

Determinants of Open Source Software Adoption

Final Thesis

Maastricht University
Faculty of Economics and Business Administration
Maastricht, July 2005
van der Luer, T.L.H.
i029440
International Business
Final Thesis
Thesis supervisor: Drs. M. Vluggen

Abstract

This thesis presents a descriptive study towards the determinants of the adoption of Open Source Software (OSS) in for-profit organizations. Based on a literature review on open source and OSS in general, in combination with a review of research in technology innovation adoption and diffusion, several relationships are hypothesized between expected influential factors and OSS adoption. In total, twenty-two relationships are proposed.

Based on a survey among IT managers in for-profit organizations, eighty-four responses were gathered for statistical analyses. Due to the large number of variables compared to a small sample size, the number of hypotheses that could be tested in the final model had to be reduced to eight (out of twenty-two). The variables which were eventually included are: task compatibility, skill compatibility, compatibility, triability, software costs, continuity, third party support, and top management support. The dependent variable was the binary adoption decision on OSS: yes or no. Three variables were found to be relevant: Perceived task compatibility, compatibility, and triability. In addition, source code availability and software quality were found to be less important.

A possible explanation for these findings could lie in the fact that this research treats adoption as a binary decision, while many of the factors which have not been found relevant might appear important at later stages than the adoption stage. In addition, the classic innovation characteristics are found to be important for the adoption stage of OSS: compatibility and triability.

Table of contents

Abstract	3
List of figures	7
Introduction	9
Problem statement	10
Significance of the study	10
Research questions	10
Thesis organization	10
Chapter 1 - The open source phenomenon	12
Systems to organize information	12
Software	12
Historical development in software co-operation	14
Types of software	15
Proprietary software	16
Freeware and Shareware	16
Commercial OSS	16
Non-commercial OSS	17
Open source licenses	17
Copyleft licenses	17
Non-Copyleft licenses	17
OSS in this paper	18
OSS development issues	19
Authority by competence	19
Participative leadership	20
Modular project structures	21
Parallel release policy	21
Motivation credit policy	21
Transparent community organization	22
Tools for communication and support	22
OSS Usability	23
OSS demand & supply	24
Demand for OSS	24
Supply of OSS	26
Total cost of ownership	28
Importance of third party support	29
Conclusion	30
Chapter 2 - Adoption and diffusion of innovations	31
The innovation process	31
Innovation adoption, diffusion and infusion	31
Innovation diffusion research	32
Classical innovation diffusion works	32
Diffusion of Innovations (DOI)	32
Innovation characteristics	33
Extending the classical DOI theory	34
Managerial influences	34

Organizational adoption	34
Network externalities	35
Knowledge barriers	37
Environmental factors	38
IT innovations	39
Diffusion of IT innovations.....	39
Frameworks for organizational diffusion of IT.....	41
IT adoption contexts.....	42
OS as an IT innovation.....	43
Conclusion.....	45
Chapter 3 – OSS adoption: Framework, determinants, and hypothesized relationships	46
Open source adoption model.....	46
Classification scheme for OSS adoption determinants	47
TEO scheme	47
IT diffusion and assimilation scheme.....	48
Factors influencing OSS adoption.....	49
OSS - organization combination	49
Organizations & adoption environment.....	56
Overview of hypotheses	59
Conclusion.....	60
Chapter 4 – Research methodology	61
Research design.....	61
Operationalization of variables	61
Data collection procedure.....	64
Sample.....	64
Instrument validation.....	67
Reliability	67
Validity.....	68
Conclusion.....	69
Chapter 5 – Results	70
Descriptive statistics.....	70
Independent variables.....	70
Dependent variable.....	72
Multivariate analysis	73
Data issues.....	73
Results	76
Alternative regression analyses.....	77
Conclusion.....	78
Chapter 6 - Discussion of results	79
Discussion	79
Relevant variables in the model	79
Irrelevant variables in the model.....	80
Discussion of the model	80

Implications for management.....	83
Limitations	84
Suggestions for future research.....	85
Bibliography.....	87
Appendices	93
Appendix A: Open source timeline.....	93
Appendix B: The OSD License.....	95
Appendix C: Overview of software licenses.....	97
Appendix D: Motivations of open source developers.....	98
Appendix E: Changes in the strategic value of applications.....	99
Appendix F: Sample of successful open source software.....	100
Appendix G : The questionnaire	102
Appendix H: Selection of independent variables.....	113
Appendix I: Hypothesized relationship of OSS licenses	114
Appendix J: SPSS Output	115

List of figures

- *Figure 1.* The software production process (in the case of Java, from source code to running program). From: *Computing concepts with Java* (p. 27), by C. Horstmann, 2003, New York: John Wiley & Sons, Inc. Copyright 2003 John Wiley & Sons, Inc.
- *Figure 2.* Categorization of software. From: *FLOSS final report - part 3: free/libre and open source software: survey and study. Basics of open source software markets and business models* (p. 4), by Berlecon Research GmbH., 2002, Berlin: Berlecon Research. Copyright 2002 Berlecon Research.
- *Figure 3.* Software value chain. From: *FLOSS final report - part 3: free/libre and open source software: survey and study. Basics of open source software markets and business models* (p. 23), by Berlecon Research GmbH., 2002, Berlin: Berlecon Research. Copyright 2002 Berlecon Research.
- *Figure 4.* The applications portfolio. From: *Strategic planning for information systems* (p. 301), by Ward, J. & Peppard, J., 2002, Chichester: John Wiley & Sons, Ltd. Copyright 2002 John Wiley & Sons, Ltd.
- *Figure 5.* Software product categories. From: *FLOSS final report - part 3: free/libre and open source software: survey and study. Basics of open source software markets and business models* (p. 30), by Berlecon Research GmbH., 2002, Berlin: Berlecon Research. Copyright 2002 Berlecon Research.
- *Figure 6.* Determinants of available OS applications. From: *The cathedral versus the bazaar (With apologies to Eric S. Raymond): An economic and strategic look at open-source software* (p. 6), by Blecherman, B. (1999). Copyright 1999 B. Blecherman.
- *Figure 7.* Application portfolios in different contexts. From: *Strategic planning for information systems* (p. 304), by Ward, J. & Peppard, J., 2002, Chichester: John Wiley & Sons, Ltd. Copyright 2002 John Wiley & Sons, Ltd.
- *Figure 8.* IT innovation adoption context. From: *Information technology diffusion: A review of empirical research* (p. 195-206), by Fichman, R.G., (1992). Copyright 1992 R.G. Fichman.
- *Figure 9.* Model for open source adoption. From: *A conceptual model for enterprise adoption of open source software* (pp. 274-301), by Kwan, S.K., & West, J., (2005). In S. Bolin (Ed.), *The standards edge: Open season* Ann Harbor, Michigan: Sheridan Books. Copyright 2005 Sheridan Books.
- *Figure 10.* Classification scheme of determinants of OSS adoption. From: *The diffusion and assimilation of information technology innovations*, by Fichman, R.G., (2000). In R.W. Zmud (Ed.), *Framing the domains of IT management: Projecting the future through the past* (Chapter 7). Cincinnati, OH, Pinnaflex Educational Resources, Inc. Copyright 2000 Pinnaflex Educational Resources, Inc.

- *Figure 11.* Platform standards. From: *Why firms adopt platform standards: A grounded theory of open source platforms* (presentation, sheet 10), by West, J., & Dedrick, J. (2003). Copyright 2003 West, J., & Dedrick, J.
- *Figure 12.* Open source adoption rates.
- *Figure 13.* Shifts in strategic importance of applications in the applications portfolio. From: *A conceptual model for enterprise adoption of open source software* (pp. 274-301), by Kwan, S.K., & West, J., (2005). In S. Bolin (Ed.), *The standards edge: Open season* Ann Harbor, Michigan: Sheridan Books. Copyright 2005 Sheridan Books.

Introduction

The software market exhibits certain, rather strange, or at least, unexpected, effects these days. Software in many categories, for many different tasks, and many different users, is freely available without the need to make *any* kind of financial commitment. Just download the software, install and freely use it. Questions or problems? Just visit the website of the program and present your question. Probably you will get a proper answer within 48 hours.

The phenomenon that is referred to just now is *open source software*. Open source software refers to all kind of computer programs that are distributed under certain non-restrictive licenses. The inner workings (i.e. source code) of these programs are for anyone to see, change, modify, and even distribute. With the Linux operating system as its most known creation by the public, open source software has been relatively unheard of by the great majority of computer users.

Clearly, open source software offers some advantages over other kinds of proprietary licensed software. Often mentioned are direct and indirect advantages following from the availability of the source code (e.g. customization, safety, reliability) and the absence of license fees. However, open source did not have the level of attention of businesses that it maybe should have been given for a long time.

Currently the number of people, organizations, and business that are consciously aware of the developments and opportunities offered by this movement is growing fast, not unimportantly spurred by the internet. Since 1998 this interest for open source software and the open source software development process has gained increased attention by many types of organizations. Three factors contributed to this rise in interest (Lerner & Tirole, 2001):

- Rapid diffusion and, in some cases, dominance, of a number of open source software packages. Examples are the Linux operating system and Apache web server.
- Large capital investments by major industry players in open source software.
- The innovative organizational structure of open source development projects.

Although there are aspects that favor the use of open source software it is not really clear what influences the adoption and diffusion of open source software in for-profit organizations. Why does a firm adopt a certain open source software package? What aspects of open source products provide *the* argument for a firm to adopt? How are they related to the size or structure of a firm? What differentiates open source from other software and makes it so innovative and attractive to adopt? This leads to the problem statement of this thesis.

Problem statement

The problem statement of this thesis follows from the discussion above:

Which factors influence the adoption of open source software among for-profit firms?

Significance of the study

While this problem statement relates to a vast amount of literature and research in the field of technology innovation and diffusion (Rogers, 1995; Tornatzky & Fleisher, 1990; Fichman, 1992), and research on open source software has been increasing over the last years (Bonaccorsi & Rossi, 2003a, 2003b; Lerner & Tirole, 2000; Krogh & Hippel, 2003; West, 2003), there has not been a lot of empirical research in this area, with the notable exceptions of Chau & Tam (1997), Berlecon (2002a, 2002c), and Ghosh & Glott (2003). Therefore, this thesis is expected to add some useful insights to the existing research literature on this topic.

Research questions

To guide the discussion and provide a pathway for finding a satisfying answer to the problem statement the following research questions are proposed:

1. What is open source software ?
2. To what extent is open source software an innovation ?
3. To what extent is the adoption and diffusion of information technology innovations different from other types of innovations (i.e. not information technology related) ?
4. Which determinants have been identified as influencing the adoption and diffusion of innovations ? How can they be categorized ?
5. Which determinants are likely to have a significant influence on the adoption and diffusion of open source software ?

Thesis organization

Chapter 1 discusses the open source phenomenon by reviewing literature on the subject. It explains how open source came about, what it does embrace and what it does not. The chapter outlines the reasons for its success so far, the co-operative software development process in open source projects, open source communities, and the role of for-profit organizations and changing market dynamics. In summary, it will deal with research question one.

Chapter 2 contains a review of the major relevant literature studies on innovation diffusion literature. Explicit attention is paid to research concerning information technology

innovations. The chapter identifies and lists the determinants for information technology diffusion. Chapter 2 deals primarily with research question 2, 3 and 4 (partly).

Chapter 3 relates chapter 1 and 2, that is, it puts open source software in the light of innovation literature and looks how it is related to information technology innovations. It will connect the determinants of innovation diffusion in chapter 2 to open source software by means of a number of hypotheses. This chapter deals with research question 5.

Chapter 4 describes the research methodology used to research the hypotheses brought forward in chapter 3. Details are provided regarding the study's population and the sampling techniques used to identify them. Also, there will be an elaboration on the data collection method and the data analysis procedures, including illustrations of how they were applied.

Chapter 5 presents the results of the research and the extent to which the hypotheses could be confirmed or not.

Chapter 6 concludes this thesis with a review of the main findings. Moreover, it proposes a number of practical implications following from these findings. Finally, the limitations of this study as well as some suggestion for further research are made.

Chapter 1 - The open source phenomenon

This chapter introduces the reader into the main ideas of the concept of open source software (hereafter OSS). A short review will be presented of the traditions in co-operative software development followed by an explanation of OSS and licensing schemes that come with it. Following, a discussion of several open source (hereafter OS) issues like the OS development process, OS communities, and the types of OSS available. In the end a short conclusion will be provided. First however, some background information into information systems and software.

Systems to organize information

Information systems have been used by businesses for a long time to face a range of problems. The following definition of Whitten, Bentley & Dittman (2001, p.8) is used to define an information system as: “*An arrangement of people, data, processes, information presentation, and information technology that interact to support and improve day-to-day operations in a business as well as support the problem-solving and decision-making needs of management and users.*”. The enabling IT consists of the different though highly related aspects: hardware, software, and telecommunications technology.

As information systems developed over time so did the technology that provided for these information systems: software, hardware, and telecommunications. Telecommunications provide for the means to interlink different systems and clusters of systems to create networks like the internet. Hardware includes the physical components that make up a computer system. Software, simply stated, is a set of bundled computer instructions that tell the computer hardware what to do. This software consists of source code that can be interpreted by a computer machine. Since the concepts of software and source code are relevant to this discussion, the next section will discuss these subjects in more detail.

Software

Software is not a physical product, but a set of manually typed instructions that tell a computer what to do. At the machine level software consists of lines of code consisting of merely ones and zeros, referred to as *binary code*. Composing such code is not very human-friendly and for that reason higher level programming languages have been developed. These languages contain the logic and structure of a program in somewhat comprehensible terms. Lines of code in such a programming language are called *source code* (see figure 1).

Large software packages contain millions of lines of such manually typed code. Before computers can interpret source code it has to be compiled into binary code since a machine can only read ones and zeros. A graphical representation of this process in the case of a Java¹ program is shown in figure 1.

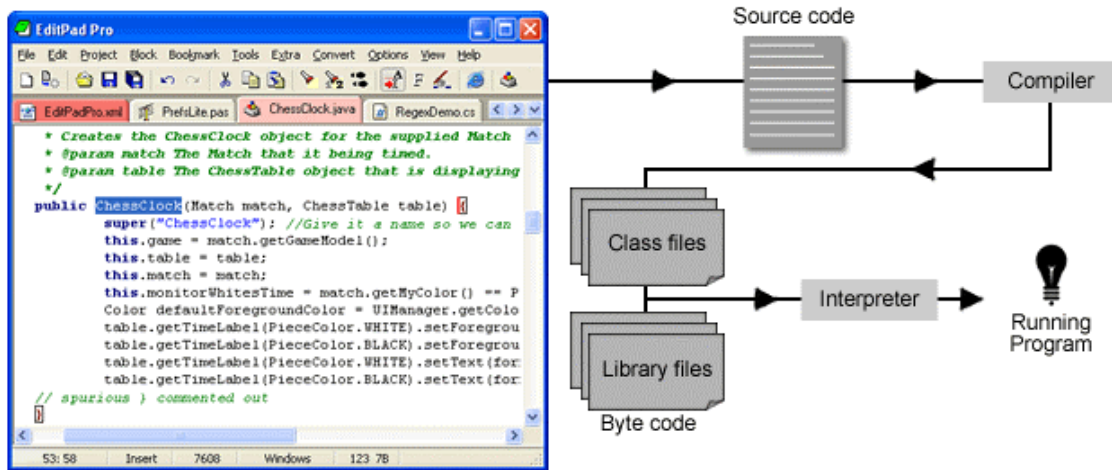


Figure 1. The software production process. In the case of Java, from source code to running program

Software comes in two types. The first is application software. These are programs most of us are familiar with and use in everyday tasks like a word processor, spreadsheet or internet browser. The second is system software², which acts as a bridge between a system's hardware and application software. An example would be the famous Windows operating system (Turban, McLean & Wetherbe, 1999).

Although different software programs are not exactly the same in terms of functionality, layout, or programming language, still duplication occurs in programming. Source code, and its underpinning ideas, can be reused to a certain extent. On the other hand, the one-time investment needed for the development of a program becomes very high for even moderately complicated software packages. Given the ease by which duplication can be done, copying a program's source code becomes very lucrative. Therefore intellectual property rights are extremely important. Licensing policies commonly determine what can, and cannot, be done with the purchased software. Software licenses exist in many forms, but usually have to be bought for every individual end-user or machine.

However, purchasing a commercial license in most cases does not mean that you also get the source code of the software you purchased. A license commonly only includes a working version of the program based on the binary code. This binary code is useless in terms

¹ Java is a programming language.

² Also referred to as platform or operating system.

of providing insights into the workings of the program. It could be somewhat compared to going to a restaurant: you get the dish you ordered, but you do not get the recipe.

Since the marginal cost of sharing a digital good like a set of lines of code is almost zero, sharing code would make sense as to minimize the overall duplication of effort among software developers. This co-operation is the topic of the next section.

Historical development in software co-operation

The fact that programming is a time consuming effort on the one hand and that it is easy to share and duplicate on the other hand would make sense for co-operation and code sharing between software developers. As a matter of fact such co-operation has existed for a long time and some periods in co-operation undertakings can be identified³ (Lerner & Tirole, 2002).

During the first period, from the early 1960s to the 1980s, developing software was largely an academic undertaking. Sharing source code was common practice among programmers. The main focus was on developing an operating system for multiple hardware platforms. This resulted in the Unix operating system and the C programming language. These development efforts were shared among the development community, greatly stimulated by the computer network Usenet. Software was distributed on basis of a nominal charge without any claims on property rights of the source code. In the early 1980s problems began to arise when AT&T began enforcing its intellectual property rights on Unix. Users had to pay license fees to employ Unix. IBM, HP, and DEC followed AT&T by also starting to develop proprietary versions of the Unix operating system (Lerner & Tirole, 2002). In addition, many developers moved from universities to private software development firms, where they were bound to non-disclosure agreements (Nuvolari, 2003).

As a response to these developments, some efforts were undertaken to formalize the co-operative process. This is the second period, which took place from the 1980s to the early 1990s. The Free Software Foundation was founded in this era, which put forward some major developments to keep co-operative developed software from getting under commercial licenses. It came up with the GNU General Public License (hereafter GPL), which required users to agree with the fact that the source code should be kept open. This restriction was not only applicable to the original piece of software, but also for all other future developed enhancements or extensions which in some way included the original source code. This so-

³ This discussion merely provides as a background to this thesis topic and is neither complete nor meant to be complete. An overview of the main facts is given in appendix A. For a more complete discussion on the history of collaborative software development see Lerner & Tirole (2002), G. Michalec (2002), Wayner (2000), or Levy (2001).

called ‘viral’ or ‘Copyleft’ effect has not made it useful for commercialization, since it rules out any business models based on license fees. At the same time, OSS projects developed a set of organizational characteristics which were used to guide the development process, including strong leadership and a core set of source code developers (Lerner & Tirole, 2002).

		Source code open ?	
		Yes	No
Price for the user	0 (Gratis)	Non-Commercial open source software Examples: Linux & Apache	Freeware Shareware Demoware Example: Free level of shoot 'm up game
	> 0 (non gratis)	Commercial open source software Example: Sugar CRM	Proprietary (Commercial) Software Example: MS Office

Figure 2. Categorization of software

The third period was spurred by the introduction of the internet and the world wide web during the early 1990s. They greatly enhanced co-operation and collaborative development efforts. In addition to the growth in the number of projects, commercial companies also became interested. They started to provide support and consultancy services for the freely available software programs. Besides, commercial companies were introduced at the actual development of these programs, caused by the coming about of new types of licenses. The ‘Debian Social Contract’ initiated this in 1995. The Debian Social Contract allowed for less restrictive licenses which were not ‘viral’ in nature like the GPL license. Thereafter it was possible to complement co-operatively developed code with proprietary code for open source software released under such terms, and logically this was much more interesting for commercialization (Lerner & Tirole, 2002). The next section will highlight the different types of software in terms of price and source code availability.

Types of software

OSS is frequently referred to as ‘free’ software. However, ‘free’ is often meant as in ‘free speech’ and not as in ‘free beer’⁴. Thus, free is a matter of freedom, not cost. The extent to which OSS can be regarded as free as in ‘free beer’ is dependent on the license it is distributed with and the purpose it serves. However, there is also proprietary software that is free, and commercial software which is open source.

Commercial software is software developed and sold by a firm in order to make a profit. Often this software is proprietary, but that is not always the case. Thus, ‘commercial’ is not a stand-in for ‘proprietary’ (Fugetta, 2003). The different types of software classified by

⁴ Read more on free software on the website of The Free Software Foundation: <http://www.gnu.org/philosophy/free-sw.html>

source code availability and price can be seen in figure 2. Next, a brief explanation of each category's characteristics.

Proprietary software

Most commercial software is proprietary, which means it costs money. Often its source code is closed and modification or unauthorized distribution are forbidden. An example would be the well-known Microsoft Office suite.

Freeware and Shareware

As the name implies, freeware means software without any financial cost. However, it comes without the source code, is still copyrighted, and its modification tends to be limited. Distributors of this kind of software hope to gain a large user group by providing the software for free. They hope it enables them to increase their market share and create a larger user group. In effect, that would provide leverage opportunities to e.g. provide complementary products or services, or try to become a standard and push competing products out of the market. For example, several companies provide free versions of their firewall and anti-virus software. This software is fully functional but does not contain any advanced features. Shareware (also called demoware) is comparable to freeware except that it can be used for only a limited period of time or with limited functionality.

Commercial OSS

For-profit firms sometimes publish the source code of their software products to the public. However, the software itself is not for free. West (2003) discusses the reactions of 3 major players in the platform market (IBM, Sun and Apple) to the OS developments. Competition have urged these companies to adopt hybrid strategies to "*find the right compromise between totally proprietary platforms and totally open ones*" (West, 2003).

Examples of such hybrid strategies as discussed by West (2003), are opening parts of the proprietary platforms (Apple) and making them partly open (Sun). Advantages gained by such strategies are faster adoption rates, increases in the number of interoperable products, and improvements by large sophisticated users that directly accrue to the proprietary platform.

Another expression that fits this category is 'Shared Source'. Shared Source is the answer of Microsoft towards OS. It allows certain large users (e.g. universities) to study the source code of Microsoft products. Yet, it does not allow for modification.

Non-commercial OSS

This category holds all the software giving free access to the source code, including the right to modify and distribute it. The major distinction in this category is between software under viral licenses (GPL like), and non-viral licenses that often fall under the open source definition (hereafter OSD). These two OS licensing schemes are explained in the following section.

Open source licenses

Whether OSS should be released under Copyleft licenses or under non-Copyleft licenses has always been a debate among advocates of OS (Agrain, 2002).

Copyleft licenses

Copyleft licenses present that “*once a program is licensed by a developer the subsequent programs based on the original must also be licensed similarly*”, Mustonen (2003) as cited in Bonaccorsi & Rossi (2003a). As mentioned above, the most well known Copyleft license is the GPL license. However, the term ‘viral’ needs some more explanation. Software does not automatically get contaminated when it derives on GPL licensed software. It has to modify the original work some way. Therefore it would be more descriptive to express the license as persistent or to refer to it as inheritance (Bonaccorsi & Rossi, 2003a). Licenses that are also Copyleft are the Lesser GPL (LGPL) license, the Mozilla Public License, and the IBM Public License (Bonaccorsi & Rossi, 2003a). In the case of Copyleft licenses, ‘free’ is more prone to refer to ‘free’ speech, since the licenses bring significant restrictions upon any commercial exploitation.

Non-Copyleft licenses

Non-Copyleft licenses are somewhat united under the definition of the Open Source Initiative (hereafter OSI), the OSD. The Open Source Initiative was founded in 1997 and has been inspired by the ‘Debian Social Contract’. The aim of the OSI was to make OS projects more attractive for commercial software firms and to overcome the main barrier for successful exploitation: the ‘viral’ nature of the GPL-like licenses.

The OSD provides the rules to which OSS licenses should comply. The main points of the definition are⁵: Royalty free redistribution, inclusion of source code, allowance for

⁵ The OSD can be found at <http://www.opensource.org/> and is also available in appendix B.

modifications and derived works, and the allowance that all modifications *may*⁶ be distributed under the same terms as the license of the original software (Lerner & Tirole, 2001).

Today there are currently almost 40⁷ licenses adhering to the OSD. Examples are the Berkeley Software Distribution (BSD) and Apache software license. However, they only represent a limited part of the available OSS. An even smaller portion is represented by Non-Copyleft licenses that are not falling under the OSD license. Appendix C provides a schematic overview of all software licenses discussed.

OSS in this paper

When using the term OSS in this paper, it refers to all software released under licenses that are accepted by the Open Source Initiative *as well as* projects under Copyleft licenses⁸. Although Copyleft licenses provide fewer opportunities for commercial firms they represent about 70% of current OSS projects (Välimäki & Oksanen, 2002). Any open source study that excludes this category would therefore severely limit itself in terms of research sample size and generalization of any results. Having outlined what the concepts of ‘free software’ and OSS licenses embrace, several OS issues will now be discussed.

