sections, the business model, the key people in the organization, the partnerships with other vendors, and many other non-technical concerns drive the corporate decision making around information technology.

4.2.3.1 Legal Concerns

As explained in section 4.1.2, most IT executives are seemingly perplexed by the legalities of open source software. This lack of understanding leads to a fear of the unknown. And this fear is heightened by some of the existing legal battles such as the infamous "SCO vs. IBM" lawsuit, which was cited by three of the interviewees.

Another legal concern arises from the business model and the strategic partnerships that some corporations have with other software vendors. A business that builds software in collaboration with another proprietary software vendor is less likely to use open source software or to contribute to the open source community for two reasons. First, using open source software, which competes with the other vendor's software, may jeopardize the relationship with the other vendor, because it is as if the firm is giving business to or trusting a competitor. Second, our interviewees mentioned that some firms fear that they may be sued by the other vendors in case any of the other proprietary software vendor's intellectual property is revealed to the open source community. However, as one of the interviewee said,

> "As more and more large companies use it [OSS], I think that people will be more comfortable to use it because they perceive less legal risks."

4.2.3.2 Human Resource Challenges

Most of the barriers to the adoption of OSS or of any other technology can be explained in terms of human resources challenges. As we saw in section 2.6.6, resistance to change is manifest at three levels: organizational level, group level, and individual level.

During the course of our interviews, we realized that young, flat, and entrepreneurial companies are more likely to adopt open source software than old and established companies. Small and privately held companies are more concerned with efficiency and innovation. These small companies tend to empower employees to choose the technology they use. Large and publicly traded companies, however, tend to have stronger inertia to any kind of change including adopting new technology. This observation was confirmed by one of our interviewees, a Director of IT Products in a global software company:

> "When I think of open source I think of a more cutting edge and innovative company, whereas the larger the organization, to be honest, the less innovative you are [...] You are less flexible, you have these processes, you are bureaucratic..."

Additionally, replacing proprietary software with OSS is likely to shift the power in the organization from people who have vested interests in using proprietary software to those who will benefit from OSS implementation, including potential new comers. We also need to recognize that businesses face numerous challenges. As explained in section 4.1.3, except for IT-intensive companies and technology companies who wish to commercialize OSS, most other organizations have higher priorities than deciding what type of software to use. For example, for most organizations the highest priority is financial sustainability, and other issues such as choosing technology have far less priority. Furthermore, even when choosing technology is a priority, the department of "business development" in most cases prefers a partnership with a software giant and oppose "IT or Engineering" department's proposal to choose open source software based on its technical merits. One of the interviewees has mentioned that their internal

policy has always been to make partnerships with large software companies, even if some other solutions could be better technically. Regardless of which is a better solution, this difference in the orientation of the departments is a source of friction and resistance to change.

At the group level, when the group is homogenously composed of people with insufficient understanding of OSS, groupthink and other faulty decision-making patterns may prevent the group from evaluating all options in search of the best solution. Also, implementing OSS may require the changing of the team or the hiring of new members. This poses a challenge in highly cohesive teams whose team members resist change. While this problem may be seen as a technology drag, it is essentially a "Human Resource" issue.

At the individual level, OSS can be very threatening to some people. It may require learning new skills, restructuring, lay offs, etc., which causes most people to feel insecure and anxious about their future. This is particularly challenging for individuals who have made their careers in proprietary software. As explained in section 4.2.1.1, competence in OSS requires a broader skill set. Programmers in proprietary software companies are not used to having their code and software architecture audited and scrutinized by numerous and sometimes anonymous individuals. They may also lack some social skills such as persuasion and negotiation skills, which are needed to effectively work in flat non-hierarchical environments such as the open source community, as explained in section 4.2.1.1 and confirmed by a senior OSS developer in the same section.

Furthermore, individuals tend to selectively filter information and "see what they want to see", which makes them less capable of expanding their horizons and embracing new ideas. One interviewee referred to this tendency he observed at many people:

> "You will find certain individuals who are more biased than others. If there is a bias, it tends to be very strong. [...] At either level. People who do have a bias, I found that to be a very strong bias, one way or the other. Everything must be Microsoft or everything must be Open Source."

At last, there is plain preference to continue doing what is familiar and resist new ways of doing things.

4.2.3.3 Corporate Policy

Some companies have corporate policies against using OSS, whether explicit or implied. From our observations, there are three main reasons for these policies: partnerships with other software vendors, regulatory requirements, and conflict of interest.

