Thesis Edwin Groffen 265639

# Abstract

Table of contents

[Abstract 1](#_Toc238725775)

[1. Introduction 3](#_Toc238725776)

[1.1 Motivation 3](#_Toc238725777)

[1.2 Problem Field 3](#_Toc238725778)

[1.2.1 Free/Open Source Software 3](#_Toc238725779)

[1.2.2 Innovation 4](#_Toc238725780)

[1.3 Research Focus 4](#_Toc238725781)

[1.3.1 Problem definition 5](#_Toc238725782)

[1.4 Research Methodology 5](#_Toc238725783)

[Thesis structure 5](#_Toc238725784)

[2. Free and Open Source Software 7](#_Toc238725785)

[2.1 The Free / Open Source phenomenon 7](#_Toc238725786)

[2.1.1 FOSS developer motivation 7](#_Toc238725787)

[2.1.2 Business models, Involvement of commercial companies 8](#_Toc238725788)

[2.2 FOSS Projects 9](#_Toc238725789)

[2.2.1 Roles of Project Members 9](#_Toc238725790)

[2.2.2 Project/Community Structure 10](#_Toc238725791)

[2.2.3 Project lifecycle 11](#_Toc238725792)

[2.2.4 Project Licensing 12](#_Toc238725793)

[3. Radical and Incremental innovation 14](#_Toc238725794)

[3.1 Introduction 14](#_Toc238725795)

[3.2 Importance of Radical and Incremental innovation projects 15](#_Toc238725796)

[3.2.1 Importance of Incremental innovation projects 15](#_Toc238725797)

[3.2.2 Importance of Radical innovation projects 15](#_Toc238725798)

[3.2.3 Portfolio management 16](#_Toc238725799)

[3.3 Management of innovation 17](#_Toc238725800)

[3.3.1 Management of Incremental innovation 17](#_Toc238725801)

[3.3.2 Management of Radical innovation 18](#_Toc238725802)

[3.3.3 Managing project risk 20](#_Toc238725803)

[3.4 Innovation in Software Development 21](#_Toc238725804)

[3.4.1 Previous research on innovation in FOSS 22](#_Toc238725805)

[3.4.2 Measurement of radicalness in FOSS 23](#_Toc238725806)

[3.4.3 Success in FOSS projects 24](#_Toc238725807)

[3.4.4 Hypotheses 24](#_Toc238725808)

[3.3.4 Conclusion 24](#_Toc238725809)

[4. Data 26](#_Toc238725810)

[4.1 Data 26](#_Toc238725811)

[4.2 Methodology 26](#_Toc238725812)

[4.3 Industry Experts 27](#_Toc238725813)

[5. Empirical Results 29](#_Toc238725814)

[5.1 FOSS development: Incremental or Radical? 29](#_Toc238725815)

[5.2 Software development Categories 31](#_Toc238725816)

[5.3 Licenses 33](#_Toc238725817)

[5.4 Developers 34](#_Toc238725818)

[6. Summary and Conclusion 37](#_Toc238725819)

[Problem definition 37](#_Toc238725820)

[Summary 37](#_Toc238725821)

[Results 37](#_Toc238725822)

[Pratical implication 39](#_Toc238725823)

[Reflection and recommendation 39](#_Toc238725824)

[7. Appendix 40](#_Toc238725825)

[Appendix 1 Cooper’s Stage Gate Model 40](#_Toc238725826)

[Appendix 2 Overview differences radical en incremental innovation 41](#_Toc238725827)

[Apendix 3 Newness to market, technology newness and impact versus success. 42](#_Toc238725828)

[8. References 43](#_Toc238725829)

# Introduction

## 1.1 Motivation

The subject I am going to cover is the phenomenon of Open Source, or to be more precise Free and Open Source Software (FOSS) development. In software development there are basically two different forms or organizations: The Commercial or Closed Source Software (CSS) developers who are organized in a company aiming at profit (like Microsoft), and the Free and Open Source Software developers who operate voluntarily in an online community. The first time I got in contact with Open Source development was during the bachelor thesis ‘E-organizations in the digital economy’, by writing a research paper on online communities. I found it fascinating that geographically distributed individuals who work voluntarily on software development can create significant results. Yet little or no research has been done to what kind of innovation takes place in those online communities. Is FOSS limited to small, incremental improvements or is FOSS able to generate big, successful radical new products? These questions formed the basis for writing this Thesis.

## 1.2 Problem Field

This section describes the problem fields that are addressed in the thesis which are *the Free/Open Source Software platform* and the field of *Innovation*.

### 1.2.1 Free/Open Source Software

Over the past 10 years, open source software has become an important cornerstone of the software industry (Riehle et al, 2009). The development method in FOSS development is different from the commercial style of development (Mockus et al, 2002). The Open Source Initiative[[1]](#footnote-2) describes Open source as a ‘development method for software that harnesses the power of distributed peer review and transparency of process’. This definition suggests a geographically distributed development in which everyone can join and where developers undertake work voluntarily. A hot topic in the FOSS literature has been the question why developers contribute to FOSS projects (e.g. Steward and Gosain, 2001; Lerner and Tirole, 2002; Crowston and Scozzi, 2002; Franke and Von Hippel, 2003). Intrinsic motivations and signaling incentives are considered important in FOSS developer contribution. Innovation in FOSS is a field that has not attracted that much attention in the literature. Results on this topic are rather scarce and controversial (Lorezi and Rossi, 2008).

The interest of commercial companies entering the Open Source market is growing (Ägerflak and Fitzgerald, 2008). This has led to a movement operating under a new name. No longer do they prefer to be part of the name ‘Open Source Software’, but they call their products ‘Free Software’. The best known group is the *Free Software Foundation[[2]](#footnote-3)*. In essence the same structure and approach is used as in the development of Open Source projects, but they apply a more pure ideology, free of any commercial intentions. The difference lies mainly in the use of licenses. OSS sometimes adds or removes freedoms or copyright privileges to end-users. Simply put: free software is always available as OSS, but OSS is not always free software. In this research both Free and Open Source projects are studied, which in short I will name FOSS projects.

### 1.2.2 Innovation

A way to capture innovativeness is to differentiate between radical and incremental innovation. In short, radical innovations can be described as major innovations that represent revolutionary changes in technology (Ettlie et al, 1987). Where incremental innovations can be described as minor improvements or simple adjustments in current technology (Munson and Pelz, 1979). A lot of research has been done on the topic of management of radical and incremental innovation projects (e.g. Cooper, 1991; Cooper and Kleinschmidt, 1995, 2002; Leifer et al, 2000). Especially for the management of radical innovation projects it is difficult to recognize general applicable success factors (Leifer et al, 2000).

## 1.3 Research Focus

The literature on innovation in Free/Open source software development is scant and controversial (Lorenzi and Rossi, 2008). Not only are there different theoretical opinions on what type of innovation suits the FOSS platform best, also the empirical research (Klincewicz, 2005; Rossi, 2009) has opposite conclusions. What I want to do in this research is to recognize what type of innovation, either more incremental or more radical innovation, is successful in FOSS development. A strong conclusion would contribute to the FOSS literature in a way that it could help make the evidence on innovation in FOSS development less controversial. The practical relevance of this research lies mainly at commercial software companies who would like to get involved in FOSS development. For those companies it is interesting to know what kind of innovation projects they can successfully undertake in the FOSS sector.

The two main concepts that are central in this thesis are the *Software Platform* and *Innovation*. I want to research if there is a relation between the type of software platform and a type of innovation.

Innovation

Software platform

Figure 1.1 Research concepts

The software platform can be divided into the Free/Open Source Software platform and the Commercial platform. The main focus of the thesis lies on the FOSS platform. This has several reasons. The FOSS platform has an open community structure where information on projects is easily accessible. Project sites like sourceforge.net provide an extensive database that is very valuable for empirical research. Project data on proprietary software is much more difficult to acquire. There is no online database available with information on commercial software projects.

The concept of innovation can be separated into two types of innovation: Incremental and Radical.

Incremental Innovation

Radical Innovation

Free/Open Source Software platform

Figure 1.2 Relation between the research concepts

Figure 1.2 illustrates the relation between the different concepts of this thesis. Characteristics of the FOSS platform have a certain effect on the development success of Incremental/Radical innovation projects. The question is what kind of relation there is between the concepts. It could be, for example, that there is a positive relation between FOSS development and incremental innovation and a negative relation between FOSS development and radical innovation. That would indicate that the FOSS platform is better suitable for incremental innovation projects.

### 1.3.1 Problem definition

The relations and issues as described leads to the central problem definition of this thesis:

**What kind of innovation suits the FOSS platform best?**

The central problem definition can be separated into different sub-questions. The answers to the sub-questions form the basis for dealing with the central problem definition.

1) What kind of innovation takes place in FOSS development projects?

2) Which type of innovation projects are most successful in FOSS development?

3) Is there a relation between the type of innovation and the success of projects in FOSS development?

## 1.4 Research Methodology

In the literature review section I want to get better insight into the concepts of FOSS development and Incremental and Radical innovation. In the section on FOSS development I will mainly focus on the characteristics of a typical FOSS project. I will elaborate on how these projects are structured and operate in general. Understanding how a FOSS project works is useful in order to link it with literature on project innovation management. A section is devoted to strategies on how commercial companies can get involved in FOSS development.

From the innovation literature I am mostly interested how different innovation projects should be managed. What are the differences between managing a radical project and an incremental project? What is the value of the innovation literature for FOSS project management? Furthermore previous literature on innovation in FOSS will be discussed in order to formulate an hypothesis on the issue which type of innovation fits the FOSS platform best.

To answer the hypothesis an empirical analysis will be performed. The sourceforge.net database forms the basis for FOSS projects information. A list of projects will be created which a panel of industry experts will rate on radicalness and success. Finally, on the hand of the results, the sub-questions and the central problem definition can be answered.