⁶ They may be distributed under the same terms, but do not have to. i.e., there is no ‘viral’ aspect.

⁷ A complete and up-to-date list of OSD approved licenses can be found on the website of the Open Source Initiative: <http://www.opensource.org/licenses/>.

⁸ Terms which can create confusion are ‘open systems’ and ‘open standards’. Although somewhat related they are no synonyms for open source.

Open standards make sure that different platforms, systems, and programs are compatible, i.e. that they can communicate properly with each other although they might belong to different vendors. Open standards are documents which outline agreed upon conventions. As long as different parties adhere to these conventions, compatibility is guaranteed. Examples of open standards are image formats like GIF and PNG, or protocols like TCP/IP (Ghosh & Glott, 2003). Open standards are neutral towards software development, welcoming all and favoring none in its quest for interoperability. Whether these standards are used by OSS or proprietary software is equally possible (Watson, 2003; Fugetta, 2003).

Open systems refer to system architectures where the ‘open’ aspect stands for the fact that the interface specifications of the system’s components are fully defined and open to the public. Such openness makes system interconnectivity possible because their interfaces are known and defined. However, essentially, this is not related to open source per se.

OSS development issues

The fact that software is categorized as OS primarily relates to the license it is distributed with. However, “*OS licenses provide the governing mechanism that enforces the non-written norms of the OS community, provides incentives for programmers, and distinguishes OS from proprietary software*” (Bonaccorsi & Rossi, 2003a). Thus, the OS licenses encourage free use and keep the code open, thereby providing the necessary preconditions for OS development. Most, if not all, OS projects are internet-based networks or communities of software developers (Krogh & Hippel, 2003). And although OS development is to a certain extent informal and decentralized, it is definitely not disorganized. There are some general characteristics that belong to OSS development (Hertel, Niedner & Herrmann, 2003):

- ❑ A culture in which authority comes from competence
- ❑ Participative leadership, with clear responsibilities and delegation
- ❑ Modular project structures
- ❑ Parallel release policy
- ❑ Motivating credit policy
- ❑ Transparent community organization, with clear rules and norms
- ❑ Standardized, internet-based communication and support tools

One point is added to the original list: OSS usability. Each point will now be shortly clarified.

Authority by competence

A project commonly starts with an individual or small informal development group that constructs a program to tackle a certain problem or to fulfill a certain need. Raymond (1999)⁹ suitably describes this as “*scratching a developers itch*”. This could be for personal, business as well as intellectual reasons¹⁰.

The fact that the person(s) in the leadership role provide the first programming contributions and ideas, gives that leadership credibility. Yet over time the original development group performs less and less programming and gets more involved into project management. They provide the project with vision, split the overall project into smaller and more well defined tasks which can be solved independently, attract other developers, and try to keep the project together instead of falling apart (Lerner & Tirole, 2002).

Whether a project gets off the ground mainly depends on the interests of other developers to join the project, and whether the first release provides a critical mass of code, which shows the job can be done and that it offers value (Lerner & Tirole, 2002).

⁹ In the often cited work ‘the Cathedral and the Bazaar’.

¹⁰ A famous example is leadership role of Linus Thorvalds in the development of the Linux operating system.

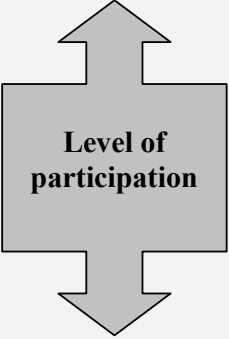
Participative leadership

Although coordination is performed by the project leader(s), to a large extent the development of the project comes from bottom-up. This is possible due to shared communication principles, standards, coordination mechanisms, and tools, which allow all involved parties to communicate effectively and efficiently.

The before mentioned article by Raymond (1999) refers to this decentralized development style as ‘bazaar style’, this as opposed to more traditional, commercial, software development, which he characterizes as ‘cathedral style’. Commercial software development tends to be strictly hierarchical and by a smaller group of developers.

The extent to which people contribute to an OS project is very diverse. There are different levels of participation. Or as Raymond (1999) notes: “*Many bazaar projects have inner and outer circles.. This simply reflects a natural gradient of interest and competence and commitment*”. Table 1 below provides an overview of the involved parties in a typical OS project and their level of involvement. Peers decide at what level a certain individual participates.

These multiple groups make up the actors in the bazaar. However, the extent to which the bazaar model and its advantages¹¹ hold for all OSS projects, especially the smaller ones, is doubtful. In addition, while OS development might have unique characteristics, proprietary software developers can copy it. Third, “*it is not proved that open source uniquely and necessarily causes software to be better, more reliable, or cheaper to develop*” (Fuggetta, 2003).

Table 1 – Participation levels in a typical OS project	
<div style="text-align: center;"> <p>High</p>  <p>Low</p> </div>	Project owner. Individual or core group responsible for design decisions concerning overall direction of the product. Contributes hundreds of hours per year to the project.
	People with write access to all or some subset of the source code. Typically secondary leaders responsible for a particular subsystem or the project. Review code submitted by others, route to project owner.
	Contribute bug fixes and small enhancements. May or may not participate on an ongoing basis.
	Use the product and debug it. Identify and report bugs. Participate in the mailing lists.
	Use the product and also suggest new features. Participate in the mailing lists.
	Use the product.

From: *A descriptive process model for open-source software development* (p. 80), by Johnson, K., 2001. Calgary, Alberta, Canada: University of Calgary, Department of Computer Science. Copyright 2001 by K. Johnson.

¹¹ Such as increased reliability, compatibility, and security.

Modular project structures

Software development is a complex undertaking, especially within the distributed OS structure. This complexity has consequences for the quality of the software and the efficiency of the production process if it is not managed properly. As mentioned before, one of the success factors of an OS project is the extent to which tasks can be broken down into independent parts. These parts have then to be presented within the software's code (Bonaccorsi & Rossi, 2003b).

The development of object oriented (OO) programming techniques provides a way to manage such complexity. This technique is a process innovation that breaks down code into objects which can be best regarded as reusable code modules. It also makes the code more comprehensible (it only requires to understand a subset of the program) and easy to change and expand. An example of a project making extensive use of such modularity is the Linux operating system¹² (Bonaccorsi & Rossi, 2003b).

Parallel release policy

Such modularity is exactly what is needed in the OS development process. Modules are dependent but can be independently developed. This is ideal for the global and loose coordination of OS projects. Ultimately they can be combined into a working program.

Such modularity enables a parallel, 'release early, release often' policy (Raymond, 1999). This means that new functionalities and bug fixes are almost constantly made available to the public.

Motivation credit policy

Developer's motivation to contribute code to an OS project has been a widespread point of debate for economists (Frank & Hippel, 2003; Krogh & Hippel, 2003; Lerner & Tirole, 2002). This is caused by the fact that OS communities mainly consist of people that contribute without being paid directly for it. Some companies do support OS projects actively or inactively, nevertheless voluntary contributions still make up the largest part of OS contributions. Several arguments have been put forward why developers do this.

Factors that have been found relevant, among others, are: personal learning and enjoyment (Krogh & Hippel, 2003), increased esteem within the community (Lerner & Tirole,

¹² Please note that modular design (i.e. object oriented code) is not a technique which is unique, or limited, to OS software. However it enables the loose structure of the OS development process to a significant extent.

2002), and signaling of quality of human capital (Bonaccorsi & Rossi, 2003b). The most important motivating factors are listed in appendix D.

Transparent community organization

The open recognition policy for contributions is part of the community's organization made public on the internet. Often, the entire project's organization can be found online, grouped towards different users. Developers, project managers, and end-users often have their own entry points in the project. All documentation is often made available, including archives on past decisions, software versions, discussions, bug databases, etc.

The fact that decisions, goals and procedures are openly stated and archived on the internet makes these projects transparent to everyone. Such transparency helps contributors to develop trust in the project and make sure that the project's governance does not suffer under ego gratification, commercial or political biases (Lerner & Tirole, 2001).

A problem with even slightly successful OS projects is often that they require significant maintenance efforts in order to preserve quality and consistency. Therefore, setting out procedures and guidelines for code contributions, bug reports, and documentation is essential. Some portals, e.g. Sourceforge, provide means for OS projects to streamline their actions in order to maximize the efficiency and effectiveness of the OS development process¹³.

Tools for communication and support

As OS projects are composed of many different parties, these parties are often also fairly geographically spread and isolated. Their primary means of communication is presented by the internet, which offers the opportunity for asynchronous communication: one party does not have to be available when the other party communicates.

Besides the fact that the internet infrastructure itself is partly based on OS creations, various tools are used within OS projects. They provide for the cost-effective coordination and communication within the OS community. Some of the most important tools are (Johnson, 2001):

- ❑ Mailing lists, newsgroups, and forums, which provide for communication.
- ❑ Programming support tools like code version management tools (CVS), source code compilers and debuggers, and other code utility programs (e.g. Unix diff).

¹³ SourceForge.net provides automated mailing lists, server space, problem report databases, etc. to open source projects. It requires no financial compensation. Their mission is to “*enrich the OS community by providing a centralized place for OS developers to control and manage OSS development*” (www.sourceforge.net).

- ❑ Secure protocols (SSH).

Although these tools are not the subject of this paper and might seem to be of merely technical importance, in fact they almost form a prerequisite for the OS process to take place. To a large extent these tools reflect the governance and decision making process in OS projects¹⁴. Another aspect to OS development is the lack of emphasis on usability issues.

OSS Usability

Usability is related to the complexity and relative advantage, which a software product poses to the end user (Fichman, 2000). Usability is often described in five terms: ease of learning, efficiency of use, memorability, error frequency and severity, and subjective satisfaction (Nichols & Twidale, 2003). Open source products generally do not have a good reputation in providing usability to the average end user (i.e. the non-hacker¹⁵) as compared to proprietary software. Nichols & Twidale (2003) mention several reasons why this is the case, among which are:

- ❑ OSS developers are not typical end-users.
- ❑ Usability problems are harder to specify and distribute than functionality problems.
- ❑ OSS development generally shows a lack of formal requirements finding, analysis, and specification: This puts code design in front of interface design.
- ❑ OS projects lack the resources to undertake high quality usability work.
- ❑ Strong incentive to add functionality, but community aspects prohibit the deletion of duplicate or irrelevant functionality.
- ❑ OSS development promotes power over simplicity: more and many advanced features instead of keeping the interface simple.

This list does not mean that OSS development has not paid any attention to usability issues. However, usability does seem to be an issue for many OSS projects, especially when compared to proprietary software. Nichols & Twidale (2003) note that OSS development might go to a similar ‘usability phase’ as did the development of proprietary software in the nineteen-eighties. The various development issues described have their influence on the supply and demand of OSS, which is the topic of the following part.

¹⁴ Most of these tools have been developed by the OS communities itself and fall under OS licenses (often GPL).

¹⁵ ‘hacker’ is a synonym for a technical competent person which actively displays his/her programming techniques, possibly to the OS community by contributing to one or more projects.

OSS demand & supply

The supply of OSS seems a strange economic event. Although the market price is zero, developers are still willing to contribute. This, as shown by the following discussion, can partly be explained by an extension of the traditional utility concept, towards the inclusion of reputation, recognition, and future enumeration (Blecherman, 1999). On the other hand there is demand for OSS. Since OSS is freely available, who would not want it? Obviously, there is more to software than just its price, namely the value it adds to the firms existing application portfolio. This section discusses these various issues related to OSS demand and supply.

Demand for OSS

Software value chain

The software value chain is shown in figure 3. The software value chain consist of two substantially different parts: the actual software *product* (the application), and the *services* provided next to it. Services come in the form of consulting, training, installation support, etc. Furthermore, these can be standardized (e.g. an office package) or customized (e.g. an inventory system for a warehouse) (Berlacon, 2002c). The product / service combination determines the value a certain application has to the purchasing unit.

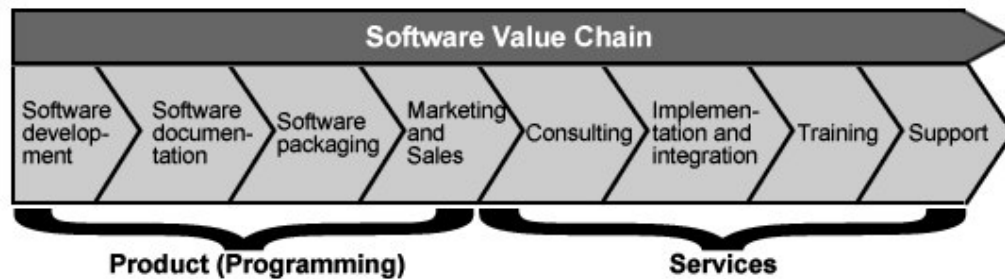


Figure 3. The software value chain.

The software value chain provides the competing ground for commercial and OSS. Which products will eventually rule the market? What type of applications are better provided by the OS community, and which by the ‘traditional’ software firms providing proprietary software? In other words, which side provides most value?

But what is the concept of value exactly? “*It is the difference between what you would pay and what it costs to make*” (Blecherman, 1999). The value added to the customer is the value the customer associates with the application and the actual production costs. This is the

so-called consumer ‘surplus’. The product providing the largest surplus simply wins (Blecherman, 1999).

How much value a specific application adds to a customer depends on the product attributes of that software. Kwan & West (2003) classify these attributes into features, risks, and costs. Ceteris paribus, firms would prefer to have the most features at the lowest risk and cost (Kwan & West, 2003). The importance of specific product attributes is largely determined by a firm’s context. Helpful insights into the context and value of a certain application within a firm can be given by means of the concept of the applications portfolio, which is discussed next.

The applications portfolio

Management’s awareness of the current information systems within the firm and their (future) contribution to the business is very important. The applications portfolio is a matrix that enables the classification of the current contributions of the available information systems to the business in a simplified way¹⁶ (Ward & Peppard, 2002). Moreover, it provides a way to compare the importance of applications between firms and the way in which that importance varies over time (Kwan & West, 2003).

Ward & Peppard (2002) compose the matrix according to the potential contribution of the IS/IT application and the degree of business dependence on that application. This gives the matrix four quadrants ranging from strategic¹⁷ to high potential, and from key operational to support.

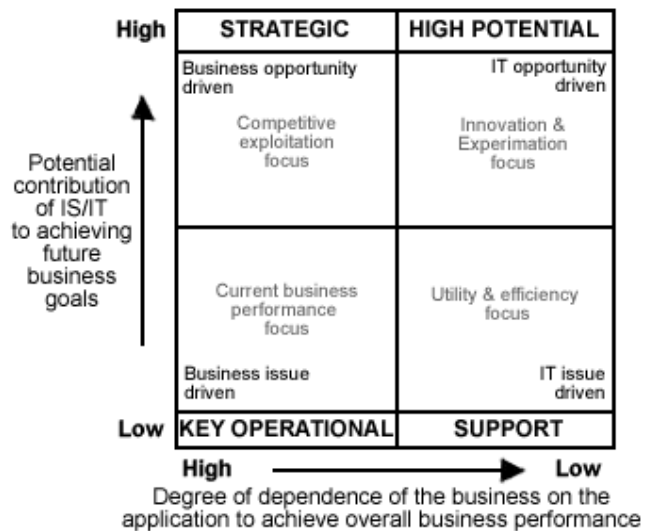


Figure 4. The applications portfolio.

The applications portfolio provides a clear overview of a company’s current and near future capabilities and needs in terms of applications. The matrix highlights where new

¹⁶ One simplified aspect is the time span. The matrix is a snapshot in time. Over time the importance and position of a certain application can, and will, change. E.g. the permanence of a competitive advantage gained by strategic application, i.e. duplication, is often not very long (Ward & Peppard, 2002). Common cycles of changing application strategic importance are shown in appendix E.

¹⁷ An example of a strategic application would be the SABRE online reservations system developed by American Airlines and IBM (Kwan & West, 2003).

applications are required and where not, and where the available products on the market can be placed according to their attributes: risk, features, and cost. Thus, the matrix highlights the areas in which there is a possible demand for applications, and how the applications available match this demand. Part of this supply of software is made up of OSS, or software that includes OS components.

It has been argued in the previous section on OS development that OSS offers some significant advantages over proprietary products in terms of, for example, functionality and reliability. Then again, OSS is not available for all software categories and quadrants. OSS that requires specialized knowledge and can be mapped to, especially, the strategic quadrant of the above portfolio tends to be limited. In which categories OSS is strong, and why this is so, is the topic of the next section.

Supply of OSS

Software categorization

Software can be categorized in many ways, e.g. application software versus systems software or desktop software versus server software. A possible categorization scheme is shown below in figure 5, which splits software in operating systems related software, enterprise solutions and standard applications. OSS tends to be underrepresented in some of these categories while being overrepresented in others (Berlacon, 2002c; Fugetta, 2003; Välkämi & Oksanen, 2002; West & Dedrick, 2003).

n
the
con
text
of
this
cla
ssif

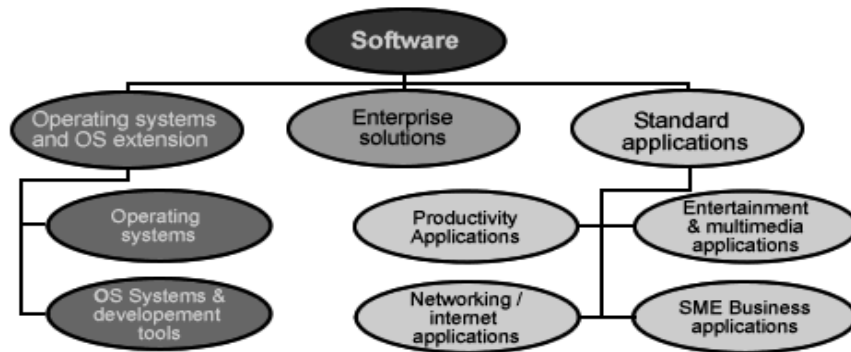


Figure 5. Software product categories.

ication, OSS is overrepresented in operating systems and operating system development tools. Also, OSS has always been strong in internet service applications like web servers and e-mail

management¹⁸. Another strong point of OSS is database management. In the other categories OSS is available, but in rather limited numbers.

Thus a logical question to arise is why the available OSS is biased towards certain software categories. This bias can be explained by referring back to the development of open source and is the subject of the next section.

Determinants of OSS supply

The discussion on the OS development process showed that advantageous characteristics associated with OSS (such as reliability, compatibility, security) to a large extent depend on the availability of a critical mass of developers. If a project cannot interest and attract other developers, the increased utility in the sense of recognition, reputation, and ego gratification do not hold and the project is likely to soon die.

Blecherman (1999) argues then, that applications with broad use and large scope apply more to the OS methodology. This is so because they require more functionality, which is an OS strength. Moreover, a broad application has a larger user group, and the chance that this group contains potential contributors is larger.

Secondly, the more the application's end-user community is composed of developers, the more likely the project will succeed. Developers will be more prone to contribute since *they* are the end-users and probably the application will require mainly the kind of knowledge *they* already possess (Blecherman, 1999).

Thus, OS applications must be in an area that can get the interest of a sufficient number of developers which are willing to contribute in order to create a sound basis for the project to continue in such terms as functionality, reliability, security, compatibility, etc., *and* utility provision. When combined this leads to the conclusion that applications which require in-depth knowledge in a certain area which is not related to general developer interests, are less likely candidates for OS efforts (Blecherman, 1999). This in turn explains why the OSS supply is limited to certain categories and is likely to be less applicable for certain application portfolio quadrants. This argument is represented in figure 6.

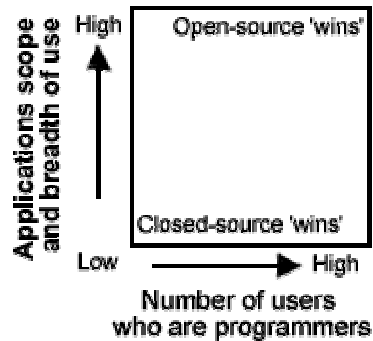


Figure 6. Determinants of available OS applications.

¹⁸ A sample of OSS products is provided in appendix F.

Total cost of ownership

The cost advantage of OSS is widely debated. Although license fees do not apply to OSS, which is a fact most agree on, other cost components are debatable. Besides license fees, software costs also include indirect and qualitative costs, which are more difficult to measure. An often-used concept to guide the discussion on costs is total cost of ownership (TCO). This concept will be briefly explained in this section.

Costs can be qualitative as well as quantitative. TCO deals with the quantitative costs. In effect, these can be direct and indirect. Thus, TCO can be seen as a measurement for all *measurable* costs of an investment¹⁹. TCO cannot provide definite answers to whether OSS or proprietary software is cheaper. TCO is very context dependent and therefore primarily useful for internal analyses (Knubben, 2004).

Therefore, TCO examinations often come up with different results, e.g. in the comparison of the TCO of Linux and Windows systems (Knubben, 2004). Although the results vary, there are some cost categories in which OSS significantly *differs* from proprietary software (Knubben, 2004; West & Dedrick, 2003):

- ❑ **Hardware:** OSS is often not dependent on a certain hardware platform. This limits vendor dependence and any excess costs related to that dependence.
- ❑ **Software:** OSS does not require any financial compensation, e.g. no license payments. License fees count for 10 to 30 percent of the total ICT budget (Ghosh & Glott, 2003). In addition, no licensing administration is required.
- ❑ **Lifecycle:** Software has a limited lifecycle. For commercial packages, support for older versions is dropped at a certain point in time. However, OSS stays often available via the communities. This postpones the update costs.
- ❑ **Operation:** OSS allows for easier process automation.
- ❑ **Technical personnel:** Personnel seems less available for Linux based systems and applications due to the different required technical capabilities. This is reason to believe that personnel costs will, at least initially, be higher.

Another issue in investment decisions on OSS versus proprietary software are switching costs. Partly because proprietary software relies to a large extent on closed standards and OSS relies on open standards, switching to OSS often brings large and one-time costs. This is especially true when organizations have to migrate from a Windows architecture to a Linux environment if they would like to adopt certain OSS.

¹⁹ TCO is therefore not limited to investments in IS/IT, but can be used for all investment decisions.

Importance of third party support

One of the prime motivators that drives OS developers is the fact that the core technical development of a project provides challenges which present intrinsic satisfaction (Bonaccorsi & Rossi, 2003b). However, not all work in OS projects provides such challenges. As discussed, usability aspects provide a weak point in OSS products, and provide an area of opportunity for for-profit firms. Examples are the development of a graphical interface, providing help and documentation to end-users, and support issues.

When considering the value chain of software products, it can be said that the OS development community primarily deals with the product development, and often only partly takes care of any service aspects. This has led to various hybrid business models where commercial firms provide for these more mundane tasks (Bonaccorsi, Giannangeli & Rossi, 2004). Additionally, especially larger firms, often require 'hard' and legal support contracts for commercial implementations of OSS (West & Dedrick, 2003).

These hybrid business models solve two issues. Commercial support complements OSS on aspects the OS community cannot or does not want to provide. This increases the adoption potential of OSS. Secondly, commercial backup for OS projects provides adopting firms with a certain continuity guarantee, which is essential for users that have to incur some level of switching cost (Bonaccorsi & Rossi, 2003b).

Conclusion

This chapter discussed the most important aspects of the OSS phenomenon. The OS development process has been considered and it has been shown that OSS *can* offer various advantages over proprietary software. However, the added value of the OS methodology seems to depend on the number of developers willing to work on a certain project, and the relative importance of each of the three product attributes. This is summarized in table 2. In addition, current OSS supply is not optimal in all software categories. Yet, this chapter did provide answers to the first research question: What is open source software?

It is important to stress for the discussion to follow that OSS is software that possesses some unique characteristics in comparison to proprietary software. These characteristics are partly caused by the development process, but this process itself is not of further interest to this thesis. This thesis only will consider how the perceived characteristics are likely to influence the adoption of the OSS product by end-users.

The next chapter will shed light on technology adoption from the perspective of innovation diffusion literature. It will also argue to which extent OS is an IT innovation.

Table 2 - Added value: Comparison of open and closed source		
Value aspect	Open source	Closed source
Features	Functionality, adaptability, security, compatibility, usability (hackers) etc.	Usability (end-users)
Risk	More reliable software No legal guarantees Source code provides sense of continuity License type determines risk for proprietary code (Copyleft aspect)	More stable in terms of legal contracts and vendor stability
Costs	No software licenses Lower hardware costs (often) Switching costs	Lower service costs
Overall critical factors	Size of the OS community (critical mass)	

Adapted from: *The cathedral versus the bazaar (With apologies to Eric S. Raymond): An economic and strategic look at open-source software* (p. 5), by Blecherman, B., 1999. Copyright 1999 by B. Blecherman.

Chapter 2 - Adoption and diffusion of innovations

Many have defined the concept of innovation, but certain aspects are more or less always included. An innovation is something that is new in its *context*. One of the works that has provided a solid and lasting foundation in innovation diffusion literature was that of Rogers (1995, original 1983). His work has become a standard, and is referred to in this paper as the ‘classical’ diffusion of innovation theory (hereafter DOI theory).