First, a company which works closely with other proprietary software vendors may adopt a policy against using OSS to minimize the risk of future litigations and avoid jeopardizing its partnerships. One of the interviewees acknowledged:

> "[Our company] is a loyal partner of Microsoft and other proprietary software vendors; thus, the internal policy does not favour OSS products or products from small vendors, even if they could be better technically".

Second, there are cases where companies are not free to choose technologies based solely on technical merits or financial benefits. Companies in certain industries such as biotechnology and the pharmaceutical industry need to meet strict regulatory mandates when choosing technologies. Therefore they may not implement OSS, simply because it is not approved by regulatory bodies. One CIO from a biotechnology company said:

> "We operate in a highly regulated environment (FDA, Health Canada, etc.) The limit we face is that OSS does not yet have a high level of acceptability with the regulators. It is often more work to validate an open source piece of software than it is to validate a comparable traditional closed source software package. This validation cost can easily eat up the savings of using OSS."

Third, for some companies using OSS would create a clear conflict of interest. These companies will ban OSS for any number of legal, technical, marketing, or public relations reasons.

4.2.3.4 Financial Justifications

It is often argued that companies can save money by adopting open source. In our findings, some companies have access to cheap proprietary software, and therefore cost savings is not a strong proposition. Large corporations and organizations receive large discounts because of their purchasing power. In other cases, some technology companies are able to access proprietary software for small or no fees through cross-licensing: they offer licences of their software in exchange for licenses of software from another vendor. And finally, some companies use pirated software. According to World Bank Data on software piracy in 2001, 38 per cent of the 14 million PCs in Canada and 25 per cent of the 178 million PCs in the U.S. used pirated copies of Windows and Office XP (Ghosh, 2003). Whichever method companies employ to acquire software, proprietary software costs less than its retail price.

Another financial barrier to the adoption of OSS is the cost of user training. Corporations have already incurred significant costs to train users to use ubiquitous proprietary software. This is a sunk cost, and a substantial portion of it cannot be recouped if they switch to open source software. Hence, they will tend to resist incurring yet another sunk cost in training employees to use the new software.

4.3 The Prospects of OSS

We have asked the interviewees about the prospects of the OSS, from the point of view of both server-side and desktop-side applications. Server-side applications (operating systems, databases, infrastructure applications) will grow in the future – this was the opinion of the majority of the interviewees. This opinion is supported by the government-sponsored study canFLOSS, where 98 per cent of the respondents believed that OSS would be used more in the future, especially for backend applications, networking, and in embedded systems (e-Cology, 2003, p. 23). While one senior IT developer thought that

> "In the next five years [...] it will continue pretty much as it has so far,"

another IT manager acknowledged that:

> "OSS is at a point where it can no longer be ignored. Its popularity will increase to the point of competing with current vendors."

Another IT manager said that he sees a promising future in the data centre area, infrastructure servers such as DNS, firewalls, DCHP servers, web servers, and even database servers, but he has doubts regarding major developments in the desktop applications area. He predicts that technical users will continue to embrace open source desktop applications, and call centres, which only require dumb terminals, may also start using OSS. Open source desktop applications will have difficulties gaining widespread adoption because of the incumbents' significant advantages. In the area of customized and particularized application, OSS will probably never take off because of the inherent difficulties that come from the OSS development model: it needs a large community of interested developers to come together around a common project, and to attain critical mass for a project to become important and viable, which is not the case for very particularized applications (such as power plant software, financial modelling software, etc.).

4.4 Interviewees' Opinions Regarding the Open Source Institute

The central subject of the interview process was the interviewees' perspective about the Open Source Institute initiative at SFU. The main ideas that came across were:

4.4.1 Increase Awareness Around OSS

The institute should increase awareness around OSS projects and products, by organizing conferences on OSS-related topics and acting like an incubator for promising projects, or by making governments more aware of the potential of OSS. One senior IT developer thought that this institute:

> "... should make the Government more aware about OSS. I am sure that there are pockets in Government where people are aware of the technology [OSS]. There is no policy in the Government around open source. For instance, the Government could create policies regarding the acquisition of software for government projects."

4.4.2 Training

Training was another issue identified by many interviewees. The need for multi-tier training was widely expressed and many of the interviewees have mentioned that the institute should have a role in assuring it. First level of the multi-tier training is represented by technical skills for students, programmers, and IT managers and administrators. One of the interviewees commented:

> " I think there is little momentum inside universities about OSS; training is lacking, people have to learn OSS by themselves, they can't go and take training, [...] they have to be very self-driven. [...] Such an institute could train people who know how to leverage OSS."