### 1.4.1 Thesis structure

The thesis continues with a section on FOSS development, followed by a literature review on radical and incremental innovation. In chapter 4 is presented how the data for the empirical research is retrieved. Chapter 5 presents the empirical results. With the help of graphic illustrations, relations and patterns between project innovation type and project success become visible. The final chapter summarizes the research, discusses the results and adds an answer on the central problem definition.

# Free and Open Source Software

## 2.1 The Free/Open Source phenomenon

Eric Raymond was the first to describe the Free/Open Source community and its method of writing software in his book “The Cathedral & the Bazaar” (Raymond, 1999). The Cathedral stands for proprietary software production as the carefully planned building of a cathedral, where Free/Open Source software production can be interpreted as the chaotic interactions of the participants in an oriental bazaar.

Free/Open Source is often characterized as a fundamentally new way to develop software (Raymond 1999). It could pose a serious challenge to the commercial software businesses that dominate most software markets today (Vixie, 1999). Open Source cannot be seen as just a new competitor that operates according to the same rules as the commercial business. It threatens to do it faster, better and cheaper. The OSS challenge is often described as much more fundamental, and goes to the basic motivations, economics, market structure, and philosophy of the institutions that develop, market, and use software (Vixie, 1999).

The development process that originates from a freely available source code is radically different from the industrial or commercial style of development. Mockus et al (2002) named the main differences between FOSS and commercial development. For once, OSS systems are built by potentially large numbers of *volunteers*. The work is not assigned to people, but OSS developers undertake the work they choose to undertake. Also there is no explicit system-level or even detailed design, no project plan, schedule, or list of deliverables.

FOSS is characterized by its geographical distributed development. The developers work in arbitrary locations, rarely or never meet face to face, and coordinate their activity almost exclusively by means of email and bulletin boards (Mockus et al, 2002). The difference with for example ‘the virtual organization’ as discussed by Chesbrough and Teece (2003) is that open source movements consist of volunteers, is open for everybody to join and do not aim at developing a commercially interesting product or making profit.

### 2.1.1 FOSS developer motivation

A commonly asked question about FOSS projects is why software developers voluntarily join and participate in FOSS development (Scacci, 2007). In their article ‘Some simple economics of open source’, Lerner and Tirole (2002) defined an important incentive for members operating in an open source software community: the signalling incentive. They separate this signalling incentive in two different incentive’s, *the career concern incentive* and *the ego gratification incentive*. The career concern incentive refers to future job offers, shares in commercial open source-based companies, or future access to the venture capital market. The ego gratification incentive involves the desire from a developer for peer recognition. Other researchers mentioned the gratification incentive as “building trust and reputation” (Stewart and Gosain, 2001) or achieving “geek fame” (Pavelicek, 2000).

More intrinsic motivations are recognized by Crowston and Scozzi (2002). They argued that developers sometimes simply see their effort as something that is “fun, personally rewarding or provides a venue where they can exercise and improve their technical competence in a manner that may not be possible within their current job”. Franke and Von Hippel (2003) contribute that: “People innovate to better satisfy their own needs”. Since users are developers, they are involved in FOSS development to create or enhance software (functions) that they would like to use themselves.

### 2.1.2 Business models, Involvement of commercial companies

Over the past 10 years, open source software has become an important cornerstone of the software industry. Commercial users have adopted it in standalone applications, and software vendors are embedding it in products (Riehle et al, 2009). Ägerflak and Fitzgerald (2008) introduce the term *opensourcing* which is what they call “the use of the OSS development model as a global sourcing strategy for an organization’s software development process”. Commercial companies and open source communities collaborate on development of software of commercial interest to the company. Ägerflak and Fitzgerald explored critical customer and community obligations that contribute to success in an opensourcing relationship. They found that terms like openness, trust, tact, professionalism, transparency and complementariness are key for a successful partnership.

 In a business model review on open source Watson et al. (2008) distinguish five models of software production or distribution. Next to the Proprietary model and the Open Communities, they name three models that are hybrids between the two extreme cases. Lerner and Tirole (2002) also identified three strategies to capitalize on the OS movement.

Firms like RedHat[[3]](#footnote-4) have emerged to create value and generate revenue by “identifying best-of-breed OSS projects, improving distribution methods for these products, and providing complementary services in order to make these OSS products more accessible” (Watson et al, 2008). Watson et al call this the *Corporate Distribution* model. Lerner and Tirole (2002) described a similar strategy for firms in which they live symbiotically off an open source project. Firms commercially provide complementary services and products that are not supplied efficiently by the FOSS community.

The second hybrid by Watson et al (2008) is the *Sponsored Open Source*  model, where corporations and foundations sponsor some OSS projects. Some sponsors do this by directly contributing development resources to OSS projects. IBM[[4]](#footnote-5) for example, contributed developers to Apache’s Web server[[5]](#footnote-6). Or corporations release previously closed source code and encourage their employees to work on the now open project. Lerner and Tirole (2002) add that code release can be advantageous to boost a firms profit on a complementary segment or is preferable when the company is too small to commercially compete.

The third hybrid is what Watson et al (2008) call the *Second-Generation Open Source* (OSSg2) model. This is essentially a hybrid between corporate distribution and sponsored OSS. A vast majority of their revenues comes from complementary services around their products and they provide the majority of the resources to create and maintain their products. The difference with corporate distribution is that OSSg2 firms generally do not sell licenses for their product. It differs from sponsored projects in a way that it typically keeps tight control over the software code and can therefore better exploit their intimate knowledge. Lerner and Tirole’s (2002) description of *Intermediaries*, organizations that act like venture capitalists who organize OS projects for corporations who wish to develop part of their software in this manner, comes close to the description OSSg2 model.

## 2.2 FOSS Projects

The FOSS community is hard to investigate as an abstract social phenomenon. It is difficult to determine who is a part of it and who is not. Fortunately, projects can be observed and analyzed due to their presence on the Internet and their publicly available communication. There a many ways in which a FOSS project can be ran, but there are some common elements. The first question is: What is a FOSS project?

Definition

*Any group of people (or sole individuals) developing software and providing their results to the public under an Open Source license constitute an Open Source project .[[6]](#footnote-7)*

The major productive assets of FOSS projects are developers. Developer is a wide term, and need not be confined to programmers, but can also include documentation writers, graphic artists and others.

### 2.2.1 Roles of Project Members

To get a better understanding of innovation in FOSS projects I will discuss how , in general, a project is structured and what the roles are of the project members. One distinct feature of a FOSS project, as compared to the commercial software development, is that members of the FOSS project assume certain roles by themselves according to their personal interest in the project, rather than being assigned a task by someone else. A member may have one of the following eight roles (derived from the work of Nakakoji et al., 2002) (see also Figure 2.1).

**Passive User.** Passive Users just use the system in the same way as most of us use commercial software; they are attracted to OSS mainly due to its high quality and the potential of being changed when needed.

**Reader.** Readers are active users of the system; they not only use the system, but also try to understand how the system works by reading the source code. Readers are like peer reviewers in traditional software development organizations.

**Bug Reporter.** Bug Reporters discover and report bugs; they do not fix the bugs themselves, and they may not read source code either. They assume the same role as testers of the traditional software development model.

**Bug Fixer.** Bug Fixers fix the bug that is either discovered by themselves or reported by Bug reporters. Bug Fixers have to read and understand a small portion of the source code of the system where the bug occurs.

**Peripheral Developer.** Peripheral Developers contribute occasionally new functionality or features to the existing system. Their contribution is irregular, and the period of involvement is short and sporadic.

**Active Developer.** Active Developers regularly contribute new features and fix bugs; they are one of the major development forces of OSS systems.

**Core Member.** Core Members are responsible for guiding and coordinating the development of an OSS project. Core Members are those people who have been involved with the project for a relative long time and have made significant contributions to the development and evolution of the system.

 **Project Leader.** Project Leader is often the person who has initiated the project. He or she is responsible for the vision and overall direction of the project.

### 2.2.2 Project/Community Structure

Although a strict hierarchical structure does not exist in FOSS communities, the structure of FOSS communities is not completely flat (O’Reilly, 2002). The influences that members have on the system and the community are different depending on what role they play. Figure 2.1 depicts the general layered structure of FOSS communities, where the role closer to the center has a larger influence (Nakakoji et al., 2002).



Figure 2.1 FOSS project structure Source: Nakakoji et al. (2002) p. 5

The roles and their associated influences in FOSS communities have to be earned through contributions to the community. Attributes like age and title are irrelevant. The roles are not fixed as each member can play a larger role if they aspire. It is important to maintain a balanced composition of all the different roles in a community, otherwise an FOSS community is not sustainable (Mockus et al. 2000). Each FOSS community has a unique structure depending on the nature of the system and its member population. The structure of an FOSS community differs at the percentage of each role in the whole community. Generally speaking, most members are Passive Users. For example, about 99% of people who use Apache are Passive Users. The percentage drops sharply from Readers to Core Members. But most open source software is contributed only by a small number of developers (Mockus et al., 2000; O’Reilly, 2002).

FOSS project teams can take the form of a "pyramid meritocracy" (Fielding 1999; Kim, 2000) operating as an Internet-based virtual enterprise (Crowston and Scozzi, 2002). A pyramid meritocracy is a hierarchical organizational form that centralizes and concentrates certain kinds of elite authority, trust, and respect for experience and accomplishment within the community or team (Crowston and Howison, 2006). Capra et al. (2008) also recognized the meritocratic hierarchy. Hey added a term called the ‘do-ocracy’, which means that decisions are made by the developers who more actively contribute to the project.

### 2.2.3 Project lifecycle

FOSS projects are organic. They do not follow strict patterns for releases. A common classification of the various stages of a FOSS Project used by large FOSS sites is Planning, Pre-Alpha, Alpha, Beta, Stable, Mature[[7]](#footnote-8).

**Planning.** No code has been written, the scope of the project is still in flux. The project is but an idea. As soon as tangible results in the form of source code appear, the project enters the next stage.

**Pre-Alpha.** Very preliminary source code has been released. The code is not expected to compile, or even run. Outside observers may have a hard time to figure out the meaning of the source code. As soon as a coherent intent is visible in the code that indicates the eventual direction, the project enters the next stage.