This chapter will review and discuss literature on the innovation process, and will point out the most important concepts²⁰. Also, it will make clear the extent to which classical innovation theories provide an explanation for modern information technology (hereafter IT) innovations and the OSS developments. In the end, a conclusion will be given. First, a short introduction into the concept of innovation.

The innovation process

The innovation process takes place from the moment that a novelty is found to be helpful until it reaches full potential. Innovative capacity is getting more and more important and even say that the extent to which a company can create and react to innovations will be the only sustainable competitive advantage left (Hamel & Prahalad, 1990). This is partly caused by the number of technological innovations, more specifically, IT innovations, which have been flooding the work floor over the last decade. Typically, an innovation process takes on certain steps. These steps will now be shortly reviewed.

Innovation adoption, diffusion and infusion

Rogers (p. 11, 1995) refers to an innovation as: *”an idea, practice, or object that is perceived as new by an individual or other unit of adoption... If the idea seems new to the individual it is an innovation”*. When an innovation is being *adopted* the decision to spend resources has been taken. When the innovation spreads over time, due to communication among social communities, other units can also start to adopt the innovation, which is denoted as *diffusion*. However, although an innovation might be adopted and diffused, it might not be extensively used in practice. For example, it might be a fad or unpractical (Newell, Swan & Galliers, 2000). When an innovation *does* get extensively used and integrated, an innovation is said to have reached a high level of *infusion*. The process from initial awareness to infusion is referred to as *assimilation* (Fichman, 2000). Next, an outline of innovation diffusion research.

²⁰ To guide the discussion two literature review studies by Fichman (1992, 2000) are used.

Innovation diffusion research

Researchers from various fields have worked and published on the concept of innovation diffusion. In general, DOI research focuses around three basic research questions (Fichman, 2000):

1. What determines the rate, pattern, and extent of diffusion of an innovation across a population of potential adopters ?
2. What determines the general propensity of an organization to adopt and assimilate innovations over time ?
3. What determines the propensity of an organization to adopt and assimilate a particular innovation ?

In addition, there are in general two styles of research, namely adopter studies and diffusion modeling studies. Adopter studies try to answer research questions two and three within a certain context. They primarily deal with specific organizations implementing (or not) specific innovations. Diffusion modeling studies are mainly constricted to the first research question, and try to find more results that can be generalized (Fichman, 2000).

Although Rogers' (1995) work has provided grounding concepts for innovation diffusion research there is no all-encompassing 'theory of innovation'. This can be explained by the fact that DOI research is highly contextual, and it is very hard to provide a general theory. It is therefore that most DOI research is dealing with specific technologies and adoption contexts (Fichman, 2000).

However, "*the current state of innovation diffusion research provides well-developed concepts and a large pool of empirical results that are applicable to the evaluation, adoption, and implementation of technologies*" (Fichman, 1992). Therefore, a solid discussion of the main concepts is highly relevant to this thesis. The following section will describe the more or less 'classical' concepts on the concept of innovation diffusion.

Classical innovation diffusion works

The classical innovation diffusion works embrace the basic concepts and theories laid down in, among others, the works of Rogers (1995) and Tornatzky & Klein (1982). These works will now be discussed.

Diffusion of Innovations (DOI)

The famous work of Rogers, 'Diffusion of Innovation' (1995), is an often-cited source in innovation diffusion literature. Rogers' work integrates over three thousand previous studies

in the field of technology adoption and diffusion. From the study several generalizations are widely accepted:

- **Definition of diffusion:** The process by which an innovation is communicated through certain channels over time among the members of a social system. This includes four important aspects: time, communication, the social system, and innovation.
- **Innovation-decision process:** The fact that adoption is a process with a knowledge stage, persuasion stage, decision stage, implementation stage, and confirmation stage.
- **Adopter categories:** Adopters can be more or less categorized according to the speed at which they adopt an innovation. The speed by which people innovate depends on various factors, for example personal characteristics.
- **Typical diffusion pattern:** S-shaped adoption curve. The various adopter categories adopt at differing speeds. In the beginning, adoption is slow and only by innovators. At a certain point in time critical mass is reached and growth rates are high, in the end the curve is flattened again.
- **Influence of change agents and opinion leaders:** The importance of individuals (knowledgeable and influential) in the diffusion process, convincing others to adopt and thereby accelerating adoption, are significant.
- **Innovations characteristics:** Certain perceived innovation properties determine the rate of adoption: compatibility, triability, relative advantage, observability, and complexity.

These generalizations provide a quick insight into the work of Rogers (1995). Tornatzky and Klein (1982) elaborated on the various innovation characteristics. These are discussed next.

Innovation characteristics

Tornatzky & Klein (1982) presented a 'meta-analysis of findings' of prior studies that dealt with innovation characteristics. In total they found thirty characteristics to be relevant in these prior studies. Of a total of ten which were selected, three were found to have a positive relationship with innovation adoption: compatibility, relative advantage, and complexity. These were also included in the five innovation characteristics mentioned by Rogers & Shoemaker (1971) as cited in Tornatzky & Klein (1982).

Although the classical DOI work has been critical for innovation diffusion research, it has some shortcomings, especially for the adoption of more complex innovations. For

example, the framework focuses on the diffusion of mass-produced items (Chau & Tam, 1997). The following section will provide several extensions to the classical DOI theory.

Extending the classical DOI theory

One of the drawbacks is that the DOI theory focuses too much on autonomous adoption decisions by individuals (West & Dedrick, 2003). Innovation adoption decisions are often not made by individuals since they are too big and complex to be understood by an individual user. These types of decisions require specialized knowledge before the moment of adoption (Fichman, 1992) and are influenced by the environment (Robertson & Gatignon, 1986). These shortcomings have resulted in various extensions that try to extend the classical diffusion theory towards more complex adoption situations, namely:

- Managerial influences
- Organizational adoption
- Network externalities
- Knowledge barriers
- Environmental factors

Each of these extensions will now be discussed.

Managerial influences

Individuals are often not in a situation in which they can freely choose what to do with an innovation. Guidelines and procedures are commonly in place that determine the range of choices that an employee can choose from. Within this range, certain reward and incentive systems might favor one solution for another, possibly ruling out the adoption of the innovation, based on less important or short-term criteria. Overall, in practice, the classical assumption of autonomous individual adoption decision might not be valid, since in most cases managerial influences of some kind are involved (Fichman, 1992).

Organizational adoption

Rogers (1995) notes that “*certain organizational characteristics do not have an individual counterpart*”. With this sentence Rogers refers to the fact that the innovation adoption and diffusion process at the individual level has different characteristics than at the organizational level. First, Rogers stresses the importance of individual (leader) characteristics. These can provide the edge to overcome the resistance to change and makes an organization to implement an innovation. Second, Rogers outlines the innovation process that continually takes place in organizations: initiation and implementation. Third, it is noted that

organizational size is often mentioned as a measure of organizational innovativeness. However, thereafter it is immediately argued that this is caused by the fact that organizational size is easy to measure, and is an aggregate of several dimensions that influence organizational innovation. These dimensions do not all have to be in place for finding a positive relation between size and innovativeness, and most research has not differentiated between the different aspects. Or, as Tornatzky & Fleischer (p. 162, 1990) state it, “*size has sometimes been mistakenly seen as an indicator of purely organizational traits*”.

As discussed before, the classical assumption that individuals adopt innovations for their own independent use, often does not hold (Fichman, 1992). Individuals frequently make their decisions within organizational boundaries. Within the organization, choices are limited by *organizational routines*. The individual adoption must fit in to the organizational process in order to be practical, efficient, and effective. Organizations and individuals in effect, base their decision frequently on the actions of other, interdependent, adopters. When the value of an innovation of one adopter depends on the number of other adopters, the innovation is said to exhibit network externalities (Shapiro & Varian, 1999). These are the topic of the next section.

Network externalities

Rogers (1995) provides a helpful summary of diffusion networks and discusses some fundamental concepts of network externalities, namely: opinion leaders, communication networks, personal threshold, and critical mass.

Opinion leaders are important in diffusion networks since they influence adopter's attitudes towards an innovation. Furthermore, mass media channels can directly or indirectly strengthen the influence of opinion leaders (Rogers, 1995).

The network that an individual adopter is part of and gets information from, and the number of links to adopters, is an important determinant for the innovation adoption decision. The level of interconnectedness of an individual to the network is positively related to adoption levels and adoption speed (Rogers, 1995).

The personal threshold refers to the number of other persons within an individual's interpersonal communication network that have to adopt an innovation before the individual will adopt it. Slow adopters are referred to as laggards, fast adopters as innovators. The individual threshold value depends on personal characteristics, which explains adoption speed. At the macro or system level, this is referred to as critical mass (Rogers, 1995).

The rate of adoption of aggregated individual adopters comes to a tipping point at a certain moment in time. This point is denoted as the point where *critical mass* is achieved. After that point the innovation's further rate of adoption becomes self sustaining (Rogers, 1995). This process is defined by Shapiro & Varian (1999) as positive feedback and can be explained by the fact that the overall size of the network after achieving critical mass makes the opportunity cost of not adopting the particular innovation too costly. "*Positive feedback makes the strong get stronger and the weak get weaker, leading to extreme outcomes: dominance of the market by a single firm or technology*" (Shapiro & Varian, 1999).

Whether or not a market is receptive to network externalities and tipping depends on the economies of scale and the demand for variety (Shapiro & Varian, 1999). High economies of scale will make the market more 'tippy' whereas more differentiated user needs will not. The supply side economies of scale can create barriers to entry for new entrants. Demand side economies of scale include network externalities, which create switching costs and lock-in for adopters (Shapiro & Varian, 1999). The presence of such network externalities might even lead to situations in which an innovation becomes a standard with a relatively small advantage, possibly not even related to intrinsic product quality but to for example the marketing efforts²¹ (Bonaccorsi, 2003b).

For new entrants, a market that exhibits network externalities it is almost impossible to penetrate. The value of a network depends on the number of users, or installed base, and in the case of a new party this value will be close to zero. To ignite positive feedback and start to create network externalities, the strategy can depend on 4 factors: the level of technology compatibility, performance, openness and control (Shapiro & Varian, 1999).

On the other hand, although an installed base of locked-in users might provide a serious hurdle for a new entrant, the reasoning behind lock-in ignores a set of variables that can have powerful impacts on the pattern of technology adoption (Bonaccorsi & Rossi, 2003b). First of all, changing the assumptions of the mathematical demonstration for lock-in can explain how new technology can spread in a market despite an installed base. Secondly, the 'virgin market condition', of 2 parties arriving at the same time on a market does not often hold. Thirdly, the potential market for a technology is not infinite and users will not inevitably stick to a technology due to some sunk costs made in the past. Fourth, the assumption of the independence of individual choices is not very realistic in today's network economy (Bonaccorsi & Rossi, 2003b).

²¹An often mentioned example is the adoption of the current standard for keyboards, QWERTY. More efficient alternatives like the keyboard by Dvorak, never made it (www.dvorak-keyboard.com).

In a research article on network externalities with spreadsheet software Brynjolfsson & Kemerer (1997) conclude that the installed base might be just as important as intrinsic product quality in affecting the market value of such products. This provides a direct trade-off between the release time of a product and increasing the quality in terms of product features.

All in all, new technology with a small installed base might be very attractive to a potential adopter because of certain features or characteristics. However, the technology might be so overly complicated that it provides a knowledge barrier to implementation to the potential adopter. Such knowledge barriers are clarified in the subsequent part.

Knowledge barriers

Rogers (1995) mentions the issue of an innovation's complexity and generalizes: "*The complexity of an innovation, as perceived by members of a social system, is negatively related to its rate of adoption*". But even if the adopting unit would be willing to adopt the innovation, regardless of its level of complexity, in many cases the adopter is not able to adopt the innovation due to a lack of knowledge (Fichman, 1992).

The fact that a potential adopter initially cannot make use of an innovation is due to the fact that the producer of the innovation and the adopter do not share a knowledge space large enough that the adopter immediately understands how to operate the innovation rapidly and to its fullest extent (Alavi & Leidner, 2001). It is argued that knowledge consist of tacit knowledge (mental models and know-how) and explicit knowledge (skills). In order for two parties to understand each other, the receiver should have enough tacit knowledge overlap with the sender of the information to develop the necessary skills, i.e. explicit knowledge. This difference in knowledge bases can be overcome by providing contextual information to the receiver (Alavi & Leidner, 2001). However, there is a limit to this. The receiver has a certain *absorptive capacity* (Fichman, 1992). This means that beyond a certain point it will not be feasible any more for the adopter to adopt the innovation because the learning curve is simply too high. This provides an opportunity cost, which might favor another alternative over the innovation at hand.

Because of the limited absorptive capacity of potential adopters in the case of complex technologies, Newell, Swan & Galliers (2000) pose that suppliers of such technologies²² often offer so-called 'black-box' solutions to overcome initial adoption reluctance. "*The knowledge of the innovation has been commodified, so that a set of such complex ideas can be presented as a 'thing' - a fixed entity - that can be slotted into any organizational context*" (Newell,

²² Examples of such complex technologies: MRP, ERP, BPR, JIT, TQM, CRM.

Swan & Galliers, 2000). These ideas are then spread among social communities as ‘best practice solutions’ via various channels, for example suppliers and, especially, professional associations (because of their *perceived* neutrality). After initial adoption this packaged knowledge has to be unpacked and integrated into the organization, which is often found difficult and leads to high failure rates (Newell, Swan & Galliers, 2000).

This section emphasizes again the importance of contextual factors. Complex technologies, often proposed as an all-solving remedy are not fitted for every organization in its unique context. Next to discuss are environmental factors.

Environmental factors

According to Tornatzky & Fleischer (1990) two aspects of the external environment are key determinants of innovative activity: *“the competitive characteristics of the industry, and the existence of a relevant technology support infrastructure”*.

The industry to which a firm belongs and the market it operates in, influence the adoption decision. Although there have been various studies on the influence of market competitiveness and concentration on the adoption (speed) of innovations, results have been diverse. First, firms in oligopolistic shaped markets will tend to offer innovations at higher prices and thereby slow the overall adoption rate. Second, vertical integration and coordination seem to positively influence innovation behavior due to an increased information flow (Robertson & Gatignon, 1986). Third, firms have found to innovate more at stable periods and higher market uncertainty (Tornatzky & Fleischer, 1990). Another hypothesized relationship is between business goals and the types of innovations adopted: *“Differences in competitive importance (price, quality, service) may affect the types of innovations a firm seeks out”* (Tornatzky & Fleischer, 1990). For applications, the competitive importance of different factors shows in the type of applications a firm has in the applications portfolio, as discussed in chapter one.

Robertson & Gatignon (1986) emphasize the shortcomings of the classical innovation diffusion research. They propose a model in which competitive factors in the supplier-, as well as the adopter-environment, are explicitly linked. They add to this discussion in providing some additional environmental supply side factors namely reputation of industry, R&D allocation, and marketing support.

Another important aspect is the availability of external resources and government regulations. The availability of skilled labor at a reasonable wage rate and with a sufficient educational background, and technology supply and support are essential to the adoption

decision (Tornatzky & Fleischer, 1990). On the other hand, certain innovations are simply not allowed, or are discouraged, to adopt, by means of government regulations.

Evidently, the adopting unit is not a static victim of its environment. The external environment represents the boundaries in which the innovation process takes place. The firm can influence these boundaries. How a firm operates in its environment is outlined in the firm's business strategy. This strategy has a large influence on the innovation adoption policy of the firm and whether or not a particular innovation fits in. Company management determines the strategy under the influence of various parties and factors, ranging from media to unions, suppliers, and shareholders (Ward & Peppard, 2002).

After having discussed classical innovation diffusion theory, and the major extensions to that theory, the next section will continue with innovation diffusion within the IT domain.

IT innovations

IT has had a growing share in overall business investments over the last decades. It is not surprising then, that the diffusion of information technology has gotten considerable attention by research and has provided a large part of the total amount of research in innovation diffusion research (Fichman, 2000).

As has been mentioned before, there is no general innovation diffusion theory and most research and theories are adapted to a specific context. In the context of IT innovations, the category to which software, and consequently OSS, belong, there are also certain factors which turn out differently, or prove to be of relevance, compared to any other context.

Fichman (2000) discusses four aspects because of which IT innovations might diffuse differently: two-part adoption decisions, knowledge barriers and organizational learning, network externalities, and incomplete products due to infrastructure dependence. These four aspects will now shortly be considered.

Diffusion of IT innovations

Whether or not to adopt an IT innovation is often a (top) management's decision. However, whether or not actually, and the extent to which locally, use the innovation, is a second decision which does not have to have the same outcome. This could be due to the level of centralization of the IT organization. More centralized control would probably result in the adoption decision being set from above (Ward & Peppard, 2002).

High individual resistance might also prohibit extensive infusion (Fichman, 2000). The five characteristics of an innovation as stated by Rogers (1995) might somewhat explain

why some individuals are more resistant to adopt an IT innovation than others, but it is a rather limited justification²³ (Pijpers, Montfort & Heemstra, 2002).

Knowledge barriers and network externalities as discussed in the previous section are highly relevant in the study of IT diffusion. Often, IT poses such knowledge barriers, for example learning a new software package or programming language. On the other hand, the importance of standards and network size in the case of specific technologies is considerable, as can be seen in the market dominance of Microsoft for various software categories.

IT innovations often consist of incomplete or specialized solutions, or depend on the provision of a certain infrastructure, for example an internet connection or a certain operating system. This means that they are not very attractive to the mass market and often have to be reconsidered. Such incomplete products obviously offer higher uncertainty and risk to the potential adopter (Fichman, 2000). Whether products with higher risk could fit into the adopter's strategy depends on the adopter context. Internally, the application portfolio can explain this, a concept that has been clarified in chapter one.

The applications portfolio relates the degree of business dependence on IS/IT to the degree of contribution of the IS/IT application. The extent to which a business is dependent on IS/IT is also referred to as the level of infusion of the application.

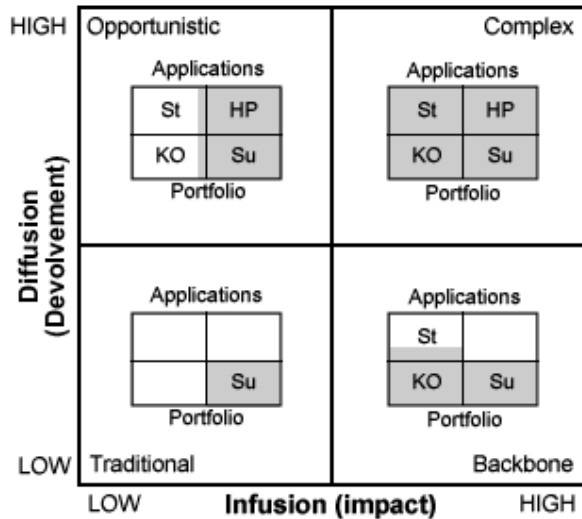


Figure 7. Application portfolios in different contexts

The potential contribution of IS/IT to future business goals, and the extent to which it is scattered throughout the organization is mentioned as the level of diffusion²⁴ (Ward & Peppard, 2002).

To which quadrant a certain application belongs in a specific organization's context determines how it should be managed and how the portfolio should be planned. Research has mixed arguments for the kind of planning and management to use, however there is agreement on the less complex quadrants: support and key operational applications (Ward & Peppard, 2002). Logically, the extent to which IS/IT is diffused and infused in an

²³ This is related to the previously mentioned technology acceptance model by Davis (1989).

²⁴ Diffusion here is referred to more as the spread *within* the organization, as opposed to *among* organizations.

organization, i.e. the IS/IT environment, will impact the strategy for managing and planning the applications portfolio.

Externally, the adoption and diffusion of IT, i.e. a specific application, is determined by the environmental factors as discussed in the previous section. Examples would be organizational slack and the business environment. As compared to strategy, IT can be an enabling factor and a defining factor. A small firm heavily relying on e-commerce obviously depends on technology for its strategy. A large firm using IT to make a global structure work is using it as an enabler (Ward & Peppard, 2002). How IT innovations diffuse in various organizations has gotten considerable research attention, resulting in several frameworks.

Frameworks for organizational diffusion of IT

Rogers (1995) summarizes previous studies on organizational diffusion. However some more comprehensive organizational diffusion frameworks of IT have been developed (Fichman, 1992).

Zmud (1982) argues that the differing results in research studies towards the influence of organizational structure (represented by the level of centralization and formalization) on organizational innovation can be explained by three issues. First, the phase in which the innovation is within the organization. Second, the compatibility of the innovation with the interests of the organizational members. Third, the organizational unit dealing with the innovation, characterized by being technical or administrative. The study found positive results for this reasoning, which confirms that innovation-diffusion research is highly contextual, in this case with regard to structural aspects of organizations.

Kwon & Zmud (1987) devote their attention towards the implementation of IT in organizations. They define five contextual factors which each may impact each of six stages of IT implementation in a process model of stages, namely: initiation, adoption, adaptation, acceptance, routinization, and infusion. The five contextual factors relate to the characteristics of the user community, the technology, the organization, the task to which the innovation is being applied, and the environment. Furthermore, interaction among these five factors was found to be important, for example the fit between the technology, task, and organization; referred to as compatibility (Cooper & Zmud, 1990).

In a study by Cooper & Zmud (1990), the adoption and infusion stages are examined in terms of (IT) innovation implementation success and the related task and technology characteristics. They argue that technological complexity is negatively, and task-technology compatibility positively, related to implementation success. Furthermore, they hypothesize

that these rational arguments are more likely to hold in the adoption stage than in the infusion stage due to the resistance to change and political motives. The research (based on MRP implementations²⁵) supported their reasoning. Rational models of implementation behavior are most relevant for early adoption stages whereas in later stages, political and learning models are more appropriate (Cooper & Zmud, 1990). A study by Rajagopal (2002) on ERP implementations confirms the use of rational models²⁶ in early adoption decisions.

Zmud & Apple (1992) further research the last two stages proposed by Kwon & Zmud (1987). These stages, routinization and infusion, deal with the incorporation of an adopted technological innovation “*within the organization’s operational and managerial work systems*” (Zmud & Apple, 1992). Routinization is defined as “*the permanent adjustments of the governance structure*” to account for the new innovation, whereas infusion is defined as “*the adjustments in work systems and social systems to fully appreciate the innovation’s impact*”²⁷ (Zmud & Apple, 1992). The results of the research indicated that routinization does not fully take account for the incorporation process and occurs at a more rapid rate than infusion. This seems logical since top-down mandates for incorporation are easier set about than actual every-day implementation that infusion refers to. Zmud & Apple (1992) note that a different set of foci might be needed to support the infusion process, as compared to the routinization process. These foci should be more arranged towards the individual acceptance of an innovation²⁸. The following section on IT innovations discusses various adoption contexts.

IT adoption contexts

Fichman (1992) developed a matrix that maps four IT adoption contexts according to the locus of adoption (individual versus organizational) and the characteristics of technology (significant network externalities and knowledge barriers). This matrix can be seen in figure 8. Cell one more or less matches the classical diffusion innovation theory, whereas the three other quadrants represent various levels of complexity. Here, classical diffusion theory alone is not a sufficient explanatory value and may be even of minor importance (Fichman, 1992).

²⁵ MRP stands for Material Requirements Planning, an IT application used in manufacturing firms since the 1980s to support production planning and control.

²⁶ For example cost benefit analyses and vendor screening.

²⁷ This idea might be easier to grasp if stated that infusion is the level of bottom-up acceptance of the innovation, while routinization is top-down recognition of the adoption decision.

²⁸ This relates to the Technology Acceptance Model (TAM) of Davis (1989), which relates the actual usage of an (IT) innovation to the intention and attitude of the user. This would mean that to reach higher infusion levels, one should influence the perceived usefulness and ease of use of an (IT) innovation.

Fichman argues for the inclusion of additional variables as control or independent variables, as well as more in-depth studies of fewer organizations.

Adoption research focuses on one of these quadrants. But, individual adoption of a type two technology can, by means of aggregate effects, lead to organizational adoption. For example, various researchers (Zmud, 1982; Swanson, 1994) make a distinction between the administrative core and technical core, which each has its own goals and vested interests. It is natural that technical innovations emerge in the technical core, and reaches the other core indirectly, and vice versa (Fichman, 1992). This is also a common sequence in the case of OSS, where individuals at the IT/IS department are often the first to come up with OSS for problems.