The second level of training mentioned by interviewees is represented by social and management skills for managers, focused especially on dealing with decentralized, widespread networks of specialists, informally grouped around certain OSS projects. The third level of training addresses strategic issues: how to leverage OSS for business benefits (for executives), how to build new products and technologies using OSS and proprietary software, how to mitigate and understand the legal issues around OSS licensing models. One interviewee acknowledged the need

> "to train the legal and business people to understand how they could build [...] software based on open source technology. I think we are experts at that. [*He is referring to his company, which succeeded in building proprietary software around OSS*] The legal aspects are interesting because it could confer a competitive advantage [in building software]."

4.4.3 Brokering between Businesses and OSS Communities

The institute could act as an intermediary between the diverse OSS communities and the small to medium-sized businesses, by recommending good technical solutions, helping in stabilizing the diverse distributions, and creating guidance for the average IT administrator to

help them in dealing with change. Referring to this intermediary role, one of the interviewees has commented:

> "[The Institute] could work with companies [involved in OSS] to establish proxy support organizations for OSS. [...] Small companies don't have time and resources to track the fast world of OSS products. [Our company] was to be this proxy between small companies and fast moving open source world."

At the same time, some end-users requirements could be centralized and the most frequent of them sent to the project communities for continuing development.

4.4.4 Research

An interesting idea came from a senior OSS developer, who has more than ten years of experience in developing and actively participating in many OSS projects. He explained that, even if the source code is free and, at least theoretically, everybody can learn from the way the issues were solved, there are numerous issues that will never be documented for the sake of increasing the knowledge base. Thus, the developer thinks the institute could play a role in documenting some sound solutions embedded in the written code, estimating that there is a "potential for research worth a number of PhD theses."

5 THE DIFUSSION OF OSS AND THE ROLE OF GOVERNMENTS

To replace proprietary software with open source is a huge undertaking for most companies. It is simply not possible to fully understand the technical, human resources, strategic, legal, and financial impacts of migrating from proprietary systems to open source software. Only organizations with substantial resources can afford to perform these studies and even then the results will be controversial. In October 2004, AT&T unveiled its plans to compare a few operating systems including Linux as future choices for its tens of thousands of desktop computers. This study will cover security, reliability, and total cost of ownership and is expected to be finished by late 2005 or early 2006 (La Monica, 2004). Furthermore, frameworks and studies used by one company may not work for another. For example, a particular security solution, which may be adequate for a company, may not be acceptable by a financial institution because of what is at stake.

The actual migration from proprietary systems to OSS is orders of magnitude more challenging than feasibility studies. For many years companies have invested in proprietary software. They have incurred all kinds of costs, have developed organizational norms and cultures, and have formed businesses around proprietary solutions, none of which can be easily changed. Therefore we have to recognize that open source software will never be adopted by some businesses.

5.1 The Stance of OSS from the Diffusion of Innovation Standpoint

In section 2.6.5 we presented some of the factors that influence the adoption speed of a new technology. Based on the results of the interviews and research we conducted, we can

comment about the potential of OSS in the future. The following sections show our current understanding.

5.1.1 Relative advantage

Most mainstream OSS applications are built around open standards and have a modularised architecture. This confers a relative advantage compared to proprietary software, which is less modularised and built around proprietary standards. The majority of interviewees as well as government-sponsored studies have asserted that OSS has fewer bugs and security issues. Additionally, the same government-funded studies confirm that OSS has lower acquisition costs compared to proprietary software. On the negative side, OSS is generally less user-friendly, and, except for technically savvy users, others find most OSS hard to use. Overall, the perceived relative advantage depends on the application, and for the average user OSS offers no substantial advantage. Many IT specialists are waiting to see what the future will bring.

5.1.2 Compatibility

IETF, W3C, ANSI, IEEE, and similar standardization organizations aim to increase compatibility and interoperability of systems. It is through the efforts of these organizations that standards such as TCP/IP protocol, Electronic Data Interchange (EDI), HTML, XML, and POSIX have emerged and increased system compatibility for OSS and closed-source software alike. With respect to adherence to open standards, the OSS movement favours and strives to implement systems based on well-established open standards. Therefore, OSS is usually compatible and interoperable with other systems that implement open standards. The problem of incompatibility arises when OSS or its user has to deal with another system that uses a proprietary standard. In other words, the extent to which closed-source software uses proprietary protocols and the size of the installed base of closed-source software influences the advantage that OSS offers in terms of compatibility. In backend systems, applications are relatively small in number (compared to

desktop applications), systems are modularised, and there is a substantial presence of OSS, therefore closed-source software vendors are forced to use open standards. Although sometimes they modify these standards slightly, they have to make sure that their systems interoperate with the critical mass of open source systems. On the desktop side, however, applications are numerous and dominated by technology compatible with Microsoft and Apple Macintosh. The applications tend to be less modularised and deeply integrated with the underlying operating system; therefore, it is harder for open source software to achieve a high level of compatibility with existing systems. Although standards such as XML and HTML have certainly increased the compatibility for desktop applications, they are still far away compared to backend systems.