**Alpha.** The released code works at least some of the time, and begins to take shape. Preliminary development notes may show up. Active work to expand the feature set of the application continues. As the amount of new features slows down, the project enters the next stage.

**Beta.** The code is feature-complete, but retains faults. These are gradually weeded out, leading to software that is ever more reliable. If the number of faults is deemed low enough, the project releases a stable version, and enters the next stage.

**Stable.** The software is useful and reliable enough for daily use. Changes are applied very carefully, and the intent of changes is to increase stability, not new functionality. If no significant changes happen over a long time, and only minor issues remain, the project enters the next stage.

**Mature.** There is little or no new development occurring, as the software fulfills its purpose very reliably. Changes are applied with extreme caution, if at all. A project may remain in this final stage for many years before it slowly fades into the background because it has become obsolete, or replaced by better software. The source code for mature projects remains available indefinitely, however, and may serve educational purposes.

Figure 2.2 Overview of SF.net project statuses

There are over 180.000 projects registered at sourceforge.net[[8]](#footnote-9). Most projects are in the planning stage. This can be explained by the ease of setting up a project. A new project can be set up in minutes. Only 18% of the projects has made it past the point of beta.

### 2.2.4 Project Licensing

Copyrights and software licenses provide the legal framework under which FOSS remains in the public domain (McGhee, 2007). Fitzgerald calls licensing in FOSS ironic: “given the perceptions that FOSS is collectivist and anti-intellectual property, the success of the open source model is due largely to the use of licensing, albeit in a form that counters the normal restrictive sense” (Fitzgerald, 2006, p. 7). Property rights are vested in the author through copyright, with liberal rights granted to others under license. The most used licenses in FOSS are the GNU[[9]](#footnote-10) Public License (GPL), the Lesser GPL (LGPL), the Artistic License, Berkeley System Distribution (BSD) and the commercially oriented Mozilla Public License (MPL). The GPL covers about 70% of the FOSS projects (Wheeler, 2007).

The GPL was created in the mid 1980s to distribute the GNU project software. Most open source software to date has been distributed under the GPL, Linux being one high-profile example (Fitzgerald, 2006). The GPL subverts the traditional concept of restricted access through copyright by ensuring complete, unrestricted access to all open source software and any derivatives. These must also be licensed under the same terms, referred to as ‘copyleft – all rights reversed’. This latter guarantee of the same rights to subsequent users caused such licenses to be termed ‘viral.’

Fitzgerald (2006) calls the GPL controversial, because it requires that all applications that contain GPL software are also released under a GPL license. This proved to be impractical in some situations and therefore the Lesser GPL (LGPL) was created. The main difference between the LGPL and the GPL, is that the LGPL does not prohibit linking the software with proprietary code.

The BSD license is another early widespread license. It has fewer restrictions than the GPL license as the BSD model offers free code distributions and allows covering derivative works under different terms. One requirement is that always the necessary credit should be given. The creation of the MPL by Netscape was significant, because it focused on the conversion of a commercial software product to an Open source project (Fitzgerald, 2006).

# Radical and Incremental innovation

## 3.1 Introduction

In order to define what a radical and an incremental innovation is, it is useful to first define what a ‘technological innovation’ is. Different backgrounds, like marketing, management, economics and engineering all have their own version of what is considered an innovation (Garcia and Calantone, 2002). A study performed by the OECD in 1991 on technological innovations captures the essence of innovations from an overall perspective: “Innovation is an iterative process initiated by the perception of a new market and/or new service opportunity for a technology based invention which leads to development, production, and marketing tasks striving for the commercial success of the invention” (OECD, 1991).

According to Garcia and Calantone (2002), the OECD definition points out two rather important distinctions:

1) the ‘innovation’ process comprises the technological development of an invention *combined* with the market introduction of that invention to end-users through adoption and diffusion.

2) “the innovation process is *iterative* in nature”. This means that the first introduction of a new innovation is followed by a reintroduction of an improved innovation. Garcia and Calantone state that “this iterative process implies varying degrees of innovativeness and thus, necessitates a typology to describe different types of innovations”.

‘Innovativeness’ is most frequently used as a measure of the degree of ‘newness’ of an innovation (Garcia and Calantone, 2002). A common way to capture and label innovativeness is to differentiate between radical and incremental innovations (e.g. Dosi, 1982). In the past the radical innovation has also been described as a discontinuous innovation (e.g. Tushman and Anderson, 1990), an architectural innovation (Henderson and Clark, 1990) or an emerging technology (e.g. Day and Schoemaker, 2000). Essentially the definitions all have common elements. Aspects that are related are: a high market and technology uncertainty, new market creation, current product cannibalization and even effects on the current knowledge base of the company (Garcia and Calantone, 2002). For the remainder of my research I will use the definitions of radical and incremental innovation.

Newness to Market

|  |  |  |
| --- | --- | --- |
|  | High | Low |
| HighTechnologicalNewness | Radicalinnovation |  |
| Low |  | Incrementalinnovation |

Figure 3.1 Innovation on Technological and Market newness

Radical innovations are fundamental changes that represent revolutionary changes in technology. They represent clear departures from existing practice (Ettlie, 1983). Leifer at al identified radical innovation as: “a product, process, or service with either unprecedented performance features or familiar features that offer potential for significant improvements in performance or cost” (Leifer et al., 2001, p.5)

In contrast, incremental innovations are minor improvements or simple adjustments in current technology (Munson and Pelz, 1979). Utterback (1996) adds that incremental innovations give way to standardization and status quo within the firm or industry.

The notion of *radicalness* is a way to capture the degree of discontinuity in marketing and technological factors.

ADD

Little continuity exists in the new product literature regarding from *whose* perspective this degree of newness is viewed and *what* is new. Although the majority of research takes a firm’s perspective toward newness, others look at new to the world (Song and Montoya Weiss, 1998), (Ettlie and Rubenstein, 1987), new to the adopting unit (Ettlie and Rubenstein, 1987), new to the industry (O’Connor, 1998), new to the market (Kleinschmidt and Cooper, 1991), and new to the consumer (Atuahene-Gima, 1995).

## 3.2 Importance of Radical and Incremental innovation projects

In this section I will focus on the main benefits of respectively incremental and radical innovation projects. What is the importance of each of the two innovation types for companies?

### 3.2.1 Importance of Incremental innovation projects

As described, incremental innovations are minor improvements in current technology that give way to standardization within the firm. Incremental innovations require lower investments than radical innovations and incremental innovations have a bigger chance of making it to commercialization (Cooper, 2003). Also, less innovative products are more familiar, less uncertain, may have higher synergies, and therefore have a higher success rate (Kleinschidt and Cooper, 1991).So the risks involved in incremental innovation are smaller and better manageable than its radical equivalent.

In their 1995 research, Banbury and Mitchell came to the conclusion that introducing incremental innovations (although they might cannibalize the current product line) are crucial for gaining market share. Their research applied mostly to industries in which there are moderate switching costs for buyers. Fisher (2007) states that incremental innovations are important to a company’s competitive strategy and long-term outlook. Incremental innovations can render a ‘steady’ revenue, which can be used to finance other incremental but also more radical innovation projects.

### 3.2.2 Importance of Radical innovation projects

Leaders of established companies acknowledge that radical innovation is critical to their long-term growth and renewal (Leifer et al, 2002). More innovative products should create more opportunities for differentiation and competitive advantage, hence impact positively on performance (Kleinschmidt and Cooper, 1991).

According to Song and Montoya-Weiss (1998) “really new products can often create significant opportunities for differentiation and competitive advantage”. However, the investments that are necessary for radical innovation projects are much higher (Kleinschmidt and Cooper, 1991) and the risks of projects not making it to commercialization are higher than incremental projects (Cooper, 2003). So the risk involved in radical innovation is much higher, but the potential rewards, in term of new product success, are also higher. This is supported by Watson (2006). He found that on average 14% of innovation projects are radical, but take account for 60% of a company’s current profits. The other 86% are low risk and take account for about 30% of the profits. So despite high risks, it pays to invest in radical innovation projects.

A special kind of radical innovation was introduced by Christensen and Bower in 1996. The term disruptive technology, which Christensen later renamed to disruptive innovation, is an innovation that typically initially underperforms the current technology in mainstream markets, but can (due to a rapid rate of technological progress) outperform current technologies in the future.

Christensen and Bower (1996) argue that leading firms focus too much on existing powerful customers. No resources are allocated to technologies which initially can only be used in emerging markets, but have potential to invade mainstream markets. Incumbents can lose significant shares of their market to firms and their disruptive technologies.

To conclude this section I would like to point out to a study performed by Kleinschmidt and Cooper (1991) on the impact of product innovativeness on firm performance. They found a U-shape (see figure 3.2) in which both low and highly innovative innovations outperform the moderate innovations on several points like return on Investment, success rate and impact on market share.

****Figure 3.2 Impact of innovativeness on profitability. Kleinschmidt and Cooper, 1991, p. 245.

### 3.2.3 Portfolio management

With decent portfolio management, companies are able to have a good mix of incremental and radical projects. Desired is a balanced portfolio, defined as an assortment of projects that enables a company to achieve the growth and profit objectives associated with its corporate strategy without exposing the company to undue risks (Hill and Jones, 1992).Portfolio management has three goals according to Cooper et al (1997): 1) Maximizing the value of the portfolio, 2) Balance in the portfolio and 3) Link to strategy.

## 3.3 Management of innovation

McDermott and O’Connor (2002) state that much less is known about effective management of the product development process of radical projects when compared to incremental development projects. Ernst (2002) acknowledged this in his review of the empirical literature on success factors of new product development (NPD). His work summarizes the most important findings in empirical NPD research in a structured way.

### 3.3.1 Management of Incremental innovation

I will mainly use the work of Ernst (2002) as a basis to elaborate the most important success factors of incremental NPD and its implications for the management of these innovation projects. Ernst structured his work in five broad categories derived from the earlier work of Cooper and Kleinschmidt (1995). I will summarize the main findings of Ernst on each of these categories.