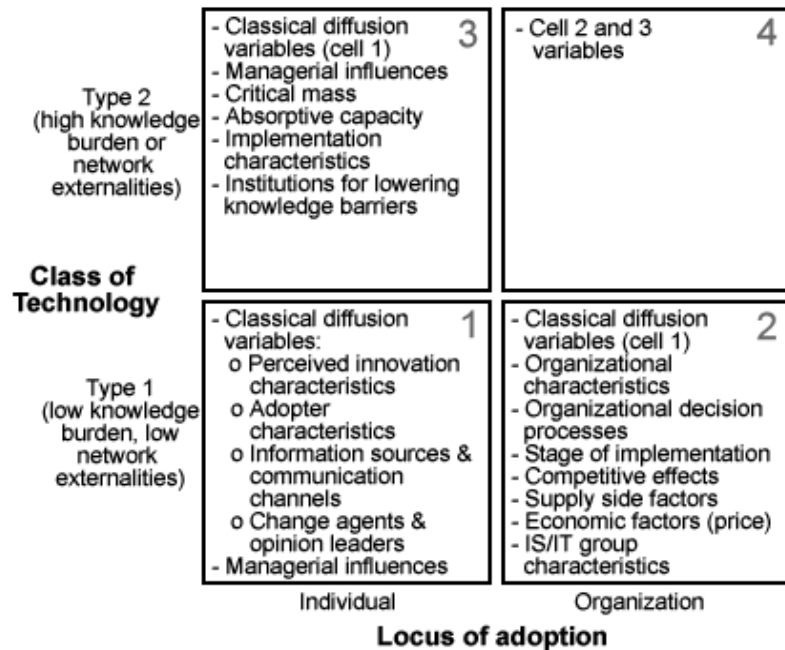


Figure 8. IT innovation adoption context.

When considering the matrix in figure 8 a logical question to arise is to what extent OSS can be considered a type one or type two technology, and which of the variables best explain OSS adoption. The following section addresses these issues in regarding OS as an IT innovation.

OS as an IT innovation

OS and OSS have been discussed in chapter one. When considering OS as an IT innovation, there are two viewpoints:

1. OS as a software development methodology, partly represented by the use of non-restrictive licenses and publicly available source code, as a *process innovation*²⁹.
2. The actual products of the open source development process, that is OSS, as a *product innovation*.

Despite the fact that an OSS product does not have to be an innovation by definition, it does possess some unique characteristics. As concluded in chapter one, this thesis considers the second viewpoint, i.e. that OSS products contain some unique characteristics. When

²⁹ The difference between product and process innovation is , among others, discussed in Zmud (1982)

considering the matrix by Fichman (1992) discussed in the previous section, where does OSS fit in?

As considering the type of technology OSS represents, one can only generalize to a certain extent. Of course, the types of software offered by OSS solutions are also 'everyday' appliances, such as word processing software, which should not pose a significant knowledge barrier to the user. However a lot of OSS is originally developed for the Linux operating system (West & Dedrick, 2003). The Linux bias in OSS does pose a knowledge barrier to the largest part of the potential adopter population, i.e. the end-users. For technical users, especially of large corporate IS/IT departments, this is expected to hold to a lesser extent since they are expected to have specialized workers that know how to operate Unix-like systems. However, not all firms have an IT department or the specific skills needed.

Secondly, network externalities are likely to hold for OSS, as well as on the demand side as on the supply side. OS communities need strong initial leadership and a critical mass of code to get started, and users need to be convinced of a product's advantages. For the demand side, reaching critical mass is likely to be easier due to the zero price (higher triability) and the fact that software is a digital good. As mentioned, the spread of OSS can be expected to be somewhat dependent on the spread of the Linux operating system. However, most OSS that has become popular has provided support for other operating systems via its own user community. Thus the limited and technical-oriented OSS supply and the Linux bias in combination with a Microsoft Windows dominance, argues for OSS as a type two technology. In contrast, OSS supply and usage is rapidly growing which results in more type one technology applications.

The locus of adoption can be individual or organizational. Here it is up to the research to determine which context to use. In this thesis, OSS at the organizational level will be studied. However, it should be kept in mind that individual adoption in an organizational setting can lead to organizational adoption.

Conclusion

This chapter discussed the DOI literature and the limitations of the classical perspectives. These perspectives are not likely to be able to explain more complex technology adoptions. They need to be extended with various factors like network externalities, knowledge barriers, and other context specific variables in order to be able to explain for such developments.

Besides the various extensions to the DOI literature, the context of IT innovations has been considered. It has been reasoned why different issues are likely to be more or less influential in the case of IT innovations. After having already reduced the level of analysis from general technology innovation diffusion to the diffusion of IT innovations, further downsizing has occurred as considering OSS as an IT diffusion.

In relation to this thesis' research the scope has been defined in more detail. When considering the type of research, this thesis will be an adopter study as stated by Fichman (2000). The specific adoption context will include several type one and type two technology OSS applications in an organizational setting, according to the Fichman (1992) matrix. This results in a measure of innovativeness limited to a binary adoption / non-adoption outcome of OSS. The research design will further be outlined in chapter 4.

The next chapter will outline the various factors influencing OSS adoption, and determine which factors are unique and characterizing for OSS adoption in comparison to 'traditionally developed' proprietary software. Secondly, hypotheses are proposed on basis of these characteristics and their influence on OSS adoption.

Chapter 3 – OSS adoption: Framework, determinants, and hypothesized relationships

From the first chapter it has become clear that open source development is a very viable software development process in the current network economy and widespread use of the internet. Following, chapter two showed that the adoption, diffusion, and infusion of new technology innovations is highly context specific and dependent on many different factors. In this chapter determinants of the adoption of OSS are discussed, thereby integrating the first two chapters.

First, the OS adoption model as proposed by Kwan & West (2003) will be highlighted and used to develop a context for this thesis' research. Secondly, two classification schemes that can be used to categorize the different contextual factors that might influence OSS adoption will be outlined. One of the schemes is selected and the various factors in the OSS context that have been found in various studies are provided along with the hypothesized effects they might have on OSS adoption. After this discussion, the various hypotheses will be summarized. In the end a short conclusion will be given.

Open source adoption model

OSS has distinct features, for example in terms of costs and licensing, which influence the OSS adoption decision. Kwan & West (2003) have developed a conceptual model, which shows the evaluation process of enterprises towards OSS adoption. This model provides a

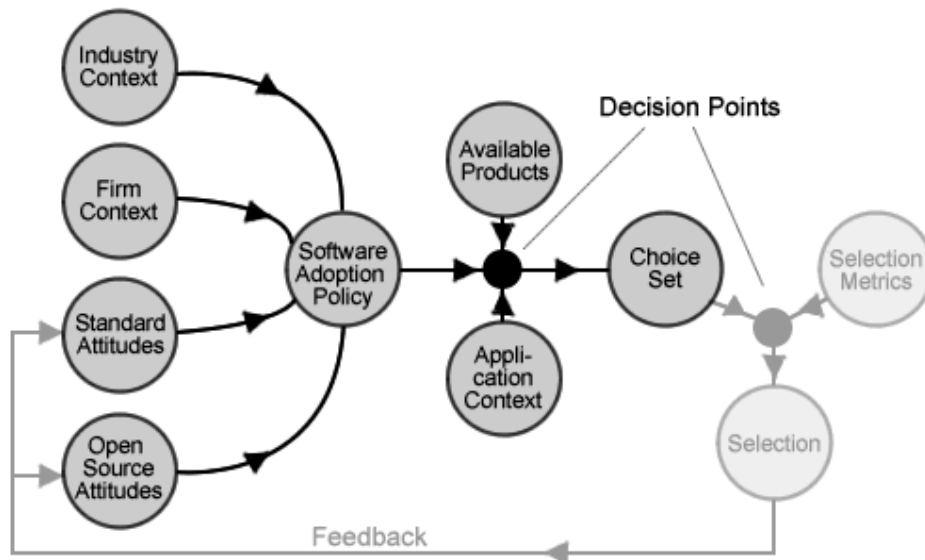


Figure 9. Model for open source adoption

setting for this thesis' research. The conceptual model is shown in figure 9, where the relevant part (left hand side) for this thesis has been put in contrast to the rest of the model.

This model distinguishes between the various aspects that influence firms in the OSS adoption decision: industry context, firm context, standard attitudes, and open source attitudes. Together, these aspects determine the policy the firm employs to guide procurement decisions. Depending on the market supply of OSS products and the match of these products with the firm's application context, in terms of risk, cost and features³⁰, the result is a certain choice set. This choice set holds the potential 'nominees' for adoption. This thesis will focus on the factors that specifically influence OSS adoption and evaluation, ultimately leading to that choice set. The next section will discuss the classification scheme, which will be used to categorize the various factors.

Classification scheme for OSS adoption determinants

Two potential classification schemes for influencing factors on OSS adoption decisions have been identified: The TEO scheme discussed by Tornatzky & Fleischer (1990) and the classification scheme presented by Fichman (2000).

TEO scheme

TEO stands for technology, environment and organization. This classification scheme maps the various factors under these three categories and interlinks them, while putting the innovation decision making decision in the middle. These three elements are conceived to interact with each other and to influence the adoption decision (West & Dedrick, 2003).

When compared to traditional diffusion innovation theory, this model includes various 'additional' factors that were discussed in the previous chapter, i.e. organizational and environmental factors. However, it does not explicitly include factors such as knowledge barriers or network externalities. Since the categorization is so broad, this does not pose a problem. In a research study on the factors affecting open systems adoption the TEO model has been successfully used and adapted to the open systems adoption context to include such variables (Chau & Tam, 1997). Another classification scheme is provided by Fichman (2000) and will be discussed next.

³⁰ This relates to the applications portfolio discussed in chapter one and two. The position an application takes in a firm's application portfolio determines the importance of the application's risk, features, and cost.

IT diffusion and assimilation scheme

In the previous chapter, the three general research questions in diffusion innovation research as presented by Fichman (2000) were discussed. Fichman also provides a classification scheme for factors that influence diffusion of (IT) innovation related to these three questions. One of the conclusions of the previous chapter was that this thesis is an adopter study, which generally focuses on the research questions 2 and 3 as identified by Fichman. In terms of the classification scheme, Fichman maps this to the categories ‘*technology-organization combination*’ and ‘*organizations & adoption environments*’. Therefore, these categories will be the categories in which various determinants have been identified. An overview is given in figure 10. The other category, ‘*technologies and diffusion environments*’ has not been included but is shown for the sake of completeness. Additionally, network externality effects have been added to the ‘*organization & adoption environment*’ category.

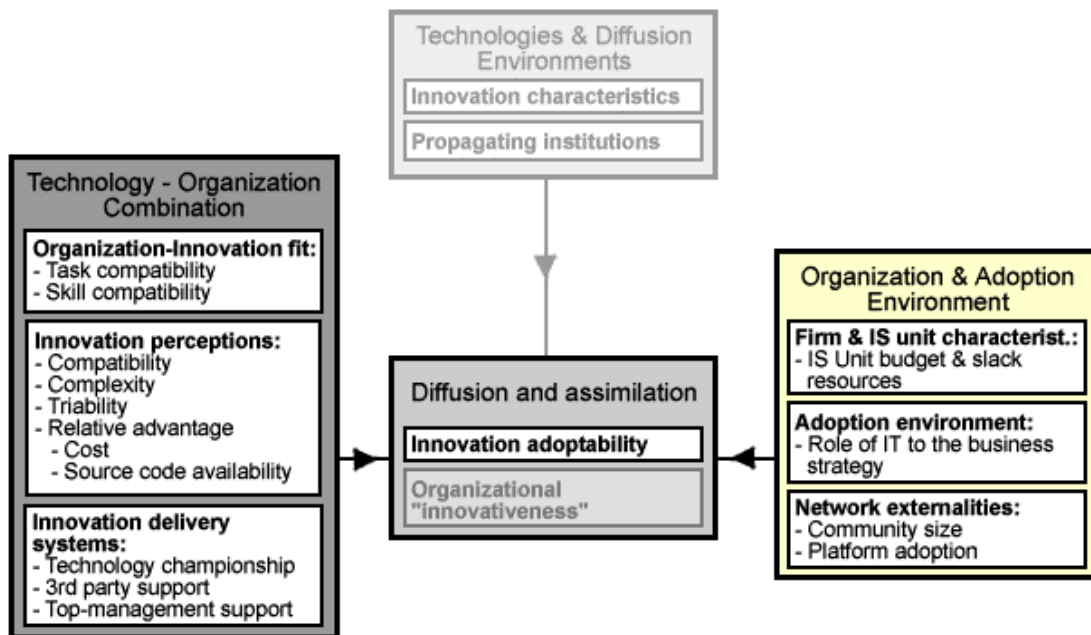


Figure 10. Classification scheme of determinants of OSS adoption.

This classification scheme is less straightforward than the TEO scheme, but they do have overlap. This scheme more or less incorporates the TEO scheme and is specific towards the adoption of IT innovations, which includes OSS. It comprises all factors discussed in the previous chapter that extend the classical diffusion innovation theory. Because of this completeness, this scheme is adopted to guide the classification of OSS adoption factors. These various factors are already included in figure 10 and will be discussed in the next section.

Factors influencing OSS adoption

This section argues for various factors that are expected to influence OSS adoption and diffusion. This is done according to the categorization provided by Fichman (2000).

OSS - organization combination

Whether OSS will fit with a specific organization will not only depend on specific characteristics of the open source product or the organization, but also on the specific combination of these factors. This is best described by terms as compatibility (Fichman, 2000). Two types of compatibility are discussed in this section: task compatibility and skill compatibility.

Organization - OSS Fit

Task compatibility

The match between a certain task and a solution is referred to as task compatibility. As discussed in the first chapter, the supply of OSS solutions is rather biased towards certain categories that comply most with the motivations of the developers. Though, hybrid solutions have come up of firms supporting OSS in various ways, fitting them to the demands of commercial organizations by providing e.g. support and consulting. Nevertheless, these hybrid models do not take away the fact that most OSS is designed for a rather limited problem domain, often rather technical in nature or at least technically packaged. Thereby these firms supporting OSS do not enlarge the supply of OSS products, but they only make the available supply qualitatively better, i.e. more attractive to adopt.

An area where OSS has traditionally been well positioned is in the provision of internet related services. From the task compatibility viewpoint this makes sense, because for most of these internet services³¹ there is no direct impact on the end-users in terms of platform switching costs (West & Dedrick, 2003). This leads to the first hypothesis:

H₁: Perceived task compatibility is positively associated with OSS adoption

Skill compatibility

One could argue that the OSS available is *open* source, which would make it easier to adopt and adapt such a solution to the company's demands. Nevertheless this is susceptible to a couple of conditions:

³¹ Examples of such internet services are web server software, content management systems, and e-mail servers.

1. Depending on the type of application (i.e. strategic or support) and the OSS license (i.e. restrictive versus non-restrictive) such adaptations might not be an option since the application might have to be ported back to the open source community as defined in the license. Obviously, for a strategic application this porting back might not be a realistic option.
2. Adaptation of the software to the firm's wishes can easily get very costly. For support applications these costs might not outweigh the benefits of closed-source, off-the-shelf, solutions.
3. The firm might not have the required knowledge in-house to do the adaptations by itself, i.e. the firm might not have the absorptive capacity to implement the OSS (which might also be the case for a standard implementation of the product).

The first two issues are more related to the OSS license and the level of task compatibility. The third factor relates to the skill compatibility, i.e. does the organization possess the required skills? Such skill compatibility is expected to positively influence OSS adoption, resulting in the second hypothesis:

H₂: Perceived skill compatibility is positively associated with OSS adoption

Another factor in terms of skill compatibility concerns the platform a firm employs (Unix versus Windows). If a firm is locked-in to Windows, it is harder to switch to Linux in terms of skills and complementary products. This aspect is discussed in the section on network externalities. The next section will discuss innovation perceptions in the case of OSS.

OSS Perceptions

When referring back to the categorization scheme, innovation characteristics are part of the technology-organization combination. Perceptions on innovation characteristics and attributes are key to the adoption decision (Fichman, 2000). As discussed in chapter two, classic diffusion of innovation literature has provided for several of these characteristics.

Classic Innovation Characteristics

Classic innovation characteristics are the five innovation characteristics³² as outlined by Rogers (1995). Tornatzky & Klein (1982) did a meta-analysis on various research in this field where they only found support for three out of these five factors. However, in that same study Tornatzky & Klein (1982) found different shortcomings in the various studies that were analyzed. Four characteristics of OSS will be highlighted in this section:

³² From Chapter one: Relative advantage, compatibility, complexity, triability and observability.

1. Compatibility
2. Complexity
3. Triability
4. Relative advantage

Relative advantage is taken as a proxy for several variables often cited in OSS studies: cost and source code availability.

Compatibility

Compatibility of an innovation is referred to as “*the degree to which an innovation is perceived as being consistent with the existing values, past experiences, and needs of the receivers*” (Tornatzky & Klein, 1982). The previous section on organization-OSS fit already dealt with the compatibility issue in terms of skills and tasks.

The extent to which OSS is compatible at a technological level depends. It will be subject to, among other factors, on the current technological infrastructure and applications portfolio of the firm. This is especially true for the availability of specific key applications on which day-to-day business depends. These applications are often built towards a specific operating system and API³³ and cannot be ported to another operating system (Linux), for which OSS is often developed (West & Dedrick, 2003). In addition, certain 3rd party applications that are used might not be compatible. Third, the current IT supplier or party that IT aspects are outsourced to in terms of operational or consultancy tasks, does not always support OSS.

H₃: Perceived compatibility with key applications and technological infrastructure is positively associated with OSS adoption

Complexity

Complexity is “*the degree to which an innovation is perceived as relatively difficult to understand and use*” (Tornatzky & Klein, 1982). In the case of software, this definition of complexity can be mapped to various terms depending on the user type: transparency due to the availability of the source code, quality of the source code, quality and availability of documentation, ease of maintenance, functionality, and compliance to standards³⁴. Overall, the complexity of a technological innovation is negatively related to its adoption (Tornatzky & Klein, 1982), which leads to the following hypothesis:

H₄: Perceived complexity is negatively associated with OSS adoption

³³ API stands for Application Programming Interface, which is the interface between a system and the software it interacts with.

³⁴ Since the definition of complexity is very related to the concept of usability, which has been discussed in chapter 1, and the fact that in practical terms, constructs for usability are found to be rather similar to the ones for complexity, usability has not been put in a separate hypothesis.

Since the source code of OSS is per definition freely available, it is less complex and more transparent to the user in the sense that the software is not per se functioning as a black box. However, the quality of the source code could be an issue. Proprietary software development often adheres to strict principles to get to the required product features in a minimum time span and within budget. On the other hand, developers of OSS have more time to review the code and come up with alternative solutions. In the case of code documentation, OSS is at a disadvantage. Since this is one of the less-attractive jobs in open source development, documentation is often lagging behind with actual releases, is incomplete or even not available at all. Documentation is one of the fields where commercial companies often fall in. These firms often have extensive knowledge with the OSS product to provide documentation, or they can provide complementary products, implementation support, or training. Fourth, Linux based systems have the reputation to be very low on maintenance time (West & Dedrick, 2003).

Triability

Triability is the extent to which an innovation can be “*experimented with on a limited basis*” (Tornatzky & Fleischer, 1982). The ‘limited basis’ might traditionally refer to the opportunity cost and monetary cost spent on the experimentation. For OSS products, the direct monetary cost is often zero, since there are no license fees of any kind. Often, commercial products offer a trial version of their products, but these are often only functional for a limited period of time or with limited functionality. However, West & Dedrick (2003) found no impact on triability between software that was ‘free’ or which could be tested for a nominal cost. Additionally, testing OSS products while such technology is unfamiliar to the testing person might involve significant time investments.

What often appears to be the case is that in many organizations programmers (i.e. technical personnel) casually test OSS at home, which reduces the perceived risk of open source adoption and influences the overall organization in adopting OSS (West & Dedrick, 2003). Therefore, triability is reflected in hypothesis five as:

H₅: Perceived triability is positively associated with OSS adoption

Relative advantage

Relative advantage is an innovation characteristic that has been found not so easy to define and study. Since the term is so broad, it is often used as a category in which various factors are dumped. If not, it is mostly referred to as the ‘gain in effectiveness’ by using the innovation. It was found to have a positive relationship with innovation adoption. (Tornatzky

& Klein, 1982). In the case of OSS, relative advantage seems a category in which various factors can be put, e.g. source code availability and cost.

Additional characteristics under relative advantage

Cost

Total Cost of Ownership (TCO) of OSS and proprietary software is an area of continued debate. Results vary, mainly due to different measurement scales and (non-) inclusion of certain variable and indirect (qualitative) costs (Knubben, 2004). Chapter one outlined the various factors that can be part of such a cost analysis. What is generally accepted though, is that replacing a 'Windows environment' with a 'Linux environment' brings one-time migration costs, which can provide a significant barrier to adoption (West & Dedrick, 2003). These migration costs include the evaluation costs, implementation labor costs, consultancy costs, human switching costs, and retraining costs.

H₆: Perceived switching costs are negatively associated with OSS adoption

On the other hand, one could argue that such migration costs are one-time sunk costs and they should not turn down the adoption decision, and in fact the investments can have a very short payback period (Fitzgerald & Kenny, 2003). OSS on the Linux platform offers some definite and short term cost advantages in terms of license fees, hardware demands (in terms of requirements and compatibility), and downtime (Knubben, 2004). In practice, cost savings associated with OS often prove important to the OSS adoption decision (Fitzgerald & Kenny, 2003; Ghosh & Glott, 2003; The Dravis Group, 2003). The other more celebrated advantages (e.g. reliability) seem to be of far lesser importance. Therefore, it is argued that:

H₇: Perceived hardware cost savings are positively associated with OSS adoption

H₈: Perceived software cost savings are positively associated with OSS adoption

Source code availability

A second feature of OSS is the fact that its source code is freely available³⁵. This effects OSS adoption at two levels, direct and indirect. The direct effects relate to the opportunity it provides for customization. Yet study shows that not all, even very few, firms are primarily interested in the availability of the source code because adapting the software does not belong to their skill set, or they are just looking for off the shelf software solutions (Fugetta, 2003; West & Dedrick, 2003). In addition, just having the source code is often not enough to adapt the software, which also requires some documentation for guidance (Fugetta, 2003). On the other hand, source code does provide for more flexibility and control regarding supplier

³⁵ For a discussion on what source code is, and what is meant by free availability, see chapter one.

independence (The Dravis Group, 2003; Fitzgerald & Kenny, 2003; Ghosh & Glott, 2003). Therefore, it is argued that:

H₉: Perceived supplier independence is positively associated with OSS adoption

Indirect effects relate to the advantages it brings in the development process. Perceived indirect advantages include increases in safety, reliability, stability, continuity and interoperability (Coppola & Neeley, 2004; The Dravis Group, 2003; Berlacon, 2002a). Interoperability however, does not only come from source code availability, but mainly by compliance to open standards (Fugetta, 2003).

Because of the fact that source code availability does not provide direct disadvantages to the firm, and indirect are expected to be merely positive, the following hypotheses are proposed:

H₁₀: Perceived safety is positively associated with OSS adoption

H₁₁: Perceived reliability is positively associated with OSS adoption

H₁₂: Perceived stability is positively associated with OSS adoption

H₁₃: Perceived continuity is positively associated with OSS adoption

H₁₄: Perceived open standards compliance is positively associated with OSS adoption

OSS delivery systems

Innovation delivery systems provide the means by which the implementation process is managed (Fichman, 2000). This does not only include the final stages³⁶ as defined by Kwon & Zmud (1987), but also the first stages. After all, this study focuses on the adoption context up to the final decision point. In this section, four aspects of the OSS delivery system will be considered: technology championship, third party support, and top management support.

Technology championship

Technology championship is an important factor in innovation adoption. This is also expected to be the case for OSS adoption. OSS is often a rather peculiar concept from a manager's point of view. Consequently OSS is perceived more risky and less legitimate. Therefore, the social influence of the technical core of the firm as a technological gatekeeper (Orlikowski, 1995) is expected to be considerable.

Individual or local adoption of OSS by information technology specialists often takes place because these persons are not scared or busy with the business aspect of OSS but work with OSS in a pure technological fashion (Fitzgerald & Kenny, 2003). This often introduces

³⁶ The six stages referred to are: initiation, adoption, adaptation, acceptance, routinization, and infusion.

OSS to the firm, and such pioneers often act as advocates of OS technology. In addition, adopters of OSS are not only found to disseminate information on OSS, but are also turning into advocates of the OS movement which try to *convince others* to adopt (Bonaccorsi & Rossi, 2003b). This leads to the following hypothesis:

H₁₅: Perceived OSS championship is positively associated with OSS adoption

Third party support

A second implementation factor is third party support. The previous chapter discussed the importance of third parties to lower knowledge barriers and their role in innovation packaging. In the case of OSS adoption, such firms³⁷ act as change agents promoting the switch towards open source, not only in terms of recognizing the technical maturity and proficiency of the products and thereby providing a sense of legitimacy, but also in terms of making their own products compatible and thereby providing complementary products. In addition, these firms provide consulting and training services and the like, which promotes OS adoption. The third party support by firms like IBM, HP, and SUN has been found to be especially important for larger organizations who are used to technology and support contracts from large IT vendors (West & Dedrick, 2003). This third party support is therefore expected to positively influence OSS adoption:

H₁₆: Perceived third party support is positively associated with OSS adoption

Top management support

To fully adopt OSS, top management support is needed. Their support provides the implementation with enough backing in terms of authority and resources. Yet, top management is often not easily convinced of OSS because such software lacks the proper legal backing. This is expected to be certainly true for business critical applications and applications which have high visibility, i.e. are on the desktop (Fitzgerald & Kenny, 2003).