5.1.3 Complexity

From the users' point of view, many OSS applications have far more difficult GUIs (if any) and users perceive them as more complex that current proprietary software. A senior OSS developer acknowledged: "OSS is not so well polished, [...] it still lacks commercial software interface." Furthermore, OSS documentation, help and user manuals are written by programmers instead of technical writers, which makes them harder to understand and learn.

5.1.4 Trialability

OSS is easy to try and freely available, though technical knowledge is usually required to make it work. Most applications run on proprietary as well as open source platforms. But some OSS, such as operating systems, either has to run on separate machines or more technical knowledge is required to make them co-exist on the same hardware as another operating system, which makes them less trialable.

5.1.5 Observability

IT professionals have long been aware of OSS, and awareness continues to grow for backend systems. However, for the large segment of population that uses desktop applications, observability is low, as many interviewees have asserted.

5.1.6 Prior technology drag

The installed base of proprietary software is a big challenge for OSS applications. In desktop computing, about 90 per cent of the world's computers run on Microsoft OS. This is a huge factor that will negatively influence the speed of adoption. On the server side, OSS becomes increasingly more popular because prior installed base is more expensive and OSS has reached maturity. The savings of running OSS are larger on the server side.

5.1.7 Irreversibility of investments

The risk of investing in OSS is lower than that of investing in proprietary software, for two reasons. First, investing in OSS is usually less expensive than investing in proprietary software (see section 2.4.2). Second, in the case of open source software, even if the investment is stopped, the source code (which is readily available) can often be reused in other applications. This versatility is a plus of OSS for technically savvy companies, but not for the average user.

5.1.8 Sponsorship

In the last years, a number of large IT companies have massively invested in the development of OSS (IBM, Sun, HP, etc.) This commitment to the development and promotion of OSS creates trust and helps the adoption.

5.1.9 Expectations

There are relatively high expectations and optimism regarding the future of OSS, especially for the server-side applications and operating systems. However, pending patent infringement lawsuits cause some specialists to be cautious.

5.1.10 Summary

The above factors show mixed signals from the diffusion of OSS perspective. On the server side the prospects are better, a fact expressed by many interviewees. There are many mature products that are leader in the market and /or have real potential. Factors such as expectations, sponsorship, trialability, compatibility, and relative advantage (including price) favour server-side OSS applications while others, such as complexity, are mitigated by the fact that the users of server side applications are usually IT specialists. For the desktop side applications the future is unclear. Prior technological drag is huge, there is a big installed base and users perceive OSS applications as more complex. We conclude that there is no apparent compelling advantage at this time, and no possibility to predict what will happen in the future.

5.2 The Role of Governments regarding OSS

As we acknowledged in section 2.6.1, OSS satisfies the definition of a public good. It is a non-rival and non-excludable good. The economic literature has treated the aspects of public goods provisions in sufficient depth, so we can extrapolate the theory to the case of OSS.

As discussed in section 2.6.1, the public goods are undersupplied because too few want to produce them. In the case of OSS, widespread groups of programmers develop software, which is available to everybody. The free riding problem is manifested in the following way: programmers have their motivations in participating in the OSS projects, but they choose what components to write and the projects develop somehow organically. There is little incentive to write helps, GUIs, or documentation for OSS, as well as to write OSS applications for very narrow, specialized

domains (Johnson, 2002, p. 639). Thus, OSS as a public good is undersupplied: it has less documentation, it is less user friendly, and it lacks some of the functionalities proprietary software has, such as the "accessibility options" for impaired users, as one senior IT developer mentioned in the interview. Therefore, there may be an opportunity for government intervention, as an institution concerned with social welfare, to support the provision of OSS as a public good. They should support those OSS applications and components that are not efficiently supplied by the market. The problem with governments' intervention is that they cannot substitute the market – as people usually misrepresent the quantities of public goods they need (believing that the personal cost of public goods is zero), governments tend in many cases to overestimate the demand for public goods and therefore oversupply them (Baye, 2002, p. 516). Governments have difficulties finding the efficient level of supply that makes society better. The aforementioned senior IT developer sees a role for the Open Source Institute in making governments more aware of OSS and its potential:

> "OSS is a reusable public asset that could be leveraged using taxpayers' money to increase public welfare."