#### 3.3.1.1 NPD Process

Of specific importance for the success of products in the NPD process is the quality of planning before entry in the development phase. In specific are mentioned: *‘broad evaluation of ideas’*, *‘execution of technical and market-directed feasibility studies´* and *´commercial evaluation of the NPD project*’ (Ernst 2002, p. 9).

Furthermore during all phases of the NPD project, a continuous commercial assessment and the orientation of the NPD process to the market are important for incremental projects (Dwyer and Mellor, 1991b; Parry and Song, 1994; Souder et al, 1997). Through a process-oriented controlling approach, unprofitable NPD projects can be terminated on a timely and consequent basis (Cooper and Kleinschmidt, 1995). Cooper’s (1990, 2002)Stage Gate Model (see Appendix 1) is a management tool that can help analyze innovation projects at different stages. The purpose is to determine whether a project has approval to proceed (‘Go’), will be terminated (‘No Go’), or will be asked to complete specified actions before the gate decision can be made.

Song and Montoya-Weiss (1998) argue that incremental NPD process should emphasize on the post-development stages, since the firm is drawing on substantial experience with existing markets and technologies.

#### 3.3.1.2 Organization

Cooper and Kleinschmidt recognized some general organizational success-factors for new product development. A cross-functional team which is responsible for the entire project along with a strong project leader are factors that have been acknowledged as successful by different authors (Ernst, 2002). In project management different types of development project teams are recognized. Wheelwright and Clark (1992) described 4 types of project teams. ADD

#### 3.3.1.3 Role and commitment of Senior Management

Support of Senior Management and adequate resource allocation are success factors in NPD (Cooper and Kleinschmidt 1993c, 1995a, 1996). Cooper and Kleinschmidt also claim that resource allocation has to go beyond the R&D budget, because market research expenditures and market launch of the new products are important for the success of new products. This links back to the success factor that market orientation of the NPD process is important.

Balachandra (1984) stated that increased support of senior management increases the probability that projects will not be terminated. It is likely that this has a negative effect on the success of incremental projects, as unprofitable projects will not be terminated in time. In the section about radical innovation projects we will see that not terminating the project due to senior support can have a positive effect on radical projects.

#### 3.3.1.4 Strategy

Ernst (2002) states it is clear that the aspect of NPD strategy has barely been examined in empirical NPD studies to that point. Portfolio management as described earlier is a strategic tool that can provide companies with a well balanced portfolio that is in line with the corporate strategy.

#### 3.3.1.5 Culture

A culture in which there is an innovation friendly climate has a positive effect for both incremental and radical development**.** Together with the support of risk-taking behavior, the innovation friendly climate has been identified as being relevant to success (Vos 1985). The stimulation of idea generation, for example, and providing free time to do creative things within the company have a positive effect on NPD (Cooper and Kleinschmidt, 1995). More specific cultural aspects of incremental development can be found in the work of Ettlie et al (1984). They state that decentralization and formalization are favorable for incremental innovation, while a more central and informal culture are favorable for radical development. Because of the lower uncertainties involved in incremental innovation more processes can be formalized and decentralized. The use of Cooper’s Stage-gate models is one example of a more formalized approach.

### 3.3.2 Management of Radical innovation

As discussed earlier, I will mainly base my this section on the findings in the book of Leifer et al (2000). This book is written by six of the most renowned researchers in the field of management of radical innovation. From 1995 on, twelve radical innovation projects in ten different companies were studied for a five year period. The aim of the book is to advice established companies how to manage radical innovation projects so that they can outsmart the upstart firms.

As mentioned earlier, Leifer at al identified radical innovation as: “a product, process, or service with either unprecedented performance features or familiar features that offer potential for significant improvements in performance or cost” (Leifer et al., 2000, p.5). Radical innovations create such a dramatic change in products, processes or services that they transform existing markets or industries, or create new ones. Given this definition every corporate leader should aim at developing breakthrough innovations, but radical innovations appear to extraordinary difficult to manage. Leifer at al identified several managerial challenges and four types of uncertainty that need to be reduced for radical innovation projects to be successful.

I will first address the challenges in managing radical innovation and its competencies required to cope with the challenge. These challenges are separated in the same categories that Ernst used in his 2002 review. I will add relevant literature findings outside the work of Leifer et al, when I believe this adds useful information.

#### 3.3.2.1 NPD Process

The first recognized managerial challenge is *to capture radical ideas in the “fuzzy front end”*. The required competencies to address this challenge are the generation of good ideas, the recognition of opportunities enabled by breakthroughs and the development and implementation of an effective approach to initial evaluation. The focus of the radical NPD process lies more within the earlier stages of the process, when compared to incremental NPD (Song and Montoya-Weis, 1998). Also the development processes are more exploratory and less customer driven than the incremental processes (Veryzer, 1998).

As read before, Song and Montoya-Weiss (1998) argued that incremental NPD process should emphasize on the post-development stages. On the other hand they claim, the development of really new products should emphasize more on the technical development and product testing activities in order to refine technological capabilities and deliver a really new product.

Also a formalized Stage Gate Model (Cooper; 1995, 2002) is only suitable for incremental innovations and not for radical projects (McCarthy, 2006; McDermott & O’Connor, 1999; Veryzer 1998). WHY?

#### 3.3.2.2 Organization

Leifer at al identified *the management of radical innovation projects* as a challenge. Important are the articulation of a vision, uncertainty-mapping capability, development of a learning plan, recruitment of champions and effective management of organizational interfaces. A champion’s vision for the product and his drive and direction to advance the development of the project are important for discontinues projects to be successfully undertaken (Veryzer, 1999).

Another challenge is *to learn about markets for radical innovation*. To address the challenge, commitment to ask different market research questions play an important role along with the willingness to conduct market research in new ways. As opposed to the management of incremental innovation, in radical innovation projects a continues market-orientation is less important or can even hinder the project. Because of the long-term duration and uncertainties radical projects could be terminated too soon or unjust because (senior) management focuses too much on the short run or current markets.

In order to force radical innovation, an organization can set up an autonomous team. In this team, members from different functional area’s are assigned, dedicated an co-located on the project team (Wheelwright and Clark, 1992). It operates outside the boundaries of the organization and is not required to follow existing practices and procedures. The team takes nothing as ‘given’, but rather builds from scratch, which makes it perfect for radical innovation.

#### 3.3.2.3 Role and commitment of Senior Management

As addressed earlier the role of a *product champion* plays an important role in radical innovation projects. McDermott and O’Connor (2002) recognized two primary groups of leadership roles in radical innovation projects: Sponsors and Champions. A Sponsor is a senior management level project supporter who provides encouragement and financial backing from above to project team members. Especially in long payback projects the sponsor can play a critical role in keeping the project alive. The importance of a project or product Champion has been elaborated multiple times in current literature. Veryzer (1999), for example, states that a Champion’s vision for the product and his drive and direction to advance the development of the project are important for discontinues projects to be successfully undertaken. Swink (2005) adds that a top management champion, who can quickly acquire unanticipated resources, is needed to execute radical new product development.

#### 3.3.2.4 Strategy

A challenge in managing radical innovation is *to resolve uncertainty in the business model* (Leifer et al, 2000). Understanding of what the firm should outsource and what new competencies it should develop is an important competency. Also the adaptation of the business model in response to learning is important.

Furthermore *bridging resource and competency gaps* is important. Resource acquisition along with establishment and management of internal and externals partnerships are competencies required to address this challenge.

And as described earlier, a well balanced portfolio that is in line with the corporate strategy is an important managerial task.

#### 3.3.2.5 Culture

According to Ettlie et al(1985) *centralization* and *informal structures* tend to support radical process adoption. McDermott (1999) also stated that *informal networks* inside and outside the company are important to stimulate radical innovation.

*Engaging individual initiative* was identified by Leifer et al as a challenge for managing radical innovation. They argue initiative can be encouraged by effectively defining the roles of senior management, key individuals and the project team. Building appropriate reward systems and career paths plus the promotion of informal networks also contribute to engage individual initiative.

### 3.3.3 Managing project risk

Leifer et al continue their findings by identifying four dimensions of uncertainty for radical innovation. The first dimensions are *market* uncertainty, and *technical* uncertainty. Radical projects involve high levels of both types of uncertainty, while incremental projects generally have low levels of uncertainty. The other recognized levels are *organizational* uncertainty and *resources* uncertainty.

Organizational uncertainty involves conflicts between the mainstream organization and the radical innovation team, and the difficulty of managing the relationship between them. As for the resources uncertainty it is mainly about which and how funding and competencies should be assigned to the project. For radical projects to mature, uncertainty must be reduced on all four dimensions (Leifer et al, 2000). Rice et al (1998) acknowledge that the primary imperative of driving radical projects is to reduce uncertainty to the point where conventional management practices are appropriate.

#### 3.3.3.1 Reduce market and technology uncertainty

McDermott and O’Connor (2002) observed three approaches to reduce market and technological risks. 1) *Leveraging from known capabilities*, building off of existing strengths helps to move down from the upper right quadrant (see figure ?.?); 2) *Outsourcing*, by forming alliances firms were able to

Figure 3.3 Source: McDermott and O’Connor, 2002, p. 430

continue their breakthrough projects without having all the skills internally and were able to reduce their own risk and move out of the suicide square; and 3) *Choosing not to face all issues of uncertainty concurrently*. This mechanism was used by project teams by simply ignoring certain uncertainty issues to be able to focus their energies at other issues and making progress more quickly.

#### 3.3.3.2 Reduce organization and resources uncertainty

Leifer et al (2000) suggested several mechanisms that firms can put into place to reduce resources and organizational uncertainties. I will shortly describe their most important findings. They claim a radical innovation hub can serve as the repository for cumulative learning about managing innovation. In the hub people from idea hunters to internal venture capitalists meet, which helps manage the interfaces between radical innovation projects and the mainstream organization. The hub can also serve as a benchmarking mechanism. Although comparability between radical projects is difficult, benchmarking can help set management’s expectations regarding timing and expenditures for breakthroughs in their industry.