Top managers often do want, at least some, support, whether that is actually really needed or not, to justify their actions. On the other hand OSS can offer cost advantages and lower supplier dependence. Implementing a risk mitigation process that addresses legal, financial, and technical³⁸ pitfalls when considering OSS adoption (Casanova, 2003) seems critical to evaluate OSS and convince management. Once top management is convinced it is expected to be positively correlated to open source adoption:

H₁₇: Perceived top management support is positively associated with OSS adoption

³⁷ Examples of firms supporting the open source community are HP, IBM, and SUN and their adoption of Linux

³⁸ Technical aspects include support, security, and development issues

Organizations & adoption environment

Some organizations are found to be more innovative than others. The source for this innovativeness is often found in the organizational and environmental characteristics (Fichman, 2000). This section considers three aspects of the organization and adoption environment: characteristics of the firm and the information systems department, the adoption environment, and related to the adoption environment, network externalities.

Firm & IS unit characteristics

When considering OSS adoption, the organizational characteristic that will be discussed in this section is the size of the information systems department in terms of slack resources. Other factors that *could* be included are the organizational deployment of internet services and the technical expertise of the workforce. However, these aspects are likely to correlate with skill- and task-compatibility, aspects that are already included in hypotheses one and two.

Slack resources and budget of the IS unit

Organizational size, as discussed in chapter two, is often taken as a stand-in for other positively related variables like slack resources, IS department size and scale. Slack resources can work in two directions. First of all, slack human resources in combination with limited financial resources and a sufficient set of skills provide solid ground for OSS adoption. On the other hand, too much slack is likely to value convenience and relaxed investment rational resulting in loose management choosing the 'easy way out'. This often results in proprietary software (Ghosh & Glott, 2003; West & Dedrick, 2003). This leads to the following hypothesized relationships:

H₁₈: Abundance of slack resources is negatively associated with OSS adoption

H₁₉: Limited financial slack resources in combination with enough slack in human in resources is positively associated with OSS adoption.

Adoption environment

The adoption environment considers factors external to the firm. One of these is the level of competitiveness and market concentration. It is hard to generalize these market factors to overall OSS adoption. One could think of many hypothesized relationships that often show difficult to be formally linked to adoption (Chau & Tam, 2000). Two determinants to OSS adoption coming from the adoption environment are discussed here, the role of IT to the business strategy and supply side economics.

IT and the business strategy

The extent to which market characteristics influence OSS adoption also depends on the firm's business strategy. The business strategy represents the way in which a firm has decided to operate in, and react to, its environment. The centrality and importance of IT to the business strategy has been found to correlate to the willingness to adopt. In such situations a small decline in costs per unit leads to much larger overall savings and strategic advantages gained by IT innovations are more important than in organizations where IT plays a support role (West & Dedrick, 2003). Thus, the importance of IT to the business strategy is expected to correlate positively to OSS adoption:

H₂₀: Perceived importance of IT to the business strategy is positively associated with OSS adoption

Although not included by Fichman as an appropriate factor, the adoption environment also encompasses the number of other adopters and levels of diffusion reached for OSS. The factors are generally referred to as network externalities.

Network externalities

From chapter two it followed that in some case of technology adoption, network externalities and compatibility with dominant interfaces are more important to the adoption decision than actual technology features and product characteristics. Bonaccorsi & Rossi (2003b) argue that the adoption and diffusion of OS are influenced by:

- The perceived intrinsic value
- The negative network externality effect coming from the dominant standard
- The positive network externality effect coming from the community access
- The competitive effects by commercial market players in the software industry

They then simulate the adoption decision for a population of heterogeneous interacting agents using a model. The simulation results show that OS diffusion "*indeed depends on the initial intrinsic values assigned to the technology by adopting agents*". As a general result they conclude that "*under many conditions commercial software and OSS are likely to **coexist** even in the limit*" (Bonaccorsi & Rossi, 2003b, p. 1256). The perceived intrinsic value is primarily determined by the perceived innovation characteristics that have been previously discussed. In this section two other aspects are discussed: community size and platform adoption.

Community size

As should be clear by now, the supply side environment of OSS is rather different as compared to proprietary software. It relies on communities of, often volunteer, contributors, which often do not undertake any marketing whatsoever, and that rely heavily on open standards and other OSS. In addition, an often-cited advantage of OSS is that it reduces

supplier dependence, i.e. source code availability, community support, and reliance on open standards minimize any vendor lock-in potential, which often take place for proprietary software (Overby, Bharadwaj & Bharadwaj, 2004). All in all, the size of such a community is likely to result in various positive effects for adopters (Stewart, Ammeter & Maruping, 2005). It is therefore argued that community size positively relates to OSS adoption:

H₂₁: Perceived community size is positively associated with OSS adoption

Platform adoption

OSS is very dependent on the platform choice: Linux versus other operating systems, *and* the type of hardware used (West, 2002; West & Dedrick, 2003). Many of the OSS applications have been primarily constructed for Linux operating systems. Having Linux is even a pre-condition to most OSS. Therefore, the greater the market acceptance of Linux, the more positive the effects (read: lower switching costs) will be for OSS adoption due to e.g. complementary assets, credibility, lower training costs, easier file sharing, etc. Yet, it has been found of major importance that complementary assets provide good fit instead of the sheer volume of available complements. In addition, ‘Unix shops’ are more prone to switch to Linux than ‘Windows shops’ in terms of compatibility of complementary applications (West & Dedrick, 2003) and lower knowledge barriers (Fitzgerald & Kenny, 2003).

Hardware is another issue. Linux and MS Windows run on commodity hardware (Intel chips). Unix on the other hand, often runs on proprietary hardware. Therefore, MS Windows is more compatible in terms of hardware (West & Dedrick, 2003). See figure 11 for an overview. Lintel (Linux and Intel) therefore are expected to be more attractive to Unix shops since they only need to switch to cheaper Intel platforms. For Windows shops the problem lies in the fit of complementary assets, which is harder to overcome.

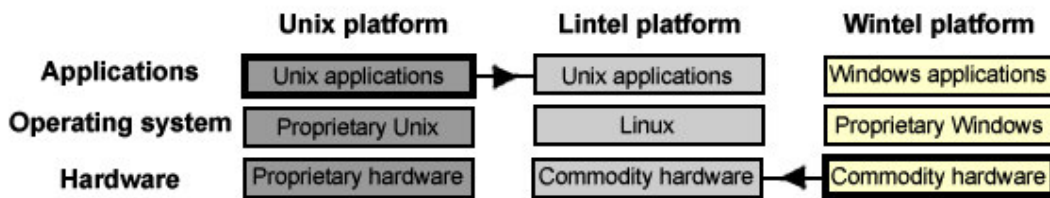


Figure 11: Platform standards.

Also, of major importance seems to be whether a firm’s IT skills are Windows based or Unix based. Windows users do not possess the necessary skill set and are thereby often too locked-in to switch to Linux-based OSS. “*The transition to Linux is incremental for Unix shops where skills are easily transferable but discontinuous for Microsoft shops that lack*

such skills” (West & Dedrick, 2003). This discussion provides background for the following hypothesis:

H₂₂: A background as a ‘Unix shop’ is positively associated with OSS adoption

The discussion on network externalities has finished the discussion on determinants of OSS adoption. An overview of all hypotheses is given below.

Overview of hypotheses

Table 3 – Overview of hypotheses	
Determinant	Hypothesis
Task compatibility	H ₁ : Perceived task compatibility is positively associated with OSS adoption
Skill compatibility	H ₂ : Perceived skill compatibility is positively associated with OSS adoption
Compatibility	H ₃ : Perceived compatibility with key applications and technological infrastructure is positively associated with OSS adoption
Complexity	H ₄ : Perceived complexity is negatively associated with OSS adoption
Triability	H ₅ : Perceived triability is positively associated with OSS adoption
Costs	H ₆ : Perceived switching costs are negatively associated with OSS adoption
	H ₇ : Perceived hardware cost savings are positively associated with OSS adoption
	H ₈ : Perceived software cost savings are positively associated with OSS adoption
Source code availability	H ₉ : Perceived supplier independence is positively associated with OSS adoption
	H ₁₀ : Perceived safety is positively associated with OSS adoption
	H ₁₁ : Perceived reliability is positively associated with OSS adoption
	H ₁₂ : Perceived stability is positively associated with OSS adoption
	H ₁₃ : Perceived continuity is positively associated with OSS adoption
H ₁₄ : Perceived open standards compliance is positively associated with OSS adoption	
Technology championship	H ₁₅ : Perceived OSS championship is positively associated with OSS adoption
Third party support	H ₁₆ : Perceived third party support is positively associated with OSS adoption
Top management support	H ₁₇ : Perceived top management support is positively associated with OSS adoption
IS slack resources and budget	H ₁₈ : Abundance of slack resources is negatively associated with OSS adoption
	H ₁₉ : Limited financial slack resources in combination with enough slack in human resources is positively associated with OSS adoption.
IT in business strategy	H ₂₀ : Perceived importance of IT to the business strategy is positively associated with OSS adoption
Community size	H ₂₁ : Perceived community size is positively associated with OSS adoption
Platform standards	H ₂₂ : A background as a ‘Unix shop’ is positively associated with OSS adoption

Conclusion

This chapter has discussed the various factors that are likely to influence the adoption of OSS. These factors have been categorized according to the classification scheme presented by Fichman (2000) and can be placed in context by means of the conceptual model for OS adoption as put forward by Kwan & West (2003). Finally, this has led to hypotheses which all indicate a relationship between an independent variable and OSS adoption.

Overall, taking in consideration the various sources, it can be said that the availability of the source code does not seem the most important advantage OSS offers to adopters. It is more the indirect effects of source code availability like supplier independence that seem advantageous. In addition, the note of free as in ‘free beer’ does appear to be appropriate as of free in ‘free speech’ in many cases. In short, at the bottom line in many businesses, the fact that OSS costs nothing to use is more important than the ideological reasoning behind the free availability of source code, as posed by OSS proponents. However, one is not likely to exist without the other.

One note should be made on the hypothesized importance of OSS licenses on OSS adoption. This relationship could be placed under relative advantage. However, seen the type of research this requires, this hypothesis does not seem to fit in this thesis. For sake of completeness, the original section and hypothesis on this subject can be found in appendix I. The next chapter will present the research methodology that has been used to acquire the data set and test the hypotheses outlined in this chapter.

Chapter 4 – Research methodology

The previous chapter identified the various hypotheses that need to be tested to answer the problem statement of this thesis. This chapter will present the research methodology that has been used to accomplish this. To conduct the research the survey method has been used, in the form of an on-line questionnaire. Details of this approach can be found in the following section on research design. Thereafter an overview will be provided of the various instruments that were used to operationalize the hypotheses. Third, a short discussion of the data collection procedure. Fourth, the initial sample selection and the actual response set will be shown. Fifth, the validity of the instruments will be discussed. In the end, a short conclusion will be given.

Research design

The research that had to be done for this thesis is classified as *descriptive research* (Churchill, 2001, p. 126), with the objective to determine the proportions of firms in a specified population (for-profit firms) behaving in a certain way (adopting OSS). This is done in order to make specific predictions on which factors influence the adoption of OSS. The research was implemented with a cross-sectional study involving a sample of firms from the population of interest, for-profit firms³⁹. This has resulted in a dataset that provides a snapshot of data of interest at a single point in time and can be used to test the various hypotheses. The data collection method used was the survey method.

In order to gather the data, a questionnaire was constructed. The questionnaire was designed to measure respondent perceptions on each of the hypotheses. In order to maximize the validity of the measurements, instruments from prior research in various areas have been used to construct the questionnaire. The following section will present in detail how each variable was operationalized.

Operationalization of variables

Most of the instruments that have been used to construct the questionnaire were taken or adapted from previous research in technology innovation diffusion, information technology, and marketing. However, for some hypotheses no exact replicate instrument could be found. In these cases, an instrument was constructed or seriously adapted.

³⁹ More on the sample design in one of the following sections.

Most items were measured by one or more statements on a 5-point Likert scale. In order to gain additional insights in the two most important variables that had been identified in chapter four, supplier independence and software cost, the possibility for posting additional comments was provided after the related constructs for these variables. In addition, several other variables (e.g. demographic) asked for a different scale.

The dependent variable, adoption of OSS, was measured by asking respondents to indicate whether they used such software, and if so, if they could specify which of a listed number of OSS products they used. This list consisted of a number of very popular OS products. The respondent could choose one of the following options:

- A. Our organization **does not** use open source software, because.....
- B. Our organization **does not** use open source software and I am not familiar with this concept. Therefore, I cannot answer this question.
- C. Our organization **does** use open source software, among which are: (followed by a list of OSS)

In case of option A, the respondent was taken to an alternative page, on which it was shortly explained that although the respondent's organization did not use OSS, the respondent could still fill out the remaining questions in order to measure non-adopter perceptions. When a respondent would opt for the second answer, option B, the respondent was taken to another page and was shown a text in which the respondent was thanked for the co-operation, and the survey was finished. As a third, when a respondent would indicate that his/her organization did use OSS and possibly indicated to use any of the listed software, the survey would normally continue.

Table 4 provides an overview of the operationalization of all variables. The first and second column indicate the involved hypothesis and variable description. The third column refers to the question / item in the survey (which can be found in appendix G). The fourth column provides the number of items from the source used for that question. The fifth column displays the source from which the instrument has been taken. A full citation of these sources can be found in the references. Items for which no matching construct could be found and which had to be constructed, are indicated with a dash in the source column.

Table 4 – Operationalization of variables				
Hypothesis	Variable	Item	Number of items	Source / Adapted From
Filter and warm-up questions				
Demographics	General questions – determine sector	a	1	e-Cology Corporation (2003)
Demographics	General questions – determine ICT manager	b	1	-
Demographics H ₁₈ , H ₁₉	General questions – determine size SME-LE	c, d	2	e-Cology Corporation (2003) Commission of European Community (2002)
Independent variable	Filter question – adopted OSS	e	1	Berlacon (2002)
Organizational characteristics				
H ₁₈	IS slack resources and budget - abundance	a, b	2	-
H ₁₉	IS slack resources and budget – limited HR	a	1	Ghosh & Glott (2003)
H ₁₉		b	1	-
H ₂₀	IT in business strategy	a	1	Rathnam, Johnsen & Wen (2004)
Ranking of software characteristics				
H ₃ – H ₁₄	Overall innovation characteristics	a	1	Ghosh & Glott (2003)
Perceived open source characteristics				
H ₃	Compatibility	a-d	4	e-Cology Corporation (2003)
H ₄	Complexity	a-d	4	Moore & Benbasat (1991)
H ₅	Triability	a-d	4	Moore & Benbasat (1991)
H ₆	RA – switching costs	a-d	4	Berlacon (2002)
H ₇	RA – hardware cost saving	e	1	Berlacon (2002)
H ₈	RA – software cost savings	f	1	Berlacon (2002)
H ₈	RA – software cost savings	g	1	-
H ₉ – H ₁₄	RA – source code availability (SCA)	a	1	Ghosh & Glott (2003)
H ₉	RA – SCA – supplier independence	b	1	Ghosh & Glott (2003)
H ₉	RA – SCA – supplier independence	c	1	-
H ₁₀	RA – SCA – safety	d	1	e-Cology Corporation (2003)
H ₁₁	RA – SCA – reliability	e	1	Ghosh & Glott (2003)
H ₁₂	RA – SCA – stability	f	1	-
H ₁₃	RA – SCA – continuity	g	1	-
H ₁₄	RA – SCA – open standard compliance	a	1	e-Cology Corporation (2003)
Implementation issues				
H ₁₅	Technology championship	a-d	4	Ray & Patnayakuni (1996)
H ₁₆	Third party support	a	1	Ghosh & Glott (2003)
H ₁₇	Top management support	a-c	3	Premkumar (2003)
Compatibility of work, tasks, and capabilities with OSS				
H ₁	Task compatibility	a-c	3	-
H ₂	Skill compatibility	d	1	Berlacon (2002)
Adoption environment: Network externalities				
H ₂₁	Network externalities – community size	a	1	Frels, Shervani & Srivastava (2003)
H ₂₂	Network externalities – platform standards	b-c	2	-

Data collection procedure

First the questionnaire was composed from the previously mentioned sources in a text editor. The first version was reviewed and adapted internally. Then, an on-line version of the questionnaire was developed by making use of the PHP scripting language and a MySQL database to capture the data. Thereafter the survey seemed sufficient to be tested.

The ‘paper’ as well as the on-line version of the questionnaire were pilot tested by a few participants, which were ICT professionals and academics. Their responses were not included in the final result set. Feedback was received on paper, via e-mail, and in face-to-face meetings. The responses led to some modifications in the questionnaire, for example the order of the question sets, a review of some of the used instruments, and the way some of the questions were stated. In addition, the usability of the questionnaire was improved by reorganizing parts of the layout.

Thereafter, the improved questionnaire was put on-line. The questionnaire was tested for the most popular browsers, and used cookies and IP-notation to prevent double entries. In addition, the questionnaire was split up over several pages, in order to capture as much as information in the database as possible if the respondent decided to stop filling out the questionnaire.

As a last, all the selected firms (which were put in a database) were sequentially e-mailed by using a script. In 10 days, about 1770 e-mails were sent to the selected respondents. These e-mails provided an introductory text and a hyperlink to the on-line questionnaire. In the e-mail the reader was asked to forward the e-mail to the ICT manager in their organization. The sample selection and setup are discussed in the next section.

Sample

From the problem statement and previous chapters the only restriction to the sample selection was the fact that the target organizations had to be for-profit, thus excluding governmental and educational organizations. This decision was made because non-profit organizations might bias the response set. It was reasoned that the lower costs associated with OSS might provide a higher incentive to use OSS in such organizations.

Besides this assumption, certain sectors are logically excluded from a practical point of view, since they are expected to be too low in IT expenditures, and consequently, software usage. This should not have to be a problem, since such software could be open source. However, since the questionnaire was designed for ICT managers having specific knowledge,

firms in such sectors were expected not to be able to respond. Such sectors include mining and agriculture.

The consideration of the level of IT expenditures also applied to the remaining firms. Firms with higher IT expenditures are likely to use more software in absolute terms, and thereby have a higher chance of using OSS. In addition, even though firms with higher IT expenditures do not have to use OSS more, at least the higher importance of IT is likely to increase their overall knowledge on IT related issues, including OSS. So they are expected to at least know *why* they do, or do not, use OSS, and thereby could bias the response rate towards these firms. This consideration was taken in account when determining the sample.

A selection of sectors was made according to the NACE⁴⁰ categorization and IT spendings as a percentage of revenue, as mentioned in Berlacon (2002a). To minimize any bias in response by higher IT spendings, more firms were selected from NACE categories that were stated to show lower IT spendings. The firms were taken from the REACH database, which holds corporate information on most Dutch firms and is accessible to students via the university's website. After applying the NACE categorization, the sample was further downsized and optimized by applying a selection of filters to the sample:

1. Only firms with a size between 5 and 5000 employees
2. Only firms with an e-mail address
3. Only firms with a status flag set to normal (no bankruptcy, etc.)
4. Only firms that were economically active (no holding structures, etc.)

A first selection resulted in about 1550 firms. These were initially e-mailed to fill out the survey. Because of a low response rate, 200 additional firms were selected from a NACE sector with high IT spendings in order to get a sufficiently large dataset. The sample that was ultimately used can be seen in table 5 and can be categorized as a *convenience* sample.

As can be seen in the table, originally (excluding the 200 additional firms) the percentages were progressive when considering the IT intensity of the various categories. After these additional firms were added, and which is currently shown, the percentage firms in the sector with the highest IT expenditures were a bit higher (19,2%) than the part of the sample with firms having medium IT expenditures (17,6%).

Secondly, the table shows the total number of firms that were ultimately used to construct the sample (1827). Of these firms the e-mail addresses were acquired as they were in the REACH database. After cleaning the list of obvious erroneous e-mail addresses, a total of 1773 firms remained. This number was used for the survey.

⁴⁰ NACE stands for Nomenclature Statistique des Activités Économiques, which is the official classification of industries in the European Union.

Table 5 – Sample statistics					
Sector	IT intensity	Number of firms			% of total
		Before filters	After filters	In final sample	
Telecommunications	High: more than 4,1% of revenues	605	93	93	19,2%
Network management, pc security, automation services, etc.		273	57	57	
Financial institutions, excluding mortgages and retirement funds.		172.702	1.056	200	
Production of machinery	Medium: 2,8% - 4,1% of revenues	493	201	201	17,6%
Production of beverages		103	20	20	
Utilities and energy trade		229	27	27	
Production of optical instruments and the like		64	15	15	
Production of electrical components		190	58	58	
Wholesaler in machinery, apparatus, and accessories	Low: Less than 2,8% of revenues	1937	896	896	63,2%
Construction, finishing of buildings		670	260	260	
Total		177.266	2683	1827	100%

E-mails were sent over a period of 10 days. It appeared that answers came either very quickly (within 24 hours) or not at all. No follow-up e-mails were sent since respondents could not be identified. Of all e-mails sent, about 45 were returned undeliverable. In addition, a number of firms returned an e-mail in which they gave a reason why they would, or could, not fill out the questionnaire. Most cited reasons were that the respondent was either too busy, or that all ICT related tasks had been outsourced, and that therefore nobody in the firm was legible to complete the questionnaire.

In total 83 questionnaires were fully completed, which results in an effective response rate of 4,78%. Not included into this response rate are respondents which indicated that they did not have any knowledge on the concept of open source (18), or respondents that only partly filled out the questionnaire (41). The respondents that only partly filled out the questionnaire mostly abandoned the survey at the fourth page, which was the longest in the whole survey. Response statistics can be found in table 6.

Table 6 – Response statistics		
Item	Number	Percentage of total (1737)
Returned e-mails, undeliverable	45	2,6%
Returned e-mails of non-respondents which indicated as a reason:		
- Too busy / Not interested	14	0,8%
- Outsourced all IT, so can't answer	9	0,5%
- All IT-issues are centralized / No ICT manager in our firm	4	0,2%
- We don't have this software	3	0,18%
- Firm does no longer exist	3	0,18%
- Questionnaire is too long	1	0,06%
- We are not commercial	1	0,06%
Total visits to survey website	255	14,7%
Fully completed questionnaires, of which:	83	4,78%
- Adopters	48	2,76%
- Non-Adopters	35	2,02%
'Don't know OSS' answer (option B)	18	1,04%
Partly completed (not all pages)	41	2,36%

The response set could not be tested for nonresponse errors since respondent information was not known (anonymous) and not all e-mails were sent out at the same time, but over a period of about 10 days. Therefore, late and early respondents could not be compared.

Instrument validation

Reliability

Cronbach's alpha was used to assess the reliability, or internal consistency, of the various instruments. The general agreed upon lower limit for this measure is 0,60-0,70. The results are shown in table 7. As can be seen, all values exceed the lower limit and thereby appear to be reliable.

Table 7 – Instrument validation using Cronbach's α	
Instrument	Cronbach α
Compatibility	0,861
Complexity (original 4 item construct)	0,615
Triability	0,903
Relative advantage (in total, 15 items)	0,806
- Switching costs separately	0,894
- Other items separately	0,768
Technology championship	0,909
Top management support	0,905
Task compatibility	0,759

Validity

In order to validate the constructs and the grouping of items factor analysis has been used. Since the size of the dataset was 83, the cutoff point for factor loadings was set at 0.60. For the constructs compatibility, triability, technology championship, top management support, and task compatibility the factor loadings were sufficient and far above the cutoff point. These constructs were also all related to a single identifiable factor. However, the factor analyses of the constructs complexity and relative advantage showed less straightforward results.

Complexity was measured using a four-item construct. The factor analysis showed that this construct resulted in two factors, each having two variables with high factor loadings. Closer examination of the questionnaire showed that an error in the validation script in the survey could have caused this error. Since the construct contained one 'short' scale that should be sufficient (ease of use) and this construct was measured without errors, this construct was diminished to the single variable, which consisted of that 'short' scale.

In chapter 3 it has already been stated that relative advantage often acts as a container for factors which cannot be categorized under any of the other innovation characteristics. In this case, the term relative advantage has been related to hypotheses on switching costs, hardware cost savings, software cost savings, and various advantages caused by the free availability of the source code. Although source code availability provides a common denominator for many variables, each is linked to a different hypothesis and measured by single item constructs. Only switching costs was a construct consisting of multiple items. This is represented in the factor analysis on relative advantage with high factor scores for the variables that are all related to switching costs. In addition, the reliability for the construct switching costs has been separately calculated, and proved sufficient (see table 7).

All constructs, which have proved valid and reliable, will be summated into single values for further analysis. This will be done by taking the average of the values of the variables in that construct.