By encouraging OSS and open standards, governments encourage competition and innovation in the software industry, with benefits for the entire society (the same senior IT developer said).

The Government of Canada, the largest user of IT in the country, has an ambitious program to increase the efficient use of IT solutions in all Government agencies. The program is called "Federated Architecture Program" and is a "government-wide approach to planning, designing and implementing the Government's strategic IM/IT (Information management / Information technology) infrastructure" (Treasury Board of Canada Secretariat, 2004). The program is being phased in through a number of "iterations." OSS is specifically part of this strategy, a number of strategic principles from iteration one being directly applicable to OSS:

1. Reduced integration complexity;

2. Security, confidentiality, privacy and protection of information;
3. Proven Standards and technologies;
4. Total Cost of Ownership (Treasury Board of Canada Secretariat, 2004).

This is good evidence that the Government of Canada considers OSS technologies an alternative worth considering. However, the Government of Canada neither prevents nor encourages the use of OSS (e-Cology, 2003, p. 5). The Open Source Institute should have an important role in promoting OSS at the government level, both provincially and federally.

6 FINDINGS AND RECOMMENDATIONS

This chapter reports the findings of our research and based on the findings we make recommendations for the SFU Open Source Institute (SFU-OSI).

6.1 Findings

In this section we synthesize what we found to be IT industry's needs, gaps between what is currently available on the market and what specialists perceive is needed, as well as issues that came across the research we conducted. Our recommendations that will come in section 6.2 are based on the following findings.

6.1.1 Insufficient Technical Support for OSS

Insufficient technical support for OSS was a leitmotiv during our research. Almost all interviewees mentioned that inadequate support is an issue, a risk, or a barrier to adoption of OSS. This is because of many issues such as the fact that OSS is less user-friendly, so the need for support is higher, the community of developers tends not to develop features they are not interested in (such as help menus, documentation, etc.), there is confusion among IT managers regarding various distributions of the same product (which one best fits their needs and has the required level of support?). Other studies sponsored by Governments (canFLOSS in Canada and FLOSS in European Union) have confirmed our finding.

6.1.2 Insufficient Awareness

Many of the interviewees said that the level of awareness within the IT community is good. Even if people hadn't actually used OSS applications, they were aware of OSS and the controversy surrounding it. However, even if they have heard about OSS, a good part of the IT

community does not see the benefits of running OSS applications (see section 2.4). This is not to say that they should abandon proprietary software they are currently using, but a balanced, hybrid approach could improve their performance. There are software companies that have successfully built hybrid business models. For the non-specialist user, who is generally using desktop applications, things are different. The awareness is very low, because they are currently using mature and market dominant proprietary software. Most interviewees did not see significant growth in desktop-side OSS applications in the near future.

6.1.3 Chasm between Proponents of OSS and Closed Source Software

During the research process we found that there is a chasm between proponents of OSS and those who prefer closed source software. This is not something we had not anticipated. We interviewed a group of IT specialists from the Greater Vancouver area, about half of them being proponents of OSS, while the other half non-users or detractors. What surprised us were the intransigent partisan positions, either for or against OSS that these interviewees held or had encountered in their organizations. These attitudes could be explained by political, inertial, business factors.

6.1.4 Skills Gap

Many of the interviewees perceived a need for supplementary training in dealing with OSS. In the last years, a number of large companies started to provide training for mainstream OSS applications. However, this is not sufficient, for two reasons: first, because the developing communities of most viable OSS applications do not provide training, and second, because there are still very few third-party providers of OSS training. In this respect, OSS differs greatly from proprietary software, where third parties provide much of the training. As a result, more academic technical training in this area is necessary, as some specialists pointed out.

Another area of training identified as necessary for creating good developers is social skills: how to behave in widespread communities of peer developers where there is no manager to tell programmers what to do and every solution has to be accepted and recognized by peer developers; how to observe these OSS communities and interact with them to obtain information and support for one's own OSS applications. Another skills gap is around business models. We presented in section 2.6.3 the current competitive strategies in the industry. Many interviewees showed that they need more knowledge on how to build products around OSS, how to commercialise them, and how to deal with competition and legal issues. There is uncertainty and many people are afraid of legal consequences (such as patent infringements, lawsuits, etc.).

6.1.5 Little Government Involvement

In our research we found that the Government of Canada does not have a clear policy around OSS. However, a strategic program that started in 2000 and is currently underway, the "Federated Architecture Program," tries to develop a balanced policy regarding acquisition of software, both proprietary and OSS. The program is still in the initial stages and promises to create a necessary policy regarding the use of OSS for public institutions in a period of 3-5 years.