To discover and encourage radical innovation within the firm the hub can also play a role. Hubs can take a leadership role in establishing an environment and corporate culture that attracts radical innovators and entrepreneurs.

## 3.4 Innovation in Software Development

In this section, the worlds of ‘Innovation’ and ‘Software development’ are linked. The main findings in the academic literature on innovation in software development are discussed, leading to an hypotheses which type of innovation suits the FOSS development best. I have to note that innovation literature on SD is rather scarce. Only a handful of researchers have made assumptions, or did limited empirical research on the different kind of innovations that take place in FOSS development. That makes creating a clear and unified hypotheses on this subject rather difficult, but I believe it does add value to my own research.

### 3.4.1 Previous research on innovation in FOSS

Murray and O’Mahony (2007) state that “under a theoretical viewpoint, it has been claimed that the free circulation of ideas, which characterizes the FOSS production and organization model, is likely to favor knowledge creation and accumulation” . According to Lorenzo and Rossi (2008) the knowledge creation and accumulation in FOSS is positive for promoting innovation.

The intrinsic motivations that drive FOSS developers to contribute to a project are fairly similar to those of scientists doing scientific research (Dalle and David, 2003). They argue that, because the progression of the projects and developers is continuous visible, programmers are stimulated for the development of new code and bright algorithmic solutions. Developers want to show their abilities to their peers. This was also referred to by Lerner and Tirole (2002) as the signaling incentive.

Rossi (2009) tried to answer the question whether programs based on FOSS solutions are more innovative than proprietary ones. She used a sample of 134 software solutions produced by Italian firms and an expert panel. What she found was that FOSS solutions seem to be more innovative than its commercial equivalents. What did she say about radical innovation?

 The researches discussed above all lean towards the theory that FOSS development is more innovative than proprietary development. The degree of innovativeness is an indication of the radicalness. However, Lorenzi and Rossi (2008) state that empirical evidence on innovativeness of FOSS products is scant and controversial. For instance, Tuomi (2005) states that, in his view, there is nothing innovative in a system like Linux, because it simply re-implements functions already present in Windows systems.

Empirical research on innovativeness in the FOSS sector was also conducted by Klincewicz (2005). He evaluated 500 of the most active projects registered at *SourceForge.net*. He used a theoretical sampling approach (“tech mining” software) to analyze the sample. Klincewicz found relatively *low* levels of technical newness, alongside a *high* interest of developers and users in the innovative projects. The low levels of technical newness could be an indication that FOSS focuses more on incremental innovation development. He only found 5 projects who could be labeled as a radical innovation that is as complex and reliable as comparable proprietary products. On the other hand the Economist (2006) published an article in which practitioners from the software field do recognize FOSS solutions as highly reliable and of high quality.

As read before in the section on ‘FOSS projects’, some FOSS projects can take the form of meritocratic hierarchies. Scacchi (2001) suggests that meritocractic enterprises embrace incremental innovation over radical innovation. Although this statement is very clear, ‘meritocratic projects favor incremental innovation over radical innovation’, it is not known what percentage of all projects adopt this meritocratic hierarchy. Therefore the value of this statement is questionable in the light of my research.

In Raymonds (1999) work, Linus’ Law is described. It explains that bugs can be fixed really fast in FOSS development, as there are many eyeballs looking for problems. The fact that small problems can be addressed very fast, because a lot of people make small contributions (recognizing and/or fixing the bugs) gave me the idea that FOSS development was perfect for quick incremental improvements. This is a personal interpretation based on a few lines of the work of Raymond, but it actually formed the basis of my research.

**Incremental**

**Radical**

Rossi, 2009

Murray and O’Mahony, 2007

Dahle and David, 2003

Tuomi, 2005

Klincewicz, 2005

Scacchi, 2001

Linus’s Law

**FOSS**

Figure 3.4 Overview literature on innovation in FOSS

### 3.4.2 Measurement of radicalness in FOSS

Previous indicators for measuring innovation suffer from several shortcomings (Kleinknecht et al., 2002) which turn out to be fairly severe when attempting to measure innovation in the so called New Economy sector (Haskel, 2007). These industries, which FOSS is part of, are characterized by elements that make traditional instruments for measuring innovation (like patents or trademarks) useless (Dahling and Behrens, 2005). After all the idea of FOSS is to discourage the use of patents and trademarks.

As the measurement of radicalness in OSS fails in traditional methods, I have come up with a more practical solution for my research. To capture the radicalness of the FOSS projects I have asked a panel of 3 industry experts to rate the projects on three different items. These items are derived from the innovation literature as discussed earlier in this chapter and are: newness to market, technology newness, and the impact of the project on the industry. The scores provided by the experts will give an indication how radical a project is. A more extensive elaboration on the measurement and methodology can be found in the chapter 4 *Data*.

### 3.4.3 Success in FOSS projects

Different measures to define the success of open source projects have been proposed in the literature. Moving from the more recent contributions, these can be classified into three categories (Comino et al, 2007): (1) software use, (2) size of the community and/or its level of activity and (3) technical achievements of the project. According to the first category, an OS software project is successful when it is widely adopted among users. In the second case it is successful when a large and active community of developers contributes to its production. The third category focuses on what a project contributes on technical terms. In my research success is measured on basis of the software usage and adoption. This category has the meeste aanhangers + waarom + waarom niet de andere categorien.

Leg uit welke ik gebruik (software use)

### 3.4.4 Hypotheses

In section 3.4.1 I concluded that the literature on innovation in FOSS is scarce and controversial. Even the two empirical articles both have different conclusions when it comes to describing FOSS as either suitable for incremental or radical innovation. Therefore I expect to find little or no significant differences between the degree of radicalness and project success in FOSS. Thus I expect:

Hypothesis 1: *The Free/Open Source Software platform is not particular suitable for the development of either incremental or radical software.*

Incremental Innovation

Radical Innovation

Free/Open Source Software platform

+/-

+/-

Figure 1.2 Relation between the research concepts

### 3.3.4 Conclusion

A lot of research has been done to recognize success factors in incremental NPD and its implications for managing incremental innovation projects. But as for the management of radical innovation projects there is no blueprint for how a project should be managed. There are some critical success factors that are clearly different when compared to incremental innovation projects. The use of a formalized stage gate model, for example, has a positive effect in incremental development, but typically hinders development in radical projects. The same goes for continuous commercial assessment and market focus, which has only a positive effect in incremental development.

Focus on the earlier phases on the NPD process and an autonomous team operating outside the original organization are elements that are profitable for radical innovation success. Furthermore management of radical innovation is about managing risk. By reducing uncertainty, radical projects become better manageable and the chances of success increase.

An overview of the main differences of different elements between incremental and radical innovation projects is presented in Appendix 2. The table is mainly derived from the book of Leifer et al (2000, p.19-20) and can offer additional insight.

The ‘traditional’ management methods in the current literature are difficult to apply to FOSS project management.

As stated earlier, the work is not assigned to people, but OSS developers undertake the work they choose to undertake. Also there is no explicit system-level or even detailed design, no project plan, schedule, or list of deliverables.

#  Data

## 4.1 Data

The projects that are examined in this research are all registered at SourceForge.net. SourceForge.net is a website originally founded by VA Linux systems. It is a comprehensive portal for FOSS projects, providing essential project management tools for software developer communities, including shared code repositories and discussion forums.

To get specific information from sf.net I made use of the data available on the FLOSSmole website. FLOSSmole (formerly known as OSSmole) is a set of tools for gathering data (metrics) about the development of Free/Open Source projects. They ‘crawl’[[10]](#footnote-11) project sites like sf.net and present the data in a structured way on a monthly basis.

A list of 100 FOSS projects was created based on project activity and downloads. The projects had to be registered before July 12008 and ought to have a development status of at least ‘stable’ (see section 2.2.3 *Project lifecycle*). This list was reviewed by two FOSS experts who both suggested adding several larger projects that were missing in this list. The main reason they were not in the original list is because they have grown beyond the SourceForge website, resulting in a low measured activity or they would never make it to the development status ‘stable’. In order to get a better expert feedback 15 projects operating outside SourceForge were added to the original list of 100 projects resulting in a list of 115 FOSS projects.

A spreadsheet was created containing the data *project name*, *project description*, *developer count* and  *date registered*. Originally *number of downloads* was also mentioned, but due to inconsistency (only downloads from the Sourceforge.net portal were counted while there are many download portals on the internet, that are not taken in account) it was removed. *Developer counts* from the 15 projects that were added to the original list were derived from those websites. Added to the sheet later where the *license type* of the project and the *category*(like desktop, office and programming) in which the project operates.

## 4.2 Methodology

A panel of 3 industry FOSS experts were asked to answer 4 questions on each project. The first three questions combined were to determine whether the project could be classified as *radical* or *incremental* or in other words to determine the radicalness of the project.

**Question 1: Was the Project new to the software world when introduced? Answer in 1-5, where 1 is not new at all and 5 is very new.**

This question refers to *what* the software solution *does*. Is the software innovative in the sense that it better satisfies needs or requests from users than other solutions available in the market? Is the software new to the software market?

**Question 2: Was the project new to the world under technological viewpoint? Answer in 1 - 5, where 1 is not new at all and 5 is very new.**

This question refers to *how* the software succeeds in accomplishing a given task. In what sense are the technical or architectural aspects responsible for offering new solutions to users? What is the technology newness?

**Question 3: What was the impact the project had on the software world? Answer in 1-5, where 1 is no impact and 5 is a very high impact.**

In what sense was the project able to leave its footprint in the software world? Did it set a new standard, did it transform the existing market, was it responsible for creating new markets or destroy existing competencies?