Conclusion

This chapter summarized the research methodology for this thesis' research. Although the questionnaire was fairly long and required specific knowledge, 83 responses have been gathered. Still, that is a fairly small dataset considering the number of hypotheses and will lead to a cutback in the number of variables that can be tested. Unfortunately, from a practical point of view it was not feasible to continue to expand the sample.

The majority of the used constructs in the questionnaire demonstrated to be valid and reliable. Besides, the data cleaning and analysis done in this chapter brought to light several issues in the survey which had not been detected before. This included using a wrong construct (for complexity), and a construct item with wrong database scales. Though, these issues could easily be accounted for and did not pose a problem or misinterpretation for the respondents in any way. The dataset thus remains valid. This dataset will be used to undertake statistical testing of the various hypotheses, of which the results will be shown in the next chapter.

Chapter 5 – Results

After having laid the theoretical background, the construction of the hypotheses, and having performed the research, this chapter consists of the results of the statistical analyses which have been performed on the dataset. In this chapter, first a look at the descriptive statistics concerning the responding firms, the independent, and dependent variables. Secondly, the procedure and results of the multivariate logistic regression will be presented and commented. Third, the various hypotheses will be considered. In the end, a short conclusion will be given. The SPSS output for statistical tests that were performed in this chapter can be found in appendix J.

Descriptive statistics

Independent variables

Respondent's professions

Table 9 shows the respondent's professions. The e-mails which were sent out for the survey asked the reader to forward them to the ICT manager, or a person with a comparable function. Although more than half of the respondents indicated that they indeed are the ICT manager, a large part consisted of various management positions and technical (non-management) positions, most of which were system administrators. The other group consisted of software engineers, software architects, consultants, and clerks.

Profession	Frequency	Percent
ICT manager	46	55,4
Management	19	22,9
System administrator	6	7,2
Other	12	14,5
Total	83	100

Sector IT intensity

Table 10 shows the frequencies of the IT intensity of the respondent's sectors. As discussed in chapter 4, the sample was composed of a relatively large number of firms with low IT intensity to compensate for the expected lower levels of OS adoption and IT knowledge in those sectors. As can be seen in the table, it appears that indeed high intensity firms had higher response rates than low intensity firms. However, the distribution of the sample nicely compensated for this, which results in a 50/50 distribution between low IT intensity firms and high/medium IT intensity firms. On the other hand, a large number of respondents, almost 40

percent, opted for the ‘other’ category. This could be due to the fact that the descriptions used for sectors where corresponding to the NACE descriptions⁴¹, which are rather broad. Apparently many firms could not identify themselves with these descriptions.

IT intensity	Frequency	Percent	Original sample division
high	19	22,9	19,2
medium	6	7,2	17,6
low	25	30,1	63,2
other	33	39,8	0
total	83	100	100

Firm categories

Table 11 shows the categorization of the respondent’s organizations according to the indicated gross revenue and number of employees, compared to official EU guidelines. As can be seen most firms are small firms. The unknown category holds firms for which either one of the two measures was not available or which did not deem trustworthy, for example indications of more than 250 employees with a gross revenue of less than 2 million euro.

Category	Gross revenue (mio.)	Number of employees	Frequency	Percent
Micro	< 2	< 10	4	4,8
Small	2-10	10-50	33	39,8
Medium	10-50	50-250	16	19,3
Large	> 50	> 250	10	12,0
Unknown	unknown	unknown	20	24,1

Software characteristics

One of the first question sets in the survey included a ranking of software characteristics. Respondents could indicate how important they rated each of the listed items. The overall results for adopters as well as non-adopters can be seen in table 12, which displays the results of the independent sample t-test. When comparing the importance between adopters and non-adopter groups the only significant difference at α 0,05 occurs with the importance of continuity. Adopters rate continuity *less* important than non-adopters.

At the α 0,10 level source code availability, suppliers and support, and reliability are rated significantly different among adopters and non-adopters. As expected, adopters rate the availability of source code more important than non-adopters. However, both group means indicate that source code availability is not that important, with scores ranging between 2 (not important) and 3 (neutral).

⁴¹ See chapter 4 for an explanation on what NACE stands for and where it was taken from.

Availability of suppliers and support is rated higher by non-adopters as compared to adopters. Again, this is as expected since open source software generally offers less contractual security and support, and therefore should be less adopted by firms which require legal liability and the like.

Both adopters and non-adopters of OSS indicate software reliability very important. Non-adopters indicate a higher perceived importance of software reliability than adopters.

Characteristic	t	Sig (2-tailed)	Adopter mean	N-Adopter mean	Std. Error Diff.
Continuity	3,684	0,000	4,15	4,63	0,125
Source code availability	-1,909	0,060	2,77	2,34	0,224
Suppliers and support	1,679	0,097	3,33	3,69	0,210
Reliability	1,629	0,107	4,48	4,66	0,109
Compatibility	1,546	0,126	4,19	4,43	0,156
Safety	1,493	0,139	4,17	4,37	0,137
Triability	1,257	0,212	3,77	4,00	0,182
Switching costs	1,190	0,237	3,92	4,11	0,166
Hardware costs	0,884	0,379	3,94	3,79	0,171
Complexity	-0,275	0,784	3,56	3,51	0,176
License agreement	0,212	0,832	3,44	3,49	0,227

Dependent variable

The dependent variable is the adoption of open source software. As discussed in chapter 4, respondents could indicate whether they used open source software or whether they did not, and if so, which of the listed OSS they used. The results for OSS adoption can be seen in figure 12.

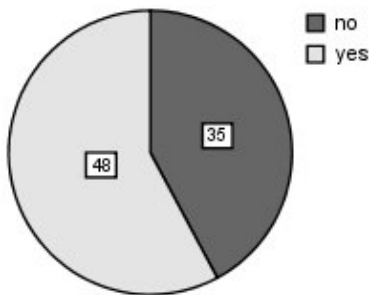


Figure 12. OSS adoption rates.

Software	Frequency	Percent (48 adopters)
Linux	33	68,8
Firefox	24	50,0
MySQL	21	43,8
Perl	20	42,0
PHP	19	39,6
Apache	18	37,5
Open source CMS	9	18,8
Squid	8	16,7
KDE	7	14,6
Open office	6	12,5
Unix	5	10,4
Postgresql	5	10,4
Gnome	5	10,4
Other	23	47,9

Table 13 lists the OSS that the adopters indicated to use in descending order. As can be seen the Linux operating system has the highest adoption percentage, followed by the firefox browser and MySQL database management system. The ‘other’ category consisted of, among others, Interbase / Firebird (3), Samba (2), and Python (2). It should be noted that this list was not a required item in the questionnaire.

Multivariate analysis

In order to examine the variables that contribute to the OSS adoption decision, the technique that deemed most appropriate was logistic regression analysis. This technique is somewhat similar to multiple regression, and can be employed instead of discriminant analysis when it involves a binary dependent variable. Logistic regression analysis provides for better results if the underlying statistical assumptions are violated when compared to discriminant analysis (Hair, Anderson, Tatham & Black, 1998). Before the logistic regression was performed, the data was looked at again.

Data issues

A few things were done before the actual regression analysis was performed. First, summated scales were created for the constructs consisting of multiple items. This involved compatibility, triability, switching costs, technology championship, top management support, and task compatibility.

Secondly, a categorical variable was constructed for the hypothesis concerning unix shop' or windows shop' influence on OSS adoption. If a respondent indicated that the firm used unix as a server operating system over the last five years, regardless of the percentage of installations of unix, it was considered a unix shop (1). If, on the other hand, the respondent indicated that it used the windows operating system over the last five years, it was considered a windows shop (0). If a firm did not opt for one of these two operating systems, that is, it only used Linux, 'other', or indicated not to have this figure, the case got a separate label (2).

Third, the construct concerning the perceived importance of open standards on OSS adoption had missing data since the construct provided for a 'don't know' answer. Since the sample size was relatively small (83), it was not considered an option to delete these cases from the sample. Nor were there any other cases available to replace the missing cases in the sample. Therefore, the missing data were replaced by the mean value of that variable based on all valid responses. However, this method does lead to some data distortion and lower variance for the open standard variable.

The fourth issue involved the sample size. Logistic regression analysis requires a sufficient sample size. The minimum size of twenty cases per group was met and the sizes of each group were not significantly different. However, since the sample size was rather small (<100) and the number of independent variables was rather large, this could result in *overfitting* the model. A minimum suggested ratio is 5-to-1 as the number of cases per

independent variable, while in this case the ratio is 2-to-1. *This posed a serious problem.* In order to take care of this issue the number of determinants had to be cut back.

The number of independent variables was decreased to achieve a 5-to-1 ratio. This asked for a maximum of eight independent variables considering the smallest group having 38 cases. Forward and backward regression techniques could not be used to determine the importance of the various variables since the data would be overfit. The selection of independent variables was based on the following considerations:

- The source of the constructs:
 - previous research vs. own constructs
 - multiple item constructs vs. single item constructs
- The correlation between the independent variables
- The comments provided by the respondents in case of non-adoption
- The theoretical significance following from the discussion in chapter 3.

The variables and the reasoning behind their ultimate inclusion in the regression analysis can be seen in table 14 on the following page. An alternative format is available in appendix H. Although switching costs and supplier independence had similar ratings they were not selected since they were expected to be sufficiently represented by respectively compatibility and continuity. The results will now be discussed.

Table 14 – Overview of included hypotheses / selection procedure			
Determinant	Hypothesis	Selected	Reason
Task compatibility	H ₁ : Perceived task compatibility is positively associated with OSS adoption	Yes	- Multiple item construct - No multicollinearity - Often mentioned by respondents
Skill compatibility	H ₂ : Perceived skill compatibility is positively associated with OSS adoption	Yes	- Not correlated - Often mentioned by respondents
Compatibility	H ₃ : Perceived compatibility with key applications and technological infrastructure is positively associated with OSS adoption	Yes	- Multiple item construct - No multicollinearity - Often mentioned by respondents
Complexity	H ₄ : Perceived complexity is negatively associated with OSS adoption	No	
Triability	H ₅ : Perceived triability is positively associated with OSS adoption	Yes	- Multiple item construct - No multicollinearity
Costs	H ₆ : Perceived switching costs are negatively associated with OSS adoption	No	
	H ₇ : Perceived hardware cost savings are positively associated with OSS adoption	No	
	H ₈ : Perceived software cost savings are positively associated with OSS adoption	Yes	- Multiple item construct - Theoretical significance from chapter 3
Source code availability	H ₉ : Perceived supplier independence is positively associated with OSS adoption	No	
	H ₁₀ : Perceived safety is positively associated with OSS adoption	No	- Showed multicollinearity with reliability, stability
	H ₁₁ : Perceived reliability is positively associated with OSS adoption	No	- Showed multicollinearity with stability, safety
	H ₁₂ : Perceived stability is positively associated with OSS adoption	No	- Showed multicollinearity with reliability, safety
	H ₁₃ : Perceived continuity is positively associated with OSS adoption	Yes	- No multicollinearity - Theoretical significance from chapter 3
	H ₁₄ : Perceived open standards compliance is positively associated with OSS adoption	No	
Technology championship	H ₁₅ : Perceived OSS championship is positively associated with OSS adoption	No	- Showed multicollinearity with top manag. support
Third party support	H ₁₆ : Perceived third party support is positively associated with OSS adoption	Yes	- No multicollinearity - Theoretical significance from chapter 3
Top management support	H ₁₇ : Perceived top management support is positively associated with OSS adoption	Yes	- Multiple item construct - No multicollinearity - Often mentioned by respondents
IS slack resources and budget	H ₁₈ : Abundance of slack resources is negatively associated with OSS adoption	No	
	H ₁₉ : Limited financial slack resources in combination with enough slack in human resources is positively associated with OSS adoption.	No	
IT in business strategy	H ₂₀ : Perceived importance of IT to the business strategy is positively associated with OSS adoption	No	
Community size	H ₂₁ : Perceived community size is positively associated with OSS adoption	No	
Platform standards	H ₂₂ : A background as a 'Unix shop' is positively associated with OSS adoption	No	

Results

To test the model the ENTER method was used and a cutting score of 0,57 in according to the weighted group sizes. The result of the logistic regression is shown in table 15.

Table 15 – Results of logistic regression analysis						
Factor	B	S.E.	Wald	df	Sig.	Exp(B)
Compatibility	3,093	1,062	8,475	1	0,004	22,040
Triability	1,497	0,688	4,729	1	0,030	4,466
Software costs	0,455	0,556	0,671	1	0,413	1,577
Continuity	-0,569	0,548	1,077	1	0,299	0,566
Third party support	0,194	0,567	0,117	1	0,732	1,214
Top management support	0,025	0,572	0,002	1	0,965	1,026
Task compatibility	1,200	0,667	3,232	1	0,072	3,320
Skill compatibility	-0,216	0,446	0,234	1	0,628	0,806
Constant	-17,317	4,566	14,381	1	0,000	0,000

Model fit

The ‘-2 log likelihood’ (-2LL) value represented by the model chi square indicates that the model is significant. This means that the null hypothesis is rejected, which states that none of the independents is linearly related to the log odds of the dependent variable. In addition, the Hosmer and Lemeshow ‘Goodness of Fit’ test provides evidence that the model’s estimates fit the data at an acceptable level. When considering the classification matrix the model displays a correct percentage of 85,5 percent, which is over 25 percent more as compared to the base model (57,8 percent). In addition, the model provides explanation for both groups separately.

Coefficients

As can be seen in table 15 three individual constructs are found to be significant, using the Wald statistic and its p-value. In addition, these constructs also show significant correlations to the dependent variable. Therefore, from the eight selected independent variables, the variables compatibility, task compatibility, and triability show to be significantly different from zero as opposed to the other five variables. Thus support was found for hypotheses 1, 3, and 5. Support was not found for the other hypotheses that were included in the model.

Assumptions

As mentioned before, variables which showed signs of (multi-) collinearity were not included in the model. The remaining independent variables did not show any correlation in the final model. Secondly, the standardized residuals were tested for outliers at three standard deviations. No outliers were found.

Alternative regression analyses

Two alternative regression analyses were performed for additional testing. First, top management was swapped for technology championship since these two variables showed significant correlation. Secondly, a variable software quality was constructed consisting of the variables reliability, safety, and stability, since these three variables showed high correlations.

Enter technology championship for top management support

Swapping technology championship for top management support was done because these factors showed high correlation in the pre-selection. The results of the altered regression analysis show no difference as can be seen in table 16. The same factors seem relevant, with a decreased significance of task compatibility. Technology championship, just like top management support, does not seem relevant, although it has slightly higher significance than top management support. The overall model $-2LL$ therefore shows a slight improvement. This supports the idea that these two constructs might resemble the same component, something that will be further discussed in the following chapter.

Table 16 – Results of alternative logistic regression analysis 1: Enter technology championship for top management support						
Factor	B	S.E.	Wald	df	Sig.	Exp(B)
Compatibility	2,915	1,086	7,210	1	,007	18,445
Triability	1,350	,677	3,977	1	,046	3,857
Software costs	,476	,574	,690	1	,406	1,610
Continuity	-,710	,614	1,338	1	,247	,491
Third party support	,338	,569	,354	1	,552	1,403
Technology championship	,622	,511	1,479	1	,224	1,862
Task compatibility	1,094	,674	2,633	1	,105	2,985
Skill compatibility	-,171	,446	,147	1	,702	,843
Constant	-17,841	4,629	14,854	1	,000	,000

Since initially, compatibility also showed significant correlation to technology championship, an additional test was done while removing compatibility. When both top management and compatibility were removed, and technology championship entered the equation, its coefficient did show to be significant at a level of α 0,05. However, the overall model's explanatory power decreased.

Adding a new component: Software quality

Software quality, representing perceptions on safety, reliability, and stability, was entered into the equation by removing top management support. This was done since these variables (three quality variables vs. top management support) showed the highest correlation among all factors in the model. The results can be seen in table 17. Software quality led to a small improvement in the model in terms of the $-2LL$ value. In addition the percentage of correct

predictions went up to 86,7%, an improvement of 1%. At the individual level, software quality does not seem to be relevant in the model with a p-value of 0,551.

Table 17 – Results of alternative logistic regression analysis 2: Enter software quality for top management support						
Factor	B	S.E.	Wald	df	Sig.	Exp(B)
Compatibility	3,084	1,062	8,431	1	,004	21,853
Triability	1,468	,659	4,959	1	,026	4,339
Software costs	,493	,566	,759	1	,384	1,638
Continuity	-,600	,558	1,155	1	,282	,549
Third party support	,210	,561	,140	1	,708	1,234
Software quality	,334	,559	,356	1	,551	1,396
Task compatibility	1,127	,674	2,800	1	,094	3,086
Skill compatibility	-,174	,444	,154	1	,695	,840
Constant	-18,061	4,783	14,257	1	,000	,000

Conclusion

This chapter presented the results of the statistical analyses. Besides the ‘common’ descriptive statistics a logistic regression analysis was performed. The sample size in combination with the large number of hypotheses asked for a rigorous reduction in the number of variables to be included in the logistic regression analysis. Since statistical tests, i.e. backward and forward logistic regression, could not be used to select for the most contributing factors, a combination of quantitative and qualitative selection criteria was employed. Finally, a model of eight variables was tested. Three variables showed to be significant: compatibility, task compatibility, and triability. In addition, technology championship appeared to be somewhat interchangeable with top management support. The next chapter will discuss the various results that have been found.

Chapter 6 - Discussion of results

The final part of this thesis involves the discussion of the results, which followed from the statistical analyses in chapter 5. In addition, the limitations of the research as well as some implications for management and suggestions for future research in this area will be outlined. First however, the results will be discussed for the various hypotheses which were presented in chapter 3 and which were included in the logistic regression model. In addition, the discussion will be related back to the classification scheme by Fichman (2000), which allowed for the categorization of these hypotheses, and the framework for OS adoption by Kwan & West (2003).

Discussion

In this thesis, twenty-two factors have been proposed which are likely to influence the adoption of OSS by for-profit organizations. Due to the large number of hypothesized relationships in combination with a relatively small sample, the number of factors that were finally used to construct the research model had to be limited to eight. Of these eight, three turned out to have a significant effect on the OSS adoption decision. These three factors provide an answer to the problem statement that was stated in the introduction of this thesis:

Which factors influence the adoption of open source software among for-profit firms?

All variables will now be considered and a clarification will be proposed for their relevance in the model.

Relevant variables in the model

Three variables were found relevant to the adoption decision: compatibility, triability, and task compatibility. Somewhat surprisingly, these three variables are not the same as the ones that showed a high ranking in table 12. Apparently, when considering overall effects instead of individual factors, the ranking does not hold. The three relevant factors will now be discussed.

Compatibility

In chapter three it was proposed that compatibility in terms of technological infrastructure and business critical, *key* applications, is positively related to OSS adoption (H_1). It is reasoned that key applications prohibit firms to switch to other software since such software is not likely to be very portable in terms of operating system, API, and third party applications. In

addition, in the case of proprietary systems data is often saved in a proprietary data format which is (more than often) not compatible with open standard formats which OSS adheres to. This makes data integration and/or portability between proprietary (key) applications and OSS more difficult. Finally, the organization's strategy regarding the corporate IS/IT architecture has to allow for OSS adoption, e.g. many firms standardize on a certain (proprietary) technology, which makes a switch later on very difficult, if not impossible. This study supports these claims.

Perceived compatibility showed a positive relation to OSS adoption. The construct used for compatibility showed extremely significant in the model. In addition, comments provided by respondents often related to this variable. These are discussed further on.

Triability

Perceived triability was hypothesized to have a positive relationship to OSS adoption (H₅). Logically, the ability to test software can be expected to have a positive relationship to adoption. In the case of OSS this relationship can be expected to be strong since OSS scores high on triability. After all, OSS is always available without any limitations in terms of time, costs, and features. The data confirmed this reasoning. Perceived triability is positively related to OSS adoption.

Task compatibility

The extent to which required software functionalities can be matched by OSS is referred to as task compatibility. In chapter three it was hypothesized that perceived task compatibility is positively related to OSS adoption (H₁). The results confirm this. Perceived task compatibility is positively related to OSS adoption.

Irrelevant variables in the model

Software costs (H₈), continuity (H₁₃), third party support (H₁₆), top management support (H₁₇), and skill compatibility (H₂) did not appear to have a significant contribution to the model. Thus, support was not found for these hypotheses. In addition, software quality (representing H₁₀, H₁₁, and H₁₂) was included in the model but did not show any effect on the adoption decision. Other excluded hypotheses have not been tested at all.

Discussion of the model

The three relevant variables included in the model are basically all classic innovation characteristics. Although task compatibility is not exactly identical to (technological)

compatibility, it is conceptually similar. Task compatibility can be seen as a part of compatibility. When referring back to the matrix of Fichman (1992) as presented in chapter two, classic innovation characteristics are part of all quadrants, including type I and type II technologies. This is reasonable, since solely the fact that software is OSS, cannot dictate whether such software poses a high knowledge burden or network externalities in specific contexts. Also, table 13 showed that simple OSS technology can relatively fast gain popularity despite negative network externalities, as seen by the second place of the Firefox internet browser in the OSS adoption ranking⁴². In sum, classic innovation characteristics seem to be most determining for OSS adoption given the eight variables fitted in the model.

A justification for the limited number of contributing variables can be sought in the fact that this research only considers adoption as a binary decision (more on this in the ‘limitations’ section). Larsen, Holck & Pedersen (2004) pose that, especially for larger organizations, a major barrier towards the *adoption* of OSS lies in the “*enterprise architecture, which constitute the strategic framework for all investments in IT*” (Larsen, Holck & Pedersen, 2004, p.13). This is identical to what is reasoned in this thesis by using the concept of compatibility. However, most of the (*known*) respondents from this dataset are small firms whereas the paper from Larsen, et al. relates this determinant primarily to observations in large firms. In sum, when considering the adoption stage of OSS as a binary decision, compatibility does show to be of major importance with the inclusion of compatibility and task compatibility.

When considering the other relevant variable, triability, once more the question arises why especially this variable is important over e.g. software cost or supplier independence. Again, a possible explanation could be the absence of a measure of adoption. The triability characteristics of OSS could be extremely important in the initial adoption stage, since the opportunity costs of trying the software is, besides employee time, close to zero. In that way, adoption can occur relatively unnoticed or even unauthorized. In later adoption stages, or in functions that require more widespread organizational overhaul, these triability aspects might be less important.

Besides the three relevant variables discussed, the five other variables included in the model were not found to be of any significant importance. Both software costs and continuity, as argued in chapter three, seemed the most important characteristics of OSS. The, often discussed, source code availability was not expected to be essential, but rather the indirect

⁴² The Firefox browser was first made available to the public in 2004.

effects it results in: supplier independence and lower software costs due to the lack of any license fees.

Especially software costs are expected to be an important reason to adopt OSS for small firms (Larsen, Holck & Pedersen, 2004), which make up the largest (*known*) portion of the respondents. However, although software costs might pose significant cost savings, these cost savings should be relatively small if an organization is incompatible with the architecture, infrastructure, or skill set that OSS demands. In such cases, switching costs might make savings on software licenses irrelevant. Thus once again, this factor might not be relevant at the adoption stage. At least, as the model shows, less relevant than compatibility issues, which, financially speaking, translate into switching costs.

The nonappearance of supplier independence, in the model represented by continuity, is difficult to explain with the current dataset. Although table 12 showed significant differences on this item between adopters and non-adopters, the model does not. One hypothesized explanation might be that since most firms do not possess the skills to work with the source code themselves, they still have to rely on outside parties for the service, maintenance, etc., of the software. Thus, they remain dependent on certain companies, whether it is a supplier or software company. This effect might even be strengthened by the lack of support organizations for OSS as compared to proprietary software (although decreasing). This could also be a possible cause to explain why third party support cannot be used to distinguish between adopters and non-adopters.

Overall, software quality issues, direct as well as indirect, and free availability do not seem important at the adoption stage. However, this result might also be due to the research target population, which are for-profit *organizations*. The much debated software quality aspect, the absence of license fees, and perhaps other determinants, might be more essential at the level of individual adoption. At that level, the zero price and free modification are expected to be more important factors than the three found in this thesis' model.

Top management support is the fourth variable not included. This could be due to the fact that top management support is especially needed for the organizational introduction of OSS, which again, might not be the adoption stage among adopters in the dataset. As long as OSS is used for specialized tasks like web server or firewall tasks, it is not expected to involve top management and in that way influence the adoption decision.

The exclusion of skill compatibility, on the other hand, seems harder to explain. Apparently, the skills required to operate and maintain OSS are not significantly different than for proprietary software. This statement more or less stands or falls with the introduction

of the Linux operating system at the desktop. As long as this change is necessary, the required skill set for common business users is expected to be rather small. Few cases are known in literature where firms fully switch to a Linux environment for the desktop.