Some IT specialists complained about this lack of policy and asserted that federal involvement in OSS and more generally, in open standards, could increase competition in the IT industry with benefits to consumers and social welfare in general. Governments, as major consumers of IT products, are better off if they reduce their dependence on certain expensive vendors, while increasing the standardisation of their infrastructure.

6.1.6 OSS Is Perceived as Not User Friendly

During our research we confirmed what other studies had ascertained about the perceived lower user friendliness of OSS. This is a major barrier to adoption. We would like to emphasize

this finding because more involvement should exist in creating user-friendlier features, and the proposed institute could definitely play a role in this area.

6.1.7 Undocumented Innovation

Even if by freely distributing open source code, at least theoretically, a certain level of transparency is assured, there are many innovations embedded in mature OSS that are not documented and others cannot benefit from them. Students and young programmers could benefit from the exposure to these technical solutions used to solve certain problems during years of development of OSS. Researching those somewhat hidden innovations and documenting, creating a repository of them, would definitely help teaching technical skills.

6.2 Recommendations

6.2.1 Brokering Support for OSS

As explained in section 4.2.1.2, lack of multi-tier support is one of the major barriers to the adoption of OSS. There are a number of initiatives that the SFU-OSI can undertake to help create multi-tier support for OSS. For example, small companies often do not have the time, money, and expertise to perform market research to find companies that can provide support for their IT needs. Similarly, small OSS support companies have difficulties in generating leads and closing support contracts with user companies. To assist both groups, the SFU-OSI can create a portal-like system to match the need for OSS support with the best local providers. This would help small OSS support companies and will make OSS a more viable option for more companies.

As mentioned in section 4.2.2.2, some companies do not adopt OSS because they cannot influence the direction of the projects. This inability stems from the fact that they don't have the financial and contractual means that they usually use to influence companies. There is nothing that the SFU-OSI can do to create these financial and contractual means. The nature of open source software, as explained in section 4.2.2, does not lend itself to these traditional influence

instruments. But, instead, the institute can act as an intermediary between local businesses and the open source community. The portal-like system can collect requirements from various businesses, prioritise them, and communicate them to the OSS communities.

6.2.2 Raising the Level of Awareness of OSS

Most OSS does not have the financial backing of large profitable enterprises to actively market them. However, it is crucial for them to be accepted, tried, and trusted by users. The SFU-OSI will never have the multi-billion dollar advertising and training budgets of large corporations, and such investments do no fit within SFU's mission as a public university – however, that should not prevent the University from raising the awareness and knowledge of OSS. The SFU-OSI can sponsor conferences, seminars, users groups, and networking events to bring users and non-users of OSS together to share their experience. This would encourage trial and word of mouth, which is the next best option in the absence of active marketing.

A solid step towards spurring new jobs is to allow young entrepreneurs to turn their ideas into commercial entities. As explained in section 4.1.1, open source software can be used by new ventures to build viable solutions with less effort than building everything from scratch. The SFU-OSI can create an "Incubator"-like initiative, which will allow new businesses to emerge, based on OSS. While the SFU-OSI may not have the financial resources to fund new ventures, all that is needed to start the dialogue and the flow of ideas among the above groups are resources such as computer labs to test demo software and rooms for brainstorming.

6.2.3 Closing the Chasm

The SFU-OSI cannot reconcile the disagreement between OSS advocates and proponents of proprietary software in the short term. Some of this disagreement is due to conflict of interest and there is not much that can be done about it. However, part of the disagreement is largely due

to misunderstanding. Hosting various events as explained in section 6.2.2 will start a dialogue between the two camps and hopefully reduce the misunderstanding over time.

6.2.4 Reducing Skills Gap

As explained in section 4.2.1.1, effectively implementing, contributing, and leveraging OSS requires deeper knowledge and a broader skill set. The SFU-OSI can work with the School of Computing Science and the Faculty of Business Administration to provide a wide rage of workshops for students, IT professionals, and entrepreneurs. These workshops should cover areas shown in Table 3:

Table 3 - Proposed Training

Training	Audience	Goal
Technical Skills	Technology Enthusiasts	Enable them to make sense of OSS and to contribute to it.
Social Skills and Open Source Culture	Programmers and Project Managers	To enable them to effectively work in the open source community
Management Skills	Senior programmers, IT managers, Project managers, and MIS Students	To enable them to strategically evaluate Open Source and Proprietary solutions, lead teams, and implement solutions
Legal Knowledge	IT managers, CEOs, CTOs, CFOs, and CIOs	To enable them to make more educated decisions regarding the use of OSS
Entrepreneurial Knowledge	Young entrepreneurs	To enable them to successfully commercialize any kind of technology including OSS.