The final score which defines the *radicalness* for each project, will be the average score of the first three questions. In this case the score will be a number between 1 and 5. *Radicalness* is a term that I use to define the distribution of projects on a scale from incremental to radical. In this case it means that a *radicalness* score of ‘1’ means that the project is rated as a very incremental project. When a project is rated as a ‘5’ it means it can be viewed as a very radical innovation project performing high on all levels of ‘newness’ and ‘impact’. When a project is rated as ‘3’ for example, it gives an indication that it probably has a certain degree of newness, but it does not have the impact and newness of a truly radical project. Yet it does outperform a truly incremental project on market and/or technology newness and/or impact.

**Question 4: Project success**

The fourth question was to reveal an implication of *project success*. Experts were asked to classify the success of each project as either *low (=1)*, *average (=2)* or *high (=3)*. *Success* is rated on basis of the project software usage and adoption. Note that project success is not the same as project *impact* as rated in the ‘radicalness’ section. Projects that did not transform (new) markets or destroyed existing competencies can be very popular and widely adopted. I do expect to find a correlation between *Impact* and *Success*.

## 4.3 Industry Experts

Three experts were sent the spreadsheet with an instruction how to fill in the sheet and additional information on the questions. A short part of the sheet is shown in figure 5.1. The first column contains the project name as it is registered at sourceforge.net. In the second column a short description of the project is formulated. The third column contains the date that the project is registered in the sourceforge.net database. And the final 4 columns are for the experts to fill in their projects scores for the corresponding 4 questions.



Figure 5. 4‑1

Expert A is a 30 year old researcher. He studied Innovation Science at the Technische Universiteit. Expert A has been active in Open Source since 1999, both on private and businesslike grounds. Nowadays he is active within the NoIV, a program that informs Dutch government organizations about the possibilities of Open Standards and Open Source Software.

Expert B studied at the TU Delft. He is currently active as a senior software engineer at both Volvo and CIMSOLUTIONS. He has been interested in Free/Open Source since 1999.

Expert C is a 37 year old Open Source Solutions consultant. His FOSS background consists mainly of consulting and implementing ICT Open Source projects. He enjoyed an education at the TU Eindhoven.

# Empirical Results

This section presents and discusses the empirical results of my research. In section 5.1 the results of rated projects radicalness and success will be compared to help determine what type of innovation is most successful in FOSS development. Later different project categories, projects licenses and number of developers per project are compared on radicalness and success to see if there are any interesting patterns. First I present table 5.1 which summarizes the most important numbers derived from the data sample.

|  |  |  |
| --- | --- | --- |
| Number of projects | **115** |  |
|   | Average | Standard deviation |
| project radicalness | 3,07 | 0,71 |
| project success | 1,74 | 0,79 |
| newness to market | 2,96 | 0,68 |
| tech. newness | 3,26 | 0,75 |
| impact | 2,77 | 1,21 |

##

The sample contains 115 FOSS projects. The average project radicalness is 3.07. This radicalness score is derived from the scores of *newness to market*, *technological newness*, and *impact of the project*. The average project success is 1.74. These numbers are useful for making comparisons in the following sections.

## 5.1 FOSS development: Incremental or Radical?

The main focus of this research was to find out whether the Free/Open Source environment is more suitable for incremental or for radical innovation projects.

Hypothesis 1: *The Free/Open Source Software platform is not particular suitable for the development of either incremental or radical software.*

In order to make any assumptions it would thus be useful to compare project *radicalness* with project *success*. I will look for patterns and relations between the two metrics in order to make a statement.

To start, I have divided the projects in four sections based on *radicalness* and *success* as can be seen in figure 5.1. The sections are separated on the basis of the average measured *radicalness* and *success*. For example: in the upper left corner are projects that are rated higher than average on *success*, but are rated below average on *radicalness*. I have simply counted the projects in my sample (Nx) that are in each section.

Figure 5.1 Number of projects divided by average radicalness and success

**-**

**+**

**-**

**+**

N4= 12

N2= 39

N1= 15

N3= 49

Radicalness

Average Radicalness = 3.07

Success

**+**

**+**

**-**

**-**

Average success = 1.74

By simply counting the projects in each section we can already see some sort of segregation. Section N2, with an above average radicalness and success score, and section N3, with an below average score on both radicalness and success, contain far more projects than section N1 and N4. This could be an indication of a relationship between project *radicalness* and project *success*. If we look at the spread and correlation of *radicalness* and *success* (see figure 5.2) we can see some sort of pattern between the two metrics.

Figure 5.2 Spread project radicalness versus success

It appears that on average, the higher a project scores on radicalness, the higher a project scores on success. This is demonstrated again in figure 5.3. In this figure the radicalness is divided into groups with a 0.5 interval on the scale of radicalness. Depicted are the number of projects per interval and the average success of the projects within the interval. For example, there are 20 projects that range in radicalness from 2 to 2.5. These 20 projects have an average success of 2.07.

Figure 5.3 Average project radicalness versus success

This picture shows a bell shaped distribution of the radicalness of the sample, indicating that there are few projects in the extremes of either very incremental or very radical. Seventy-seven percent of the projects have a radicalness of 2 to 4, which indicates they have some degree of newness, but cannot be rated as very incremental. When we look at the average success of the projects per radicalness interval we see some sort of relation. On average, the higher a project is rated on radicalness, the higher the project scores on success. In other words, more radical rated FOSS projects score better on success than the projects who can be labeled as more incremental.

In the next sections I will have a look at certain characteristics of the FOSS projects that are available in my data sample. These characteristics are: Software development categories, Projects Licenses and Number of developers per project. It hope to see some interesting results which can maybe help clarify the relation between radicalness and success as found in this thesis.

## 5.2 Software development Categories

One of the experts from the panel suggested dividing the projects into different categories in order to see in what areas Free/Open Source solutions perform best. The projects are divided in six main categories: desktop, database/server/network, tools, office, programming and communication. The distribution of the projects is shown in figure 5.5.

Figure 5.5 Distribution of the projects per category

The category *desktop* contains the most projects. It is a rather broad category, with software like file-sharing programs and media players. Almost 20% of the projects can be labeled as *programming* projects. These projects have the aim to develop a product that can be used to write new programs/projects. Examples are PHP, Perl and Python. Figure 5.6 shows the average radicalness and success of the projects per category.

Figure 5.6 Radicalness and Success per category

The categories *database/server/network* and *programming* perform well above the average radicalness (average radicalness is depicted with the blue horizontal line). Also on success these two categories perform above average (success average is depicted with the red horizontal line).

What is noticeable is that the category *communication* is rated lowest on radicalness, but scores very high on project success. It seems that programs designed to communicate do not include very radical designs or solutions, but are widely adopted.

## 5.3 Licenses

In this section I will compare the use of different software licenses that the sampled projects use. Figure 5.7 shows the distribution of the projects per License. The category labeled as *FSF approved* contains special licenses that are acknowledged by the Free Software Foundation. These are licenses granted for large projects like PHP and Python. Under the category *other* are licenses like ‘Z lib/ libpng license’, the ‘artistic license’ and the ‘NASA opensource license’.

Figure 5.7 Distribution of the projects per License

In my sample, 68% of the projects used the GPL license. The LGPL license took 10% for its account.

Figure 5.8 Radicalness and Success per License-type

Figure 5.8 depicts the rated radicalness and success for each of the licenses. The red line is the average radicalness of all projects at 3.10. The FSF approved licenses score the highest on radicalness (4.11). Furthermore the MPL and BSD license score above the average radicalness. The GPL scores the lowest, just under 3 points.

It is difficult to draw any conclusions on this sample. It is clear that the *FSF approved* licenses score very high on radicalness and success, but this is average is based on only four projects (namely the Apache webserver, PHP, Python and the Ruby on Rails project). These very projects actually formed the basis for those licenses (the Apache Software License, the PHP License, the Python License and the Ruby License). Derivative programs based on this license will probably be less radical and less successful. Also the more commercially oriented Mozilla Public License (MPL) scores above average on both radicalness and success. Again, the scores of the MPL are based on only four projects. Furthermore no shocking results can be derived from this sample.

## 5.4 Developers

Since the developers per project data is available, it could be interesting to see whether there are any relations with project success and radicalness. The number of developers per project ranged from one per project to several thousand developers for the really large projects. In figure 5.9 is shown how many projects there are based on the number of developers per project.

Figure 5.9 Distribution of the projects on Number of developers per project

What perhaps is most surprising is that 92% of the projects have less than 100 developers registered, and 50% of all projects even have a maximum of 10 developers per project. There are 4 projects in the sample that have more than 500 developers contributing.

I have visualized the average radicalness and the average success per *number of developers group* in figure 5.10. The first thing I notice is the significant difference between the average radicalness of the group of more than 500 developers and the other groups. The projects with more than 500 developers seem to be able to produce more radical products than the other groups. Although it must be said that the value of 4.33 is the average of only a sample of 4 projects. Furthermore the group of 6 to 10 developers per project produces the least radical products. The group of 1 to 2 developers outperforms the other groups with developers up to 500, be it only with a slight difference.

Figure 5.10 developers per project versus average radicalness and succe

When I compare the number of developers per project with project success I get almost similar results as in the developers – radicalness comparison. Again it is the group of the projects with more than 500 developers that clearly outperforms the other groups on project success. Also the groups of 1 to 2 developers per project slightly outperforms the remaining groups with a slight difference.

A reason why the projects with more than 500 developers are so successful could be that there was a need for (different functions of) the project in the earlier phases. Developers join the project because they notice the project has potential and they contribute to eventually use the software themselves. It could also be the other way round. A project is already widely adopted and becomes more and more popular, with the result that it attracts and accumulates even more developers.

# Summary and Conclusion

## Problem definition

The central problem definition of this thesis:

**What kind of innovation suits the FOSS platform best?**

Sub-questions:

1) What kind of innovation takes place in FOSS development projects?

2) Which type of innovation projects are most successful in FOSS development?

3) Is there a relation between the type of innovation and the success of projects in FOSS development?

## Summary

The literature review begins with a section on FOSS development. Several issues like FOSS developer motivation and strategies for commercial companies to enter FOSS development were discussed. The focus on this section was on characteristics of FOSS development projects. Most noticeable is that the structure of FOSS projects is not completely flat. Project members can contribute to a project by fulfilling a role varying from project leader to passive user. Each role forces a different influence on the project.