When referring back to figure 10, the categorization scheme of Fichman (2000), it can be seen that the three relevant variables all fall in the ‘technology-organization combination’ category. In addition, the adoption framework put forward by Kwan & West (2003), as presented in figure 9, shows the expected adoption process. The factors that have been found significant in this thesis cannot clearly be mapped to single factors in this model. For example, compatibility would likely be an issue when comparing the OSS to the application context in terms of already used products and the associated risk of introducing another technology, possibly requiring the adoption of new standards or protocols. On the other hand, compatibility also regards the key applications of the firm, which are likely to be very related to the industry the firm operates in. Triability could possibly be represented by OS attitudes as well as application context, where OSS would decrease the adoption risk since it could be thoroughly tested without any side effects.

Implications for management

Besides the general conclusions that can be drawn from this paper, this research offered some insights into general considerations among firms in adopting OSS that might be relevant to the managers. Besides the quantitative data gathered, some qualitative data has been gathered, consisting mainly of comments provided by non-adopters. Table 16 shows reactions by respondents to the question why they did not adopt OSS.

Table 18 – Reasons for non-adoption of OSS	
Reason	Count
Hard support needed (legal, contracts, etc)	10
No need yet for such software / Redundant	8
Required skills are not available	7
Standardized on MS products	6
Key applications require MS products	6
Centralized IT management or outsourced IT	3
Limited IT personnel	2
Too expensive to maintain	2
Literally: compatibility	2
Software quality doubtful (stability, reliability, etc)	2
Is not part of corporate IS/IT strategy	1

As can be seen in table 18, most arguments for non-adoption are related to lock-in to Microsoft products, the lack of skills to employ/deploy such software, the fact that such

software is not required and, above all, the perceived lack of support in terms of 'hard' contracts and legal guarantees.

The argument that there is no need for OSS is unexpected. Although OSS is relatively underrepresented in certain types of software, the fact that it is free would make it more favorite for adoption. Again, this could be due to the before mentioned argument of compatibility with IT architecture. Organizations might focus on software which offer a complete solution for many tasks, although OSS might be better for certain individual tasks.

Most implications from this table are for companies which offer OSS or related products or services. Hard support is still asked for as well as skills to implement and help maintain OSS.

Limitations

Results and conclusions from this thesis should be read with some caution. First, although theoretically underpinned, the large number of hypotheses has led to some limitations. From a practical point of view, the number of variables to be measured and the resulting length of the survey could be seen as a reason for the low response rate. This has limited the number of hypotheses that could be tested since the sample size, in relation to the number of variables, was rather small for the technique used, logistic regression. In addition, the Wald test used to estimate the importance of individual coefficients is expected to be less reliable at smaller sample sizes.

Secondly, the research involved a wide array of issues concerning the ICT function like technical considerations, software purchasing issues, and open standards. Although ICT managers were the target group of the survey, it cannot be determined to what extent each individual had the required knowledge of all aspects. This is certainly the case for non-adopters. These respondents might be less knowledgeable on OSS or open standards. On the other hand, respondents from other business functions could be even less apt to fill out the questionnaire, and the questionnaire was set up in such a way as to sort out unknowledgeable respondents as much as possible.

Thirdly, the questionnaire does not consider the various levels of adoption that could exist. Since this thesis has limited itself to adoption vs. non-adoption this is not an issue, but differentiating between various adoption levels, as pointed out by Kwon & Zmud (1987), could prove important in recognizing more factors of differing importance at different stages.

Suggestions for future research

This section will suggest some topics that are interesting for further research. As already mentioned in previous chapters, the supply of OSS, e.g. developer's motivation and project organization, has gained relatively much attention. Less is known about the demand side of OSS. This thesis has tried to consider many aspects on the demand side, some of which might be interesting to consider separately, especially those which were deleted because of overfitting the model. Some other factors which were left out before hand could also seem important, above all the influence of the type of license agreements on OSS adoption levels.

A related area of interest involves researching additional constructs. Of the twenty-two variables used, the factor analysis showed only eight components. This number could even be reduced to six when the summated scale software quality was used. This is unexpected, since (especially) the constructs taken from previous research should only load on single components. When regarding the components, classic innovation characteristics seem to load on one factor. In addition, IS/IT budget and human resources, and importance of IT to the business strategy load on one component as well as top management support and technology championship. Most of the components seem to represent identifiable concepts. Although more detail is beyond the scope of this text, researching these effects might provide for better results.

Third, the foundation of this research has focused on Rogers' (1995) concept of diffusion of innovations. Other models, which focus especially on the adoption of information systems, i.e. software, like the technology acceptance model by Davis (1989), might provide additional insights in OSS adoption determinants. In addition, since a number of non-adopters indicated not to use OSS because they were satisfied with existing software, a need-pull model might be relevant. Need-pull factors relate to the satisfaction of organizations with their current systems to adoption of new systems, as described by Chau & Tam (2000).

Fourth, one could look beyond the adoption decision. As discussed, adoption stands far apart from infusion, i.e. full integration of the technology. This research only considers what factors made a firm to adopt or not adopt OSS. Relatively little is known on the dynamic context, i.e. the decisions that lead to OSS adoption and the extent of OSS adoption and infusion. More detailed research involving case studies could be used to refine and test the model and the importance of factors in such situations. Also, one could focus on the adoption level of a few OS programs and consider the importance of various factors at the various stages by means of a discriminant analysis.

The fifth recommendation concerns the other categories of the Fichman classification scheme that have not been included in this thesis. This research has not considered the category ‘technology & diffusion environment’, which involves aspects such as propagating institutions, which have been shortly discussed in chapter 2.

Bibliography

- Agrain, P.(2002). *A framework for understanding the impact of GPL copylefting vs. non copylefting licenses*. MIT working paper. Retrieved February 5, 2005, from the World Wide Web: <http://opensource.mit.edu/papers/aigrain2.pdf>
- Alavi, M., & Leidner, D.E. (2001). Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 25 (1), 107-136.
- Berlecon. (2002a). *FLOSS final report - part 1: free/libre and open source software: survey and study: Use of open source software in firms and public institutions, evidence from Germany, Sweden and UK*. Berlin: Berlecon Research GmbH.
- Berlecon. (2002c). *FLOSS final report - part 3: free/libre and open source software: survey and study. Basics of open source software markets and business models*. Berlin: Berlecon Research GmbH.
- Blecherman, B. (1999). *The cathedral versus the bazaar (With apologies to Eric S. Raymond): An economic and strategic look at open-source software*. Retrieved December 16, 2004, from the World Wide Web: http://www.ite.poly.edu/htmls/chapel_printable.htm
- Bonaccorsi, A., & Rossi, C. (2003a). *Licensing schemes in the production and distribution of open source software. An empirical investigation*. Pisa, Italy: Sant' Anna School of Advanced Studies, Institute for Informatics and Telematics.
- Bonaccorsi, A., & Rossi, C. (2003b). Why open source software can succeed. *Research Policy*, 32 (7), 1243-1258.
- Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2004). *Entry strategies under dominant standards - Hybrid business models in the open source software industry*. Retrieved January 15, 2005, from the World Wide Web: <http://opensource.mit.edu/papers/bonnacorsirossiGiannangeli.pdf>
- Brynjolfsson, E., & Kemerer, C.F. (1997). Network externalities in microcomputer software: An econometric analysis of the spreadsheet market. *Management Science*, 42 (12), 1627-1647.
- Casanova, C. (2003). *Open source risk mitigation process*. Bethesda, USA: The SANS Institute. Retrieved February 15, 2005 from the World Wide Web: <http://www.sans.org/rr/whitepapers/bestprac/1174.php>
- Chau, P.Y.K., & Tam, K.Y. (1997). Factors affecting the adoption of open systems: An exploratory study. *MIS Quarterly*, 21 (1), 1-24.
- Chau, P.Y.K., & Tam, K.Y. (2000). Organizational adoption of open systems: a 'technology-push, need-pull' perspective. *Information and Management*, 37, 229-239.

- Churchill, G.A. (2001). *Basic Marketing Research* (4th ed.). Dryden: The Dryden Press.
- Commission of European Community. (2002). Aanbeveling van de commissie van ... tot wijziging van Aanbeveling 96/280/EG betreffende de definitie van kleine en middelgrote ondernemingen [Recommendation of the Commission of .. to the changing of Recommendation 96/280/EG concerning the definition of small and medium-sized enterprises]. Retrieved March 31, 2005, from the World Wide Web: http://europa.eu.int/comm/enterprise/consultations/sme_definition/consultation2/153_sme_definition_25_6_2002_pp1_10_nl.pdf
- Cooper, R.B., & Zmud, R.W. (1990). Information technology implementation research: A technology diffusion research. *Management Science*, 36 (2), 123-139.
- Coppola, C., & Neelley, E. (2004). *Open source – opens learning: Why open source makes sense for education*. Phoenix: The r-smart group. Retrieved February 22, 2005, from the World Wide Web: <http://www.rsmart.com/assets/OpenSourceOpensLearningJuly2004.pdf>
- Davis, F.D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *Behaviour & Information Technology*, 18 (4), 319-339.
- Dwan, B. (2004). Open source vs. closed. *Network Security*, 2004 (5), 11-13.
- e-Cology Corporation. (2003). *Open source software in Canada: A collaborative fact finding study*. Retrieved February 21, 2005, from the World Wide Web: <http://www.e-cology.ca/canfloss/report/>
- Fichman, R.G. (2000). The diffusion and assimilation of information technology innovations. In R.W. Zmud (Ed.), *Framing the domains of IT management: Projecting the future through the past* (Chapter 7). Cincinnati, OH, Pinnaflex Educational Resources, Inc.
- Fichman, R.G. (1992). Information technology diffusion: A review of empirical research. *Proceedings of the thirteenth international conference of information systems*, pp. 195-206. Dallas.
- Fitzgerald, B., & Kenny, T. (2003). *Open source software can improve the health of the bank balance – The Beaumont Hospital experience*. University of Limerick, Ireland: Fitzgerald, B., & Kenny, T. Retrieved March 15, 2005, from the World Wide Web: <http://www.netproject.com/docs/Beaumont.pdf>
- Frels, J.K., Shervani, T., & Srivastava, R.K. (2003). The integrated networks model: explaining resource allocations in network markets. *Journal of Marketing*, 67 (January), 29-45.
- Fugetta, A. (2003). Open source software – an evaluation. *The journal of systems and software*, 66, 77-90.

- Ghosh, R.A. (1998). *Cooking pot markets: an economic model for the trade in free goods and services on the internet*. Retrieved December 15, 2004, from the World Wide Web: <http://dxm.org/fm/cookingpot/>
- Ghosh, R.A., & Glott, R. (2003). *Open standards and open source software in The Netherlands: A quantitative study on attitudes and usage in Dutch authorities*. Maastricht: MERIT, Universiteit Maastricht. Retrieved January 12, 2005, from the World Wide Web: <http://www.for-the-people.org/blog/entries/entry.2003-12-16.0227>
- Hair, J.F., Jr., Anderson, R.E., Tatham, R.L., & Black, W.C. (1998). *Multivariate Data Analysis* (5th ed.). Upper Saddle River: Prentice Hall.
- Hamel, G., & Prahalad, C.K. (1990). The core competence of the corporation. *Harvard Business Review*, pp. 79-91.
- Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research Policy* 32, 1159-1177.
- Johnson, K. (2001). *A descriptive process model for open-source software development*. Calgary, Alberta, Canada: University of Calgary, Department of Computer Science. Retrieved November 20, 2004, from the World Wide Web: <http://sern.ucalgary.ca/students/theses/KimJohnson/kjohnsonMSc.pdf>
- Knubben, B.S.J. (2004). *Investeren in openheid: Een analyse van TCO-onderzoeken betreffende open source software* [Investing in openness: An analysis of TCO-research concerning open source software]. Den Haag: ICTU, Programma OSOSS.
- Krogh, G., von, & Hippel, G. von. (2003). Special issue on open source software development. *Research Policy*, 32 (7), 1149-1157.
- Kwan, S.K., & West, J. (2005). A conceptual model for enterprise adoption of open source software. In S. Bolin (Ed.), *The standards edge: Open season* (pp. 274-301). Ann Harbor, Michigan: Sheridan Books.
- Kwon, T.H., & Zmud, R.W. (1987). Unifying the fragmented streams of information systems implementation research. In: R. Boland & R. Hirschheim (Eds.), *Critical issues in information systems research*. Chichester, England: John Wiley & Sons Ltd.
- Larsen, M.H., Holck, J., & Pedersen, M.K. (2004). *The challenges of open source software in IT adoption: Enterprise architecture versus total cost of ownership*. Copenhagen, Denmark: Copenhagen Business School, Department of Informatics. Retrieved Mai 20, 2005, from the World Wide Web: <http://w3.msi.vxu.se/users/per/IRIS27/iris27-1135.pdf>
- Lerner, J., & Tirole, J. (2001). The open source movement: key research questions. *European Economic Review*, 45, 819-826.

- Lerner, J., & Tirole, J. (2002). Some simple economics of open source. *Journal of industrial economics*, 50 (2), 297-234.
- Levy, S. (2000). *Hackers: heroes of the computer revolution*. New York: Penguin.
- Michalec, G. (2002). *Free software: history, perspectives, and implications*. Oxford, Ohio, United States: Miami University, School of Interdisciplinary Studies. Retrieved January 20, 2005, from the World Wide Web: <http://greg.primate.net/sp/thesis.pdf>
- Moore, G.C., & Benbasat, I. (1991). Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information Systems Research*, 2 (3), 173-191.
- Mustonen, M. (2003). Copyleft – the economics of Linux and other open source software. *Information Economics and Policy*, 15, 99-121.
- Newell, S., Swan, J.A., & Galliers, R.D. (2000). A knowledge focused perspective on the diffusion and adoption of complex information technologies: the BPR example. *Information Systems Journal*, pp. 239-259.
- Nichols, D.M., & Twidale, M.B. (2003). The usability of open source software. *First Monday*, 8 (1). Retrieved January 20, 2005, from the World Wide Web: http://www.firstmonday.org/issues/issue8_1/nichols/
- Nuvolari, N. (2003). *Open source software development: some historical perspectives*. ECIS working papers, Eindhoven Centre for Innovation Studies, Faculty of Technology Management. Retrieved January 20, 2005, from the World Wide Web: <http://opensource.mit.edu/papers/nuvolari.pdf>
- Oksanen, V., & Välimäki, M. (2002). *Evaluation of open source licensing models for a company developing mass market software*. Retrieved February 20, 2005, from the World Wide Web: http://www.hiit.fi/de/valimaki_oxsanen_lawtech_2002.pdf
- Orlikowski, W.J. (1995). *Evolving with notes: Organizational change around groupware technology*. Cambridge, USA: W.J. Orlikowski. Retrieved February 25, 2005 from the World Wide Web: <http://ccs.mit.edu/papers/CCSWP186.html#org>
- Overby, E.M., Bharadwaj, A.S., & Bharadwaj, S.G. (2004). *An investigation of firm-level open source software adoption: theoretical and practical implications*. Atlanta: Overby, E.M., Bharadwaj, A.S., & Bharadwaj, S.G. Retrieved December 20, 2004, from the World Wide Web: http://userwww.service.emory.edu/~eoverby/files/overby_open_source_adoption_study.pdf
- Pijpers, A.G.M, Montfort, K., van, & Heemstra, F.J. (2002). Acceptatie van ICT: Theorie en een veldonderzoek onder topmanagers [Adoption of ICT: Theory and fieldresearch among topmanagers]. *Bedrijfskunde*, 74 (4), 76-84.
- Premkumar, G. (2003). A meta-analysis of research on information technology implementation in small business. *Journal of Organizational Computing and Electronic Commerce*, 13 (2), 91-121.

- Rai, A., & Patnayakuni, R. (1996). A structural model for CASE adoption behavior. *Journal of Management Information Systems*, 13 (2), 205-234.
- Rajagopal, P. (2002). An innovation-diffusion view of implementation of enterprise resource planning systems and development of a research model. *Information & Management*, 40 (2), 87-114.
- Rathnam, R.G., Johnson, J., & Joseph Wen, H. (2004). Alignment of business strategy and IT strategy: A case study of a Fortune 50 financial services company. *Journal of Computer Information Systems*, (XLV, 2, 1).
- Raymond, E. S. (1999). *The cathedral & the bazaar. Musings on Linux and open source by an accidental revolutionary*. Sebastopol, CA: O'Reilly & Associates, Inc.
- Robertson, T.S., & Gatignon, H. (1986). Competite effects on technology diffusion. *Journal of Marketing*, 50 (3), 1-12.
- Rogers, E. M. (1983, 1995). *Diffusion of innovations* (3rd, 4th ed.). New York: The Free Press.
- Rogers, E.M., & Shoemaker, F.F. (1971). *Communication of innovations: A cross-cultural approach*. New York: The Free Press.
- Shapiro, C., & Varian, H.R. (1999). *Information Rules*. Boston: Harvard Business School Press.
- Stewart, K.J., Ammeter, A.P., & Maruping, L.M. (2005). A preliminary analysis of the influences of licensing and organizational sponsorship on success in open source projects. *Proceedings of the 38th Hawaii International Conference on System Sciences, January 2005*. Retrieved February 20, 2005, from the World Wide Web: <http://csdl.computer.org/comp/proceedings/hicss/2005/2268/07/22680197c.pdf>
- Swanson, E.B. (1994). Information systems innovation among organizations. *Management Science*, 40 (9), 1069-1092.
- The Dravis Group. (2003). *Open source software: Case studies examining its use*. San Fransisco: The Dravis Group. Retrieved March 20, 2005, from the World Wide Web: http://www.pgsql.com/pdf/OpenSourceSoftware_Dravis_.pdf
- Tornatzky, L.G., & Klein, K.J. (1982). Innovation characteristics and innovation adoption implementation: A meta-analysis of findings. *IEEE transactions on engineering management*, (EM-29-1), 28-45.
- Tornatzky, L.G., & Fleischer, M. (1990). *The process of technological innovation*. Toronto: Lexington Books.
- Turban, E., McLean, E., & Wheterbe, E. (1999). *Information technology for management* (2nd ed.). New York: John Wiley & Sons, Inc.

- Ward, J., & Peppard, J. (2002). *Strategic planning for information systems* (3rd ed.). Chichester: John Wiley & Sons, Ltd.
- Watson, B. (2003). No title. Retrieved December 15, 2004, from the World Wide Web: http://expertanswercenter.techtarget.com/eac/knowledgebaseAnswer/0,295199,sid63_gci983994,00.html
- Wayner, P. (2000). *Free for all: how linux and the free software development undercut the high-tech titans*. New York: HarperBusiness.
- West, J. (2003). How open is open enough? Melding proprietary and open source platform strategies. *Research Policy*, 32 (7), 1259-1285.
- West, J., & Dedrick, J. (2003). *Adoption of open source platforms: An exploratory study*. Presented at HBS – MIT Sloan free/open source software conference – New models of software development. Retrieved December 5, 2004, from the World Wide Web: <http://opensource.mit.edu/papers/west.pdf>
- Whitten, J. L., Bentley, L.D., & Dittman, K.C. (2001). *Systems analysis and design methods* (5th ed.). New York: McGraw-Hill.
- Zmud, R.W. (1982). Diffusion of modern software practices: Influence of centralization and formalization. *Management Science*, 28 (12), 1421-1431.
- Zmud, R.W., & Apple, L.E. (1992). Measuring information technology infusion. *Journal of Product Innovation Management*, 9 (2), 148-155.

Appendices

Appendix A: Open source timeline

The open source timeline has been composed from various internet sources.

Table 17 - Open source timeline	
When?	Event
1971	Richard Stallman begins his career at MIT in a group that uses only free software
1978	Donald Knuth of Stanford University begins working on the Tex system, and distributes it as free software.
1979	Eric Allman writes a precursor to Sendmail, called Delivermail. It is shipped with 4.0 and 4.1 BSD Unix.
1980	Early era of nonproprietary software for academic use is largely over. Most software has become proprietary; that is, it is privately owned and its source code is not publicly available.
1983	<ul style="list-style-type: none"> - Richard Stallman writes the GNU Manifesto, in which he calls for a return to publicly shareable software and source code. - GNU Project begins. Developers begin creating a wide range of generally Unix-like tools and software such as compilers. The kernel is not covered by these early efforts. - TCP/IP protocols (which are open standards) adopted by the U.S. military.
1984	AT&T began enforcing its intellectual property rights on Unix
1985	Richard M. Stallman starts the Free Software Foundation, a nonprofit organization to manage and support the development of free software.
1986	Larry Wall develops the Perl programming language.
1988	Richard M. Stallman and the FSF created the GNU General Public License (GPL).
1989	Cygnus, the first company to identify business opportunities in free software, is founded.
1990	Python programming language invented by Guido Van Rossum at CWI in Amsterdam.
1991	Linus Torvalds creates version 0.01 of Linux in August. The first "official" version, version 0.02, appears in October
1993	FreeBSD project begins; first CD-ROM and Web distribution of FreeBSD 1.0.
1994	<ul style="list-style-type: none"> - Official version of Linux 1.0 released - Marc Ewing begins the Red Hat Linux distribution. Like the Debian distribution, it is intended to improve on the then-dominant Softlanding Linux System (SLS) distribution.
1995	<ul style="list-style-type: none"> - Apache webserver 1.0 released - Debian Social Contract
1997	<ul style="list-style-type: none"> - Eric S. Raymond wrote an analysis of free/open source software development entitled <i>The Cathedral and the Bazaar</i>. - Linux receives numerous industry awards, including InfoWorld's Network Operating System Award - The Open Source Initiative was founded
1998	<ul style="list-style-type: none"> - Apache and derivatives now running 50 per cent of all Web sites. - Netscape announces freeware strategy for its browser; launches Mozilla.org to promote freeing of its source code. - Corel Corp. announces supports for Linux and plans to port its software to an open source OS platform. - IBM Corp. announces support for Apache.
1999	<ul style="list-style-type: none"> - IBM, Compaq Corp., Dell Computer and Hewlett-Packard start selling systems designed for use with Linux OS. - Linux 2.2.0 is released, promising refinements that will make it easier for businesses to adopt it as their OS. - VA Linux Systems sets new record for the largest first-day gain of any initial public offering, capping a year of huge gains for Linux- related stocks. - Red Hat software, a popular commercial Linux distributor, goes public raising over 68 million dollars in venture capital.

2000	<ul style="list-style-type: none">- Core Corp.'s stock soars after announcing a version of its Linux- based software will allow users to run Windows applications over any connection.- IBM announced that it would dedicate nearly \$1 billion to open source development, advertising, and services- Sun Microsystems open sources StarOffice, its productivity application suite.
------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Appendix B: The OSD License

Retrieved at http://www.opensource.org/docs/definition_plain.php on 14 March, 2005.

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

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.

10. License Must Be Technology-Neutral

No provision of the license may be predicated on any individual technology or style of interface.

Appendix C: Overview of software licenses

Table 18 - Overview of software licenses							
Software license	Available at no cost	Distribution allowed	No usage restrictions	Source code freely available	Source code modification allowed	Derived work must be free again (viral aspect)	Linking with proprietary software allowed
Proprietary							
Shareware	X	X					
Freeware	X	X	X				
Public domain (USA)	X	X	X	X	X		X
GPL (GNU GPL)	X	X	X	X	X	X	
LPGL	X	X	X	X	X	X	X
MPL	X	X	X	X	X	X	X
BSD	X	X	X	X	X		X

Appendix D: Motivations of open source developers

Table 19 - Why do programmers contribute? Motivational factors.			
Factor	Explanation	Mentioned in	Motivation
Altruism		Raymond (1999) Bonaccorsi & Rossi (2003b)	Intrinsic motivation
Intrinsic utility	Comparison to scientific discovery Recognition and prestige in the community Private reputations	Bonaccorsi & Rossi (2003b) Lerner & Tirole (2002) Lerner & Tirole (2000)	
Art form	Artistic satisfaction in solving complex computing problems	Bonaccorsi & Rossi (2003b)	
Personal learning Personal enjoyment Pleasure of creativity	Rediscovering the pleasure of programming, often lost in commercial settings due to deadlines and market laws	Bonaccorsi & Rossi (2003b) Hippel & Krogh (2003)	
Signaling of quality of human capital	Making your skills known to commercial firms via an OS project in hope of e.g. a job.	Lerner & Tirole (2000) Bonaccorsi & Rossi (2003b)	Extrinsic rewards
Self production	Provide for a program that does not appear to exist to fulfill a need or solve a problem	Lerner & Tirole (2000) Bonaccorsi & Rossi (2003b)	
Financial compensation	Some commercial firms contribute programmers to a project which are on the firm's payroll.	Lerner & Tirole (2000) Lakhani, et al. (2002)	

Appendix E: Changes in the strategic value of applications

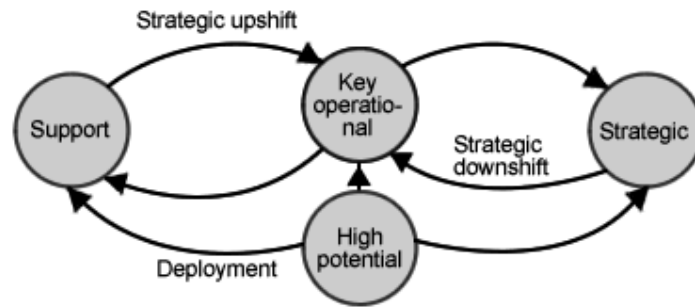


Figure 13. Shifts in strategic importance of applications in the applications portfolio

Appendix F: Sample of successful open source software

Please note that this list is not complete nor aims to be complete. It only provides examples of some successful open source software projects.