6.2.5 Working with Governments

The federal government has acknowledged that OSS provides significant opportunities for governments. It has taken several OSS-related initiatives, including one to create a level playing field for OSS (Chief Information Officer Branch, 2004). It is equally important for the OSS community to understand the role and the needs of the governments. The SFU-OSI can actively work with the government to conduct research on its behalf on the usage of OSS; this

will benefit both the government and the OSS businesses. The SFU-OSI can also actively solicit the opinion of decision makers and regulators within the government and share the results with the OSS community in order to make sure OSS continues to meet governments' needs.

6.2.6 Making OSS User-Friendly

One of the highly cited reasons for not adopting OSS is that it tends to be less user-friendly than proprietary software. This is because, historically, only programmers have been involved in open source projects and they were not interested in features other than those that solved their immediate needs. Proprietary software companies hire other experts, such as graphic designers, technical writers, and behavioural scientists to make software user-friendly. We believe that the SFU-OSI can undertake several similar initiatives to make OSS user-friendlier. First, the Institute can develop guidelines, standards, and templates for making user interfaces and writing how-to documents. This would enable independent contributors to follow guidelines and create a consistent look and feel for open source applications and documents. Second, the SFU-OSI can start an initiative similar to the "OpenLaw[10]" initiative by Harvard Law School to attract some of these non-programmer experts to brainstorm and contribute to making open source software more user-friendly. For example, the SFU-OSI can create a portal similar to Wikipedia[11], but dedicated to user-friendly documentation of open source software. This would allow programmers as well as non-programmer experts to document various OSS in a peer-reviewed environment with a consistent look and feel.

6.2.7 Documenting Innovation

The SFU-OSI, in collaboration with faculties of computing science, business administration, and social sciences, can research open source software in a few different ways.

---

[10] Berkman Center for Internet and Society, 2004. "OpenLaw" is an open forum for crafting legal arguments available to all Internet users, lawyers and non-lawyers.
[11] Wikipedia, 2004. Wikipedia is an Internet free encyclopedia.

For example, it is believed that substantial amounts of technical knowledge and innovation are embedded in many mature open source applications. By mining these projects for knowledge, the SFU-OSI can document and perhaps reuse innovations that otherwise may be forgotten. This can lead to new businesses and improved products.

6.2.8 Other Possible Roles for the Open Source Institute

The SFU-OSI can also study some of the managerial issues of open source software development and create new knowledge for successful execution of open source projects. On a purely social level, the SFU-OSI can study the open source movement, including software development, reputation systems, power and influence mechanisms, etc., to help prepare new generations of developers that will be better equipped to interact with both the open source and the proprietary software communities.

APPENDIX– INTERVIEW QUESTIONS

1. How would you describe your organization's level of awareness of OSS?
2. Do you think there are enough Open Source Software developers and supporters (in your company and in the market) to make OSS a viable alternative to closed source?
3. Are there any viable OSS applications for your industry?
4. How do you perceive the risks of running OSS in your company?
    a) How do you compare them with the risks of running proprietary software?
5. What are the barriers to the adoption of OSS in your organization?
6. How do you see the prospects of OSS?
7. Have you ever made a case for or against the use of OSS?
8. What can THE SFU Open Source Institute do to promote the use of Open Source in the local business community?

REFERENCE LIST

Baye, M.R. (2002). *Managerial Economics & Business Strategy (4th ed.)*. McGraw-Hill Companies

Berkman Center for Internet and Society. (2004). *OpenLaw*. Retrieve November 10, 2004 from http://cyber.law.harvard.edu/openlaw/

Bretthauer, D. (2002). *Open Source Software: A History. Information Technology and Libraries*. 3-10

Chieft Information Officer Branch. (2004). *GoC Proposed Position on Open Source Software and Next Steps*. Retrieved November 17, 2004 from http://www.cio-dpi.gc.ca/fap-paf/oss-ll/oss-ll/oss-ll_e.pdf

Christensen, C. M., & Anthony, S. D., & Roth, E. A. (2004). *Seeing What's Next, Using the Theories of Innovation to Predict Industry Change*. Harvard Business School Press, 1-27

Clemons, E.K., (1991). *Investments in Information Technology.* Communications of the ACM, Volume 34 No. 1, 23-36

e-Cology Corporation (2003). *Open Source Software in Canada.* Retrieved November 11, 2004, from http://www.e-cology.ca/canfloss/report/Canfloss_Report.pdf