The most important part on the innovation literature section was on the management of both incremental and radical innovation projects. Differences between the two have been discussed. The value of traditional innovation literature on FOSS development is very limited as FOSS development operates under different rules. Furthermore previous literature on innovation in FOSS was discussed in order to formulate an hypothesis on the issue which type of innovation fits the FOSS platform best.

As traditional measurement of innovation fails in FOSS development I came up with an alternative method for measuring the radicalness of innovation. A list of 115 FOSS projects was created from the sourceforge.net website. Industry experts were asked to rate the projects on different items. Together these rating would give an implication of each projects radicalness and success.

## Results

1) What kind of innovation takes place in FOSS development projects?

The radicalness of the projects measured ranged from 1.3 to 5.0. This implies that there are very incremental projects with no to low levels of newness, but also extremely radical, market changing projects. The distribution of the projects on radicalness, as can be seen in figure 5.3, is bell shaped. Most of the sampled projects are concentrated between 2 and 4 on the scale of radicalness, indicating they inhibit some forms of newness. Fourteen percent of the projects are rated very radical, with an radicalness score of 4 or higher. These are the projects that have high levels of newness and have significant impact on the software world.

2) Which type of innovation projects are most successful in FOSS development?

The projects that are rated highest on success are the projects with a radicalness of 4 to 5 (see figure 5.3). So especially the projects that are considered very radical are widely adopted among users.

3) Is there a relation between the type of innovation and the success of projects in FOSS development?

Figures 5.2 and 5.3 indicate a relation between project radicalness and success. A pattern is visible in which the rate of success rises with the rate of radicalness. On average, the more radical a project is rated, the higher it will score on success.

**What kind of innovation suits the FOSS platform best?**

This research showed that the FOSS platform is not limited to just incremental projects. The projects ranged from very low levels of newness to very radical innovation projects. Especially the projects that are recognized as very radical score high on success. A pattern is visible in which the rate of success increases with the rate of radicalness. This gives the implication that the FOSS platform is better suitable for more radical innovation projects. Especially the areas of *programming* and *database/server/network* are able to deliver radical and successful software solutions.

I will try to clarify this relation by reverting back into the literature section and with the input of an industry expert who I confronted with my findings.

It looks like the free circulation of ideas (as described by Murray and O’Mahony, 2007) in FOSS development promotes innovation which could lead to more radical projects. Because developers want others to notice their work, the signaling incentive (Lerner and Tirole, 2002) can stimulate developers into more creative and innovative behavior, hence more radical solutions.

A FOSS project team operates more like an autonomous team (Wheelwirght and Clarck, 1992) (used to foster radical innovations) than a lightweight team (more suitable for incremental innovations). An autonomous team operates outside the boundaries of the original organization. The team is not required to follow existing organizational practices and procedures. Also it is not limited by a strong market focus of the original company. A FOSS project team has similar characteristics. It is not limited to the boundaries of an organization, but operates as a dedicated, independent team. Nor does it have to worry whether their product will be profitable on the market. These are all characteristics that have a positive effect on the adoption of radical innovation. Thus could help explain why the FOSS community is able to successfully produce more radical products.

Furthermore FOSS projects are described as informal and highly centralized, in the sense that most of the work and decisions are in the hands of a small group of core developers (Mockus et al, 2002; Nakakoji et al, 2002). According to Ettlie et al (1987), centralization and informal structures tend to support the process of radical innovation.

Expert C was willing to give a reaction on the results of the research that I presented him. He said he was not surprised by the statement that FOSS is better suitable for more radical projects. He says, that a project often becomes successful when there is a demand for some kind of functionality that is not available in the market yet. “You often see users and developers start to support the project, because there is a certain need and interest in the non-existing functionality”. According to expert C, “an implementation of a less radical functionality, of which there is already a solution in the market, is often less successful. Unless the existing implementation is not sufficient or very expensive”. The statement of expert C focuses on non-existing functions in current software that FOSS projects try to develop. The ‘users are developers’ quote could indicate that users can better satisfy their own needs in the non-existing function by contributing to the development. The more radical solution will be more widely adopted because it better satisfies the functional or technological needs of users.

Also I measured that 50% of the projects have 10 or less developers registered. On average, no significant differences in radicalness and success have been measured between projects ranging from 1 up to 500 developers. A project with only 1 or 2 developers can outperform projects with more than 100 developers. Only projects with more than 500 developers are rated very radical and successful.

## Pratical implication

To provide a practical example. There is a commercial software company specialized in database software. A market research report shows there is a small upcoming niche market for database software designed for a new Open Source operating system for governmental institutions. At this time, the market is not big enough to allocate a lot of resources to a new in-house project. What the company could do is use a *second generation Open source (OSSg2)* Model (Watson et al, 2008), to set up an Open Source project. It could act as a corporate sponsor by (partly) releasing previous in-house proprietary code under a license to form the basis of the project. FOSS developers would be interested in the project, as there is a need for a radical new database design for the new system. Database software appears to be an area in which FOSS development projects score well. The company could make revenues by delivering complementary services and can acquire new knowledge for their own products in other markets. If the new market offers bigger potential in the future the company already has a head-start on the competition by keeping (tight) control over the software code. In this case both the company and the FOSS world can benefit on the project.

## Reflection and recommendation

? anders moeten doen

?waarde resultaten

The value of the results is limited. A sample of 115 projects in not large enough to form a conclusion that is generally applicable.

?wat zelf geleerd

Aanbevelingen voor onderzoek

Relative small sample > bigger sample would be better

More experts to reduce the impact of subjective evaluations

Number of developers. > Successful projects attract more developers so results could be a bit biased.

# Appendix

## Appendix 1 Cooper’s Stage Gate Model

Source: Cooper (1990) p.46

## Appendix 2 Overview differences radical en incremental innovation

|  |  |  |
| --- | --- | --- |
|  | Incremental | Radical |
| Project time line | Short-term – six months to two years. | Long term – usually ten years or more |
| Trajectory | There is a linear and continues path from concept to commercialization following designated steps. | The path is marked by multiple discontinuities that must be bridged. The process is sporadic with many stops and starts, hibernations and revivals. Trajectory changes occur in response to unanticipated events, outcomes and discoveries. |
| Idea generation and opportunity recognition | Occurs at the front end; critical events are largely anticipated. | Occur sporadically throughout the life cycle, often in response to discontinuities (funding personnel, technical, market) in the project trajectory. |
| Process | A formal, approved process moves from idea generation through development and commercialization. | There is a formal process for getting and keeping funding, which is treated by participants as a game, often with disdain. Uncertainty is too high to make the process relevant. The formal process has real value only when the project enters later stages of development. |
| Business case | A complete and detailed plan can be developed at the beginning of the process because of the relatively low level of uncertainty. | The business model evolves through discovery-based technical and market learning and likewise the business plan must evolve as uncertainty is reduced. |
| The players | Assigned to a cross-functional team, each member has a clearly specified responsibility within his or her area of expertise. | Key players come and go during the early life of a project. Any are part of the informal network that grows up around a radical innovation project. Key players tend to be “cross-functional” individuals. |
| Organizational structures | Typically, a cross-functional project team operates within a business unit. | The project often starts in R&D, migrates into some sort of incubating organization, and transitions into a goal-driven project organization. |
| Resources and competencies | The project team has all the competencies required to complete the process. The project is subject to the standard resource allocation process for incremental projects. | Creativity and skill in resource and competency acquisition – from a variety of internal and external sources – are critical the survival and success of the project. |
| Operating unit involvement | Operating units are involved from the beginning. | Informal involvement with operating units is important, but the project must avoid becoming captive to an operating unit too early. |

Source: Leifer et al (2002) table 2-1, p. 19-20

## Apendix 3 Newness to market, technology newness and impact versus success.

This figure shows the project success in relation to the newness to the market, the technology newness and impact of the rated projects. The three variables form the building blocks of the average project radicalness. Project impact is almost a in linear function with success. Newness to market and technology newness have similar patterns. All three variables score higher on success when they are rated higher on newness/impact themselves.

# References

Ägerflak, P. and Fitzgerald, B. (2008), “Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy”, *MIS Quarterly*, Vol. 32, No. 2, pp. 385-409.

Anderson, P. and Tushman, M. (1990). “Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change”, Administrative Science Quarterly, Vol. 35, No.4, pp. 604-633.

Atuahene-Gima, K. (1995), “Involving organizational buyers in new product development”, *Industrial Marketing Management*, Vol. 24, No. 3, pp. 215-226.

Banbury, C. and Mitchell, W. (1995), “The Effect of Introducing Important Incremental Innovations on Market Share and Business Survival”, *Strategic Management Journal*, Vol. 16,

No. S1, pp. 161-182.

Castelluccio, M. (2008) “Enterprise Open Source Adoption”, *Strategic Finance,*  Vol. 90, No. 5, pp.  57-58.

Chesbrough, H. and Teece, D. (2002), “Organizing for Innovation: When is Virtual Virtuous?”, *Harvard Business Review*, Volume 80, No. 8, pp. 127-134.

Christensen, C. (1997), *The Innovator’s Dilemma, when new technologies cause great firms to fail*, Harvard Business School Press, Boston, Massachusetts.

Christensen, C. and Bower, J. (1996), “Customer Power, Strategic Investment, and the Failure of Leading Firms”, *Strategic Management Journal*, Vol. 17, No. 3, pp. 197-218.

Cooper, R. (1990), “Stage-Gate Systems: A New Tool for Managing New Products”, *Business Horizons*, Vol. 33, No. 3, pp. 44-54.

Cooper, R. and Kleinschmidt, E. (1995), “Benchmarking the Firm’s Critical Success Factors in New Product Development”, *The Journal of Product Innovation Management*, Vol. 12, No. 5, pp.374-391.