Table 20 - Sample of successful open source software			
Name	Type of software / Area	License	Link
Apache	HTTP (web) server, one of the most popular and successful open source projects.	The Apache Software License	http://www.apache.org/
ArgoUML	UML tool, used to model software requirements in the Unified Modeling Language.	BSD license	http://www.argouml.org
CVS	CVS is a version control system, essential to develop software.	GPL	http://www.gnu.org/software/cvs/
Bind	BIND is an implementation of the Domain Name System (DNS) protocols. Used to support a major part of the internet infrastructure.	BSD style license Free to use but restricted	http://www.isc.org/index.pl?sw/bind/
Emacs	Editor. The original free software project by Richard Stallmann.	GPL	http://www.gnu.org/software/emacs/
Firebird	Relational database system	IDPL (Adapted Mozilla Public License)	http://firebird.sourceforge.net/
Firefox	Browser based on Mozilla	Mozilla Public License	http://www.firefox.com/
Free BSD	FreeBSD is a UNIX operating system.	BSD type	http://www.freebsd.org/
Freetype	A free and portable TrueType font rendering engine.	BSD style license Free to use but restricted	http://www.freetype.org
GIMP	GIMP is the GNU Image Manipulation Program.	GPL	http://www.gimp.org/
Gnome	GNOME is the GNU Network Object Model Environment. This project is building a complete, user-friendly desktop.	GPL	http://www.gnome.org
Interbase	Cross-platform embedded database	Interbase Public License	http://www.borland.com/interbase/
KDE	KDE is a powerful graphical desktop environment for Unix workstations.	GPL	http://www.kde.org
Linux	Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds.	GPL	http://www.linux.org
Mambo	Content Management System	GPL	http://www.mamboserver.com/
MySQL	MySQL is a SQL database server	Dual license: Commercial and GPL	http://www.mysql.org/
Open BSD	The OpenBSD project produces a FREE, multi-platform UNIX-like operating system.	BSD type	http://www.openbsd.org/
Openoffice	An international office suite that will run on all major platforms (also commercialized by Sun: StartOffice)	Dual licensing: LGPL SISSL (Sun license)	http://www.openoffice.org/
PHP	PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.	PHP license (BSD variant)	http://www.php.net/
Perl	Perl is a high-level, general-purpose programming language.	Artistic and GPL	http://www.perl.com
PostgreSQL	PostgreSQL is a robust, next-generation, Object-Relational DBMS (ORDBMS), derived from the Berkeley Postgres database management system.	BSD Type	http://www.postgresql.org
Python	Python is an interpreted, interactive, object-oriented programming language.	BSD type	http://www.python.org

Samba	Samba is an Open Source/Free Software suite that has provided file and print services to all manner of SMB/CIFS clients.	GPL	http://www.samba.org
Sendmail	Sendmail is a Mail Transfer Agent, which is the program that moves mail from one machine to another.	OpenSource	http://www.sendmail.org/
Squid	Squid is a high-performance proxy caching server for web clients.	GPL	http://www.squid-cache.org/
Zope	Zope is a free, Open Source web application platform used for building high-performance, dynamic web sites.	OpenSource	http://www.zope.org

Appendix G : The questionnaire

Below is the questionnaire which has been used. Please note that the questionnaire posed to respondents was in **Dutch**, not in English.

Questionnaire research open source software

Thomas van der Luer - student Maastricht University



Dear Madam / Sir,

On this webpage you will find the questionnaire which has been constructed to do research in the field of open source software. You are visiting this webpage as a result of the e-mail you have received.

As posed in the e-mail, this research is taking place for my final thesis at Maastricht University. The problem statement of this thesis is:

“Which factors determine the adoption and use of open source software in commercial organisations?”

The questionnaire consists of questions related to perceptions towards open source software. **It is not of importance whether your organisation actually uses open source software.**

Your organisation was selected on the basis of a random selection. The name of your organisation will not be reported in my thesis, nor in any other publication.

Naturally all information you provide will be treated confidentially and will only serve the purpose of this research. This information will by no means be given to any third party.

If you wish, the results and the final report can be offered to you in digital format upon completion. This question will be asked to you at the end of the questionnaire.

The questionnaire will take about 9 to 13 minutes of your time.

I would like to thank you in advance for your time!

Thomas van der Luer

student International Business
Maastricht University

Part 1 // Filter and warm-up questions

- a. In what sector is your organization operating ?
- [] Telecommunications
 - [] Networkmanagement, pc security, automation services, etc.
 - [] Production of machinery
 - [] Production of beverages
 - [] Production, distribution, and trade in electricity, natural gas, and hot water
 - [] Production of optical instruments and the like.
 - [] Production of electrical components
 - [] Wholesaler in machinery, apparatus, and accessories
 - [] Construction, finishing of buildings
 - [] Other
- b. What is your responsible position ?
- [] ICT manager
 - [] Other:
- c. What is the annual gross revenue of your organization ?
- [] gross revenue < € 2 million
 - [] € 2 million < gross revenue < € 10 million
 - [] € 10 million < gross revenue < € 50 million
 - [] gross revenue > € 50 million
 - [] unknown
- d. How many people does your **entire** organization employ ?
- [] employees < 10
 - [] 10 < employees < 50
 - [] 50 < employees < 250
 - [] employees > 250
 - [] unknown

- e. Please indicate whether your organization uses open source software, and if so, which open source software it uses.

If there is no open source software in use in your organization then you can also indicate this below. If you, in no way, know what the concept of open source software stands for, please read this short introduction⁴³ first, or select the third option.

- [] Our organization **does not** use open source software, because.....
- [] Our organization **does not** use open source software and I am not familiar with this concept. Therefore, I cannot answer this question.
- [] Our organization **does** use open source software, among which are:
 - [] Linux operating system
 - [] Free / Open BSD operating system
 - [] MySQL database
 - [] PostgreSQL database
 - [] Interbase database
 - [] KDE Linux desktop environment
 - [] Gnome Linux desktop environment
 - [] Mozilla-Firefox internet browser
 - [] StarOffice/OpenOffice suite
 - [] Apache HTTP / webserver
 - [] PHP programming language
 - [] Perl programming language
 - [] Squid proxy server and web cache
 - [] Open source content management system (CMS)
 - [] Other open source software namely:

PLEASE NOTE:

If you choose option 1:

You indicated that your organization **does not use open source software** but that you **are familiar** with the concept of open source software. Therefore, I would like to ask you to complete the questionnaire and continue with part 2.

If you choose option 2:

If you indicated at the last question (question e) that your organization **does not use** open source software **and** you are **not familiar** with this concept, and therefore could not answer the question (option 2), then you don't have to answer the following questions and you can directly go to **part 8**.

⁴³ A short introduction to open source software and the concept of open source was provided.

Part 2 // Organizational characteristics

IS slack resources and budget

- a. What is, **roughly**, the budget for information communication technology (ICT) expenses **as a percentage of the revenue**?

- [] 1-3% of the revenue
- [] 4-6% of the revenue
- [] 7-10% of the revenue
- [] more than 10% of the revenue
- [] unknown

- b. The allocated ICT budget is sufficient to compensate for all expenses of the ICT department

Strongly disagree Disagree Neutral Agree Strongly agree

- c. How many people work, **roughly**, in the ICT department of your organization ?

- 0
- 1 – 2
- 3 – 9
- 10 – 25
- > 25
- unknown

- d. The number of people available to the ICT department to properly perform all tasks is sufficient

Strongly disagree Disagree Neutral Agree Strongly agree

IT in relation to the business strategy

- a. Information technology is of importance in shaping and executing current business strategies of your organization

Strongly disagree Disagree Neutral Agree Strongly agree

Part 3 // Ranking of software characteristics

- a. When you buy software, what are, in general, important factors that affect your choice of software?

Please grade **each** of the options below, where:

1 = totally not important

2 = not important

3 = neutral

4 = important

5 = very important

- [] Compatibility with existing applications and ICT infrastructure
- [] Complexity of the software
- [] Possibility to test the software
- [] The license the software is distributed with
- [] The associated, direct and indirect, switching costs
- [] The costs of the hardware the software requires to operate
- [] The availability of the source code
- [] The number of available suppliers and support organizations
- [] The safety offered by the software
- [] The reliability of the software
- [] The continuity which the software supplier offers
- [] The compliance of the software with open standards

Part 4 // Perceived open source software characteristics

Compatibility

- a. Open source software has a good fit with enterprise strategic IT architectures
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. Open source software provides application portability
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. Open source software provides data integration
- Strongly disagree Disagree Neutral Agree Strongly agree
- d. Open source software can coexist with your organization's key applications, i.e. applications that your organization is heavily dependent on to achieve its business goals.
- Strongly disagree Disagree Neutral Agree Strongly agree

Complexity

- a. Open source software is easy to use
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. It is easy for me to remember how to perform tasks using open source software
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. Using open source software requires a lot of mental effort
- Strongly disagree Disagree Neutral Agree Strongly agree
- d. Using open source software is frustrating
- Strongly disagree Disagree Neutral Agree Strongly agree

Triability

- a. You, or your organization, had a great deal of opportunity to try various open source software solutions
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. You, or your organization, knows where to go to satisfactorily try out various uses of open source software solutions

Strongly disagree Disagree Neutral Agree Strongly agree

- c. Before deciding whether to use any open source software, you or your organization was able to properly try them out.

Strongly disagree Disagree Neutral Agree Strongly agree

- d. You were able to experiment with the open source software as necessary

Strongly disagree Disagree Neutral Agree Strongly agree

Relative advantages

Cost issues

- a. Costs regarding training of users in using open source software are significant

Strongly disagree Disagree Neutral Agree Strongly agree

- b. Costs regarding installation of open source software are significant

Strongly disagree Disagree Neutral Agree Strongly agree

- c. Costs regarding integration and customization of open source are significant

Strongly disagree Disagree Neutral Agree Strongly agree

- d. Costs regarding consulting and support services of open source are significant

Strongly disagree Disagree Neutral Agree Strongly agree

- e. Open source software brings along hardware cost savings

Strongly disagree Disagree Neutral Agree Strongly agree

- f. Open source software brings along lower license costs or no license costs

Strongly disagree Disagree Neutral Agree Strongly agree

- g. If you would like to comment on the last question, with regard to licensing costs, you can do that here:.....
.....

Source code availability

- a. It is important that software source code is open and available

Strongly disagree Disagree Neutral Agree Strongly agree

- b. Your organization is too dependent on its software vendors
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. If you would like to comment on the last question, with regard to dependence on software suppliers, you can do that here:
-
- d. Open source software is capable of providing greater security
- Strongly disagree Disagree Neutral Agree Strongly agree
- e. Open source software is more reliable than proprietary software
- Strongly disagree Disagree Neutral Agree Strongly agree
- f. Open source software is more stable than proprietary software
- Strongly disagree Disagree Neutral Agree Strongly agree
- g. Closed source software suppliers provide a better long-term perspective than open source software suppliers
- Strongly disagree Disagree Neutral Agree Strongly agree

Open standards compliance

The next statement concerns **open standards**

Open standards are ICT-standards for the sake of interoperability of informationsysteminteroperability of informationsystems (that is, the ability for data exchange between ICT systems). Standards can be 'open' or 'closed'. Open standards are, among other things, royalty free and the specifications of such standards are freely available.

(bron: <http://www.ososs.nl>)

Examples of open standards are HTML, XML, HTTP, FTP, and JPEG.

Examples of closed standards are DOC, PDF, and XLS.

The statement is:

- a. Open source software embodies open standards critical to technology advancement and flexibility
- Strongly disagree Disagree Neutral Agree Strongly agree **Don't know**

Part 5 // Implementation issues

Technology championship

- a. Open source software has strong advocates in your organization
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. There are one or more people in your organization who are pushing for open source software very enthusiastically
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. Nobody in our organization has taken the lead in pushing for adoption of open source software
- Strongly disagree Disagree Neutral Agree Strongly agree
- d. There are one or more people here who are pressing for open source software usage
- Strongly disagree Disagree Neutral Agree Strongly agree

Third party support

- a. It is difficult to find companies that provide support for open source software (implementation support, training support, consultancy, etc.)
- Strongly disagree Disagree Neutral Agree Strongly agree

Top management support

- a. Your organization's top management enthusiastically supports the adoption of open source software
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. Your organization's top management has allocated enough resources for adoption of open source software
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. Your organization's top management actively encourages employees to use open source software in their daily tasks
- Strongly disagree Disagree Neutral Agree Strongly agree

Part 6 // Compatibility of work, tasks, and capabilities with OSS

- a. There is few open source software which offers the required functionality
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. The structure and flow of open source software matches well with the tasks it is supposed to support, i.e. it requires no or small procedural changes
- Strongly disagree Disagree Neutral Agree Strongly agree
- c. There is enough diversity in the kind of open source software available, i.e. for most of your organization's software needs there is an applicable and satisfying open source solution
- Strongly disagree Disagree Neutral Agree Strongly agree
- d. Open source software requires the same skills as existing software solutions within your organization.
- Strongly disagree Disagree Neutral Agree Strongly agree

Part 7 // Adoption environment: network externalities

- a. The size of the (on-line) communities of open source software used within your organization is significant
- Strongly disagree Disagree Neutral Agree Strongly agree
- b. Which of the following operating systems were operationally used on **servers** in your organization *during the last five years*?
- [] Unix
 - [] Windows
 - [] Linux
 - [] Other
 - [] Your organization does not employ any servers
- c. Could you please indicate, roughly, the percentage share of installations of the following operating systems on the total number of **servers** employed by your organization *during the last five years*
- [] % Unix
 - [] % Windows
 - [] % Linux
 - [] % Other
 - [] Not applicable
 - [] Unknown

Part 8 // Finalization

This was the last question. If you have any other remarks, then you can provide them below.

.....
.....

If you would like to receive the results of this research, please provide your e-mail address below. You will receive the opportunity to download the report when finished.

.....

Part 9 // End

The questionnaire is completed.

Your answers have been processed.

The report will be due shortly.

If you submitted your e-mail address you will receive a message when the report is finished and can be downloaded.

Thank you for your time and effort!

Appendix H: Selection of independent variables

The selection was based on the following considerations:

- The source of the constructs:
 - previous research vs. own constructs
 - multiple item constructs vs. single item constructs
- The correlation between the independent variables
- The comments provided by the respondents in case of non-adoption
- The theoretical importance following from chapter 3

Table 21 – Selection of independent variables					
Variable	Source	Not Correlated	Often mentioned in comments?	Theoretical significance	Score & Selection
Task compatibility					3
Skill compatibility					2
Compatibility					3
Complexity					1
Triability					2
Switching costs					2
Hardware costs					1
Software costs					2
Supplier independence					2
Safety					0
Reliability					0
Stability					1
Continuity					2
Open standards compliance					1
Technology championship					1
Third party support					2
Top management support					3
Shortage of IS budget and slack of IS HR					1
Slack of financial resources					1
IT in business strategy					1
Community size					1
Platform standards					1

Appendix I: Hypothesized relationship of OSS licenses

Chapter one showed that OSS is available under many licenses which all adhere to guidelines set by the Open Source Initiative. The license under which an OSS product is available can limit its potential. Any license (GPL, LGPL) which includes so-called Copyleft or viral aspects, is not very attractive to invest in for a firm, since the software has to be ported back to the open source community, as defined in the license. OSS under these kind of restrictive licenses would therefore be merely useful for end-users or only for those firms that have to make minor adaptations. It would not be attractive to heavily invest, adapt or expand such software since it would not provide any competitive advantage to the firm. *“These restrictions may therefore constrain commercialization of OSS applications”* (Stewart, Ammeter & Maruping, 2005).

OSS under such licenses as the BSD license are not restrictive. This provides significant opportunities. Software can use BSD licensed software to provide the starting point for a software package which can be further developed into a (partly or totally) proprietary solution (West, 2003). Overby, Bharadwaj & Bharadwaj (2004) refer to this aspect as *“Appropriability of benefits”*: benefits which might flow back to the public domain in case of restrictive licenses and which might transfer to the firm when considering non-restrictive licenses. For that reason it is reasoned that:

Hypothesis: Copyleft and viral licenses are negatively associated with OSS adoption

Appendix J: SPSS Output

Perceived importance of innovation characteristics

Group Statistics

	Adopted oss	N	Mean	Std. Deviation	Std. Error Mean
compatibility	0	35	4,43	,655	,111
	1	48	4,19	,734	,106
complexity	0	35	3,51	,818	,138
	1	48	3,56	,769	,111
trialability	0	35	4,00	,907	,153
	1	48	3,77	,751	,108
license agreement	0	35	3,49	1,173	,198
	1	48	3,44	,897	,129
switching costs	0	35	4,11	,718	,121
	1	48	3,92	,767	,111
hardware costs	0	35	3,94	,684	,116
	1	48	3,79	,874	,126
source code availability	0	35	2,34	1,056	,178
	1	48	2,77	,973	,140
suppliers and support	0	35	3,69	,993	,168
	1	48	3,33	,907	,131
safety	0	35	4,37	,547	,092
	1	48	4,17	,663	,096
reliability	0	35	4,66	,482	,081
	1	48	4,48	,505	,073
continuity	0	35	4,63	,490	,083
	1	48	4,15	,652	,094

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means						
		F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
									Lower	Upper
	Equal variances assumed	,020	,889	1,546	81	,126	,241	,156	-,069	,,561
	Equal variances not assumed			1,574	77,676	,120	,241	,153	-,064	,,561
	Equal variances assumed	,258	,613	-,275	81	,784	-,048	,176	-,398	,,302
	Equal variances not assumed			-,272	70,723	,786	-,048	,177	-,402	,,302
	Equal variances assumed	,192	,662	1,257	81	,212	,229	,182	-,134	,,436
	Equal variances not assumed			1,220	64,729	,227	,229	,188	-,146	,,436
nt	Equal variances assumed	2,752	,101	,212	81	,832	,048	,227	-,404	,,308
	Equal variances not assumed			,204	61,152	,839	,048	,237	-,425	,,308
	Equal variances assumed	,018	,893	1,190	81	,237	,198	,166	-,133	,,437
	Equal variances not assumed			1,203	76,034	,233	,198	,164	-,130	,,437
	Equal variances assumed	3,878	,052	,851	81	,398	,151	,178	-,202	,,437
	Equal variances not assumed			,884	80,564	,379	,151	,171	-,189	,,437
ibility	Equal variances assumed	,696	,407	-1,909	81	,060	-,428	,224	-,874	,,022
	Equal variances not assumed			-1,885	69,789	,064	-,428	,227	-,881	,,022
pport	Equal variances assumed	,023	,879	1,679	81	,097	,352	,210	-,065	,,637
	Equal variances not assumed			1,655	69,366	,102	,352	,213	-,072	,,637
	Equal variances assumed	,043	,836	1,493	81	,139	,205	,137	-,068	,,637
	Equal variances not assumed			1,539	79,704	,128	,205	,133	-,060	,,637
	Equal variances assumed	4,826	,031	1,617	81	,110	,178	,110	-,041	,,637
	Equal variances not assumed			1,629	75,330	,107	,178	,109	-,040	,,637
	Equal variances assumed	,003	,957	3,684	81	,000	,483	,131	,222	,,752
	Equal variances not assumed			3,850	80,904	,000	,483	,125	,233	,,752

Logistic regression on original 8 selected variables

Block 0: Beginning Block

Iteration History(a,b,c)

Iteration		-2 Log likelihood	Coefficients
			Constant
Step 0	1	113,018	,313
	2	113,018	,316
	3	113,018	,316

a Constant is included in the model.

b Initial -2 Log Likelihood: 113,018

c Estimation terminated at iteration number 3 because parameter estimates changed by less than ,001.

Classification Table(a,b)

Observed		Predicted		
		Adopted OSS		Percentage Correct
		0	1	
Step 0	Adopted OSS	0	1	
		0	35	,0
		0	48	100,0
	Overall Percentage			57,8

a Constant is included in the model.

b The cut value is ,570

Block 1: Method = Enter

Omnibus Tests of Model Coefficients

		Chi-square	df	Sig.
Step 1	Step	53,617	8	,000
	Block	53,617	8	,000
	Model	53,617	8	,000

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	59,401(a)	,476	,640

a Estimation terminated at iteration number 7 because parameter estimates changed by less than ,001.

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	9,976	8	,267

Alternative Logistic regression 1: Enter technology championship

Iteration History(a,b,c,d)

It.	-2LL	Coefficients								
		Constant	Triability	Software costs	Continuity	Third party support	Task comp.	Skill comp.	Technology champion.	Comp.
1	71,201	-6,918	,685	,181	-,132	,087	,456	-,063	,199	,844
2	61,361	-11,674	,999	,344	-,278	,131	,743	-,064	,367	1,576
3	58,330	-15,345	1,206	,436	-,506	,228	,960	-,107	,514	2,323
4	57,831	-17,337	1,320	,469	-,668	,315	1,068	-,157	,600	2,792
5	57,811	-17,819	1,349	,476	-,709	,338	1,093	-,170	,621	2,909
6	57,811	-17,841	1,350	,476	-,710	,338	1,094	-,171	,622	2,915
7	57,811	-17,841	1,350	,476	-,710	,338	1,094	-,171	,622	2,915

a Method: Enter

b Initial -2 Log Likelihood: 113,018

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	57,811(a)	,486	,653

a Estimation terminated at iteration number 7 because parameter estimates changed by less than ,001.

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	10,305	8	,244

Classification Table(a)

Observed		Predicted			
		Adopted OSS		Percentage Correct	
		0	1		
Step 1	Adopted OSS	0	33	2	94,3
		1	10	38	79,2
	Overall Percentage				85,5

a The cut value is ,570

Variables in the Equation

		B	S.E.	Wald	df	Sig.	Exp(B)
Step 1(a)	Triability	1,350	,677	3,977	1	,046	3,857
	Software costs	,476	,574	,690	1	,406	1,610
	Continuity	-,710	,614	1,338	1	,247	,491
	3rd party support	,338	,569	,354	1	,552	1,403
	Task comp.	1,094	,674	2,633	1	,105	2,985
	Skill comp.	-,171	,446	,147	1	,702	,843
	Techn. champ.	,622	,511	1,479	1	,224	1,862
	Compatibility	2,915	1,086	7,210	1	,007	18,445
	Constant	-17,841	4,629	14,854	1	,000	,000

a All variables entered at step 1.

Alternative Logistic regression 2: Enter software quality

Iteration History(a,c)

It.	-2LL	Coefficients								
		Constant	Triability	Software costs	Continuity	Third party support	Task comp.	Skill comp.	Comp.	Softw. quality
1	71,734	-6,871	,778	,176	-,138	,025	,554	-,086	,983	-,069
2	62,349	-11,704	1,130	,332	-,264	,027	,835	-,089	1,745	,089
3	59,479	-15,584	1,343	,438	-,458	,111	1,018	-,121	2,515	,257
4	59,053	-17,617	1,446	,484	-,575	,190	1,107	-,162	2,977	,324
5	59,040	-18,045	1,467	,493	-,599	,209	1,126	-,174	3,080	,334
6	59,040	-18,061	1,468	,493	-,600	,210	1,127	-,174	3,084	,334
7	59,040	-18,061	1,468	,493	-,600	,210	1,127	-,174	3,084	,334

a Method: Enter

c Initial -2 Log Likelihood: 113,018

Model Summary

Step	-2 Log likelihood	Cox & Snell R Square	Nagelkerke R Square
1	59,040(a)	,478	,643

a Estimation terminated at iteration number 7 because parameter estimates changed by less than ,001.

Hosmer and Lemeshow Test

Step	Chi-square	df	Sig.
1	7,227	8	,512

Classification Table(a)

Observed		Predicted		Percentage Correct
		Adopted OSS		
		0	1	
Step 1	Adopted OSS	0		
		33	2	94,3
		9	39	81,3
	Overall Percentage			86,7

a The cut value is ,570

Variables in the Equation

	B	S.E.	Wald	df	Sig.	Exp(B)
Step 1(a)						
Triability	1,468	,659	4,959	1	,026	4,339
Software costs	,493	,566	,759	1	,384	1,638
Continuity	-,600	,558	1,155	1	,282	,549
Third party support	,210	,561	,140	1	,708	1,234
Task comp.	1,127	,674	2,800	1	,094	3,086
Skill comp.	-,174	,444	,154	1	,695	,840
Compatibility	3,084	1,062	8,431	1	,004	21,853
Software quality	,334	,559	,356	1	,551	1,396
Constant	-18,061	4,783	14,257	1	,000	,000

a All variables entered at step