Fichman, R. G., & Kemerer C. F. (1993). *Adoption of Software Engineering Process Innovation: The Case of Object Orientation*, Sloan Management Review, Winter 1993, Volume 34 Issue 2, 7-22

Free Software Foundation. (1991). *GNU General Public License.* Retrieved October 27, 2004, from http://www.gnu.org/licenses/gpl.txt

George, J.M., & Jones, G.R. (2002). *Organizational Behaviour, Third Edition.* Prentice Hall

Gosh, R. A. (2003). *License Fees and GDP per Capita*. Retrieved November 10, 2004, from http://www.firstmonday.dk/issues/issue8_12/ghosh/

Gosh, R. A., Krieger, B., Glott, R., Robles, G. (2002). *Open Source Software in the Public Sector: Policy within the European Union*. Retrieved October 15, 2004, from http://flossproject.org/report/FLOSSFinal_2b.pdf

Hax, A.C., Wilde, D.L. (1999). *The Delta Model: Adaptive Management for a Changing World.* Sloan Management Review, Winter 1999, Volume 40 Issue 2, 11-28

Hill, C.W.L., (1997). *Establishing a Standard; Competitive Strategy and Technological Standards in Winner-Take-All Industries.* Academy of Management Executive, Volume 11 No. 2, 7-25

Johnson, P. J., (2002). *Open Source Software: Private Provision of a Public Good.* Journal of Economics & Management Strategy, Volume 11 No. 4, 637-662

Kogut, B., & Metiu, A., (2001). *Open Source Software Development and Distributed Innovation.* Oxford Review of Economic Policy, Volume 17, No. 2, 248-264

Konig, J., (2004). *Seven Open Source Business Strategies for Competitive Advantage.* Retrieved October 18, 2004, from http://management.itmanagersjournal.com/management/04/05/10/2052216.shtml?tid=85 La Monica, M. (2004). *AT&T Looks Into Closing its Windows.* Retrieved Oct 27, 2004 from http://news.zdnet.com/2100-3513_22-5397748.html

Lerner, J. & Tirole, J. (2002). *The Simple Economics of Open Source.* The Journal of Industrial Economics, Volume L No.2 (June 2002), 197-234

Microsoft. (2004). *Shared Source Initiative.* Retrieved October 27, 2004, from, http://www.microsoft.com/resources/sharedsource/Government/opensource.mspx

Mustonen, M. (2002). *Copy-left-Economics of Linux and Other Open Source Software.* Information Economics and Policy, 15 (2003), 99-121

Netcraft. (2004). *October 2004 Web Server Survey.* Retrieved October 27, 2004, from http://news.netcraft.com/archives/web_server_survey.html

O'Reilly, T. (2004). *The Open Source Paradigm Shift.* Retrieved Oct 13, 2004, from http://tim.oreilly.com/opensource/paradigmshift_0504.html

Open Source Initiative. *The Open Source Definition (version 1.9).* Retrieved Oct 1, 2004, from http://www.opensource.org/docs/definition.php

OpenOffice.org. *Product Information.* Retrieved October 28,2004, from http://www.openoffice.org/product/

Rogers, E. M., (1983). *Diffusion Of Innovation*, New York Free Press

R-smart Group. (2004). *Open Source-Opens Learning.* Retrieved October 20, 2004 from http://www.rsmart.com/assets/OpenSourceOpensLearningJuly2004.pdf

SourceForge.net. (2004). *Development Status.* Retrieved October 22, 2004 from www.SourceForge.net.

Treasury Board of Canada Secretariat. (2004). *Vision.* Retrieved October 31, 2004, from http://www.cio-dpi.gc.ca/fap-paf/documents/vision_e.asp.

Treasury Board of Canada Secretariat. (2004). *Open Source Software Position.* Retrieved October 31, 2004 from http://www.cio-dpi.gc.ca/fap-paf/oss-ll/position_e.asp.

Weiss, T. (2003). *LinuxWorld: Unilever Moving to Linux for Global Operations.* Retrieved November 12, 2004, from http://www.computerworld.com/softwaretopics/os/linux/story/0,10801,77816p2,00.html

Wheeler, D.A. (2004). *Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!* Retrieved October 5, 2004, from http://www.dwheeler.com/oss_fs_why.html

Wichmann, T. (2002). *Use of Open Source Software in Firms and Public Institutions.* Retrieved October 15, 2004, from http://www.berlecon.de/studien/downloads/200207FLOSS_Use.pdf

Wikipedia, (2004). Retrieved November 10, 2004 from http://en.wikipedia.org/wiki/Main_Page