Cooper, R., Edgett, S. and Kleinschmidt, E (2002), “Optimizing the Stage-Gate® Process, What Best Practice Companies are Doing – Part One”, *Research Technology Management*, Vol. 45, No 5, pp. 21-27.

Crowston, K. and Howison, J. (2006), “Hierarchy and Centralization in Free and Open Source Software Team Communications”, Knowledge, Technology, and Policy, Vol. 18, No. 4, pp. 65-85.

Crowston, K. and Scozzi, B. (2002), “Open Source Software Projects as Virtual Organizations: Competenctyrallying for Software Development”, *IEE Proceedings Software*, Vol. 149, No. 1, pp. 3-17.

Dalle, J. and David, P. (2003) “The allocation of software development resources in Open

Source production mode”, *SIEPR Discussion Paper* , No. 02-27, Stanford Institute for Economic

Policy Research, Stanford University, Stanford, CA.

Day, G., Schoemaker, P. and Gunther, R. (2000), *Wharton On Managing Emerging Technologies*, Wiley & Sons, New York.

Garcia, . and Calantone, R. (2002), “A critical look at technological innovation typology and innovativeness terminology: a literature review”, *The Journal of Product Innovation Management*, Vol. 19, No. 2, pp. 110-132.

Hage, J. (1980), *Theories of Organization*, Wiley Interscience, New York.

Herbsleb, J. and Grinter, E. (1999), “Splitting the organization and integrating the code: Conway's law revisited”, *Proceedings of the 21st international conference on Software engineering*, May 16-22, Los Angeles, California, United States, pp. 85-95.

Ernst, H. (2002), “Success factos of new product development: a review of the empirical literature”, *International Journal of Management Reviews*, Vol. 4, No. 1, pp. 1-40.

Ettlie, J. (1983), “Organizational Policy and Innovation Among Suppliers to the Food Processing Sector”, *Academy of Management Journal*, Vol. 26, No. 1, pp. 27-44.

Ettlie, J. and Rubinstein, A. (1987), “Firm size and product innovation”, *Journal of Product Innovation Management*, Vol. 4, No. 2, pp. 89-108.

Fielding, R. (1999), “Shared leadership in the Apache project”, *Communications of the ACM*, Vol. 42, No. 4, pp 42-43.

Fisher, J. (2007) “Incremental Innovations”, *NPN,* Vol.99, No. 5, p. 46.

Fitzgerald, B. (2006), “The Transformation of Open Source Software”, *MIS Quarterly*, Vol. 30, No. 3, pp. 587-598.

Franke, N. and Von Hippel, E. (2003), “Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software”, *Research Policy*, Vol. 32, No. 7, pp. 1199-1215.

Kim, A. (2000), *Community-Building on the Web: Secret Strategies for Successful Online communities*, Peachpit Press.

Kleinschmidt, E. and Cooper, R. (1991), “Impact of Product Innovativeness on Performance”, *The Journal of Product Innovation Management*, Vol. 8, No. 4, pp. 240-251.

Leifer, R., McDermott, C., O’Connor, G., Peters, L., Rice, M. and Veryzer, R. (2000), *Radical Innovation, How Mature Companies Can Outsmart Upstarts*, Harvard Business School Press, Boston, Massachusetts.

Lerner, J. and Tirole, J. (2002), “Some Simple Economics of Open Source”, *Journal of Industrial Economics*, Vol. 50, No. 2, pp. 197-234.

Lorenzi, D. and Rossi, C. (2008), “Openess and Innovation: Comparing Free/Open Source and Proprietary Solution”, *Working Paper Series*.

McGhee, D. (2007), “Free and Open Source Software Licenses: Benefits, Risks, and Steps Toward Ensuring Compliance”, *Intellectual Property & Technology Law Journal*, Vol. 19, No. 11, pp. 5-11.

Mockus, A., Fielding, R. And Herbsleb, J. (1999), “A case study of open source development: The apache server”, *In 22nd International Conference on Software Engineering*, Limerick, Ireland, June 4-11 2000, pp. 263-272.

Mockus, A., Fielding, R. And Herbsleb, J. (2002), “Two case studies of open source software development: Apache and Mozilla”, *ACM Transaction on Software Engineering and Methodology*, Vol. 11, No. 3, pp. 309-346.

Muson, F. and Pelz, D. (1977), “The Innovating Process: A Conceptual Framework”, *Working paper*, University of Michigan.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. and Ye, Y. (2002), “Evolution Patterns of Open-Source Software Systems and Communities”, *In Proceedings of the international workshop on Principles of software evolution*, at http://www.kid.rcast.u-tokyo.ac.jp/~kumiyo/mypapers/IWPSE2002.pdf, (accessed March 11, 2009).

O’Connor, G. (1998), “Market Learning and Radical Innovation: A Cross Case Comparison of Eight Radical Innovation Projects”, *Journal of Product Innovation Management*, Vol. 15, No. 2, pp. 151-166.

O’Reilly, T. (1999), “Lessons from Open-Source Software Development”, *Communications of the ACM*, Vol. 42, No. 4, pp. 32-37.

OECD (1991), *The nature of innovation and the evolution of the productive system*, Technology and productivity-the challenge for economic policy, Paris: OECD, pp. 303–314.

Pavelicek, R. (2000), *Embracing Insanity: Open Source Software Development*, SAMS Publishing, Indianapolis, IN.

Raymond, E. S. (1999), “The Cathedral and the Bazaar”, http://www.tuxedo.org/-

esr/writings/catherdral-bazaar (accessed February 1, 2009).

Rice, M., O'Connor, G., Peters, L., Morone, J. (1998), “Managing discontinuous innovation”, *Research Technology Management*, Vol. 41, No. 3, pp. 52-58.

Riehle, D., Ellenberger, J., Menahem, T., Mikhailovski, B., Natchetoi,Y., Naveh, B., Odenwald, T. (2009), “Open Collaboration within Corporations Using Software Forges”, IEEE Software, edition26 (2): p. 52-58.

Rossi, C. (2009), “Software innovativeness. A comparison between proprietary and Free/Open Source solutions offered by Italian SMEs”, *R&D Management*, Vol. 39, No. 2, pp. 153-169.

Scacchi, W. (2007), “Free/Open Source Software Development: Recent Research Results and Emerging Opportunities,” *Proc. 6th Joint Meeting European Software Eng. Conf. and the ACM SIGSOFT Symp. Foundations of Software Eng.* (ESEC/FSE 07), ACM Press, pp. 459–468.

Song, X. and Montoya-Weiss, M. (1998), “Critical Development Activities for Really New versus Incremental Products”, *Journal of Product Innovation Management*, Vol. 15, No. 2, pp. 124-135.

Stewart, K. and Gosain, S. (2001), “An Exploratory Study of Ideology and Trust in Open Source Development Groups”, *Proceedings of the 22nd International Conference on Information systems (ICIS-2001)*, in New Orleans, LA.

Swink, M. (2005), “Exploring new product innovation types and performance: the roles of project leadership, functional influences, and design integration”, *International Journal of Product Development*, Volume 1, Nos. 3-4, pp. 241-260.

Tuomi, I. (2005) *The Future of Open Source, How Open is the Future?*, VUB Brussels University Press, Brussels, pp. 429-459.

Veryzer, R. (1998), “Discontinuous Innovation and the New Product Development Process”, *The Journal of Product Innovation Management*, Vol. 15, No.4, pp. 304-321.

Vixie, P. (1999)“Software engineering”, In *Open Sources: Voices from the Open Source Revolution*, O’Reilly, Sebastopol, California, pp. 91-100.

Watson, R., Boudreau, M., York, P., Greiner, M. and Wynn, D. (2008), “The Business of Open Source”, Communication of the ACM , Vol. 51, No. 4, pp. 41-46.

Wheeler, D. (2007),”Why Open Source Software / Free Software (OSS/FS, FLOSS or FOSS)? Look at the Numbers!”, http://www.dwheeler.com/oss\_fs\_why.html, accessed 24-9-2009.

Zaltman, G., Duncan, R. and Holbek, J. (1973), *Innovations and Organizations*, John Wiley & Sons, New York, 1973.

1. The Open Source Initiative (OSI) is a non-profit corporation formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open-source community. Derived from www.opensource.org, accessed 19-8-2009. [↑](#footnote-ref-2)
2. www.fsf.org [↑](#footnote-ref-3)
3. A major Linux distribution vendor, http://www.redhat.com/ [↑](#footnote-ref-4)
4. International Business Machines (IBM) Corporation is a [multinational](http://en.wikipedia.org/wiki/Multinational_corporation) [computer](http://en.wikipedia.org/wiki/Computer) [technology](http://en.wikipedia.org/wiki/Technology) and [IT consulting](http://en.wikipedia.org/wiki/IT_consulting) [corporation](http://en.wikipedia.org/wiki/Corporation) headquartered in the [United States](http://en.wikipedia.org/wiki/United_States). The company is one of the few [information technology](http://en.wikipedia.org/wiki/Information_technology) companies with a continuous history dating back to the 19th century. Wikipedia, 2009. [↑](#footnote-ref-5)
5. The Apache web server has been the most popular http server in the internet since 1996 and is developed and maintained by an open community of developers. [↑](#footnote-ref-6)
6. www.opensource.org/docs/osd, accessed 19-3-2009 [↑](#footnote-ref-7)
7. Derived from the sourceforge.net database [↑](#footnote-ref-8)
8. Information retrieved from http://ossmole.sourceforge.net/, accessed 22-04-2009. [↑](#footnote-ref-9)
9. The GNU Project was launched in 1984 to develop a complete Unix-like operating system which is free software: the GNU system. Derived from www.gnu.org, accessed 30-07-2009. [↑](#footnote-ref-10)
10. A Web crawler is a computer program that browses the [World Wide Web](http://en.wikipedia.org/wiki/World_Wide_Web) in a methodical, automated manner. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will [index](http://en.wikipedia.org/wiki/Index_%28search_engine%29) the downloaded pages to provide fast searches. Source: Wikipedia, 24-7-2009. [↑](#footnote-ref-11)