# Abstract

**Table of contents**

Table of contents

[1. Abstract 1](#_Toc228343015)

[2. Introduction 2](#_Toc228343016)

[2.1 Motivation 2](#_Toc228343017)

[2.2 Problem Field 2](#_Toc228343018)

[2.3 Aim of Research 2](#_Toc228343019)

[3. Free and Open Source Software 3](#_Toc228343020)

[3.1 The Free / Open Source phenomenon 3](#_Toc228343021)

[3.1.1 FOSS developer motivation 4](#_Toc228343022)

[3.1.2 Business models, Involvement of commercial companies 4](#_Toc228343023)

[3.2 FOSS Projects 5](#_Toc228343024)

[3.2.1 Roles of Project Members 5](#_Toc228343025)

[3.2.2 Project/Community Structure 6](#_Toc228343026)

[3.2.3 Project lifecycle 7](#_Toc228343027)

[3.3 FOSS project characteristics 8](#_Toc228343028)

[3.3.1 Centralization 8](#_Toc228343029)

[3.3.2 Formalization (formal or informal ties). 8](#_Toc228343030)

[3.3.3 Complexity 8](#_Toc228343031)

[4. Radical and Incremental innovation 8](#_Toc228343032)

[4.1 Definitions of radical and incremental innovation 8](#_Toc228343033)

[4.1.1 Technological discontinuities 9](#_Toc228343034)

[4.1.2 The product life cycle model 9](#_Toc228343035)

[4.1.2 (Economical) effects of radical and incremental innovation (?) 10](#_Toc228343036)

[4.1.x Differences radical and incremental innovation 10](#_Toc228343037)

[4.3 Measuring innovation 10](#_Toc228343038)

[4.4 Previous research on innovation in FOSS 10](#_Toc228343039)

[4.4 Organizational structure and types of innovation 10](#_Toc228343040)

[4.5 Innovation and success rate 11](#_Toc228343041)

[5. Data and Methodology 13](#_Toc228343042)

[5.1 Data 13](#_Toc228343043)

[5.2 Methodology 13](#_Toc228343044)

[5.3 Industry Experts 14](#_Toc228343045)

[6. Empirical Results 14](#_Toc228343046)

[7. Conclusions 16](#_Toc228343047)

[8. References 16](#_Toc228343048)

# Introduction

## 2.1 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.

## 2.2 Problem Field

Over the past 10 years, open source software has become an important cornerstone of the software industry (Riehle et al, 2009).

General introduction to open source

The influence of commercial companies entering the Open Source market 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[[1]](#footnote-2)*. 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.

General introduction to radical and incremental innovation

## 2.3 Aim of Research

The first goal is to identify what kind of innovation takes place in FOSS development. A panel of industry experts will judge 115 FOSS projects on its ‘radicalness’. Furthermore the ‘success’ of each project will be determined. This way I will try to recognize whether FOSS development is more suitable for either incremental or more radical innovation projects.

In order to get a complete picture, the same identification should take place in a closed source or commercial software development. This way it could be determined which development environment is preferred for certain kind of innovation projects. In this research I will focus solely on the FOSS side of the picture. Still, the findings can be interesting for (commercial) software companies who engage in new software development whether or not to get involved in the Open Source movement.

**Radical**

**Incremental**

**FOSS**

**CSS**

?

?

?

?

Figure 2.1 Successful radical / incremental projects in Closed of Free/Open Source software development

# Free and Open Source Software

## 3.1 The Free / Open Source phenomenon

Eric Raymond was the first to describe the Open Source community and its method of writing software in his book “The Cathedral & the Bazaar” (Raymond, 1999). The title is an allegory: proprietary software production as the carefully planned building of a cathedral, Open Source software production as the chaotic interactions of the participants in an oriental bazaar. Strong, centralized management versus loosely related developers organized in several thousand seemingly independent projects.

Open Source is often characterized as a fundamentally new way to develop software (Raymond 1999) that 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 OSS 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.

These differences suggest an extreme case of geographically 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. 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.

### 3.1.1 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).

Crowston and Scozzi (2002) 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”.

### 3.1.2 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[[2]](#footnote-3) 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 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 OS community.

The second hybrid 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 for example, contributed developers to Apache’s Web server. Or corporations release previously closed source code and encourage their employees to work on the now open project. Lerner and Tirole 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, also comes close to the OSSg2 model.

Relevance to my research: 2nd and 3rd model probably most interesting. Firms want to see what kind of project they want to develop in these models.

## 3.2 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. Clearly, there are as many ways to run a FOSS projects as there are projects, but some common threads emerge nevertheless. 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 .[[3]](#footnote-4)*

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.

### 3.2.1 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 (Nakakoji et al., 2002) (see also Figure 3.1).

**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.

### 3.2.2 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 3.1 depicts the general layered structure of FOSS communities, where the role closer to the center has a larger influence (Nakakoji et al., 2002).



Figure 3.1 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. Most open source software is contributed only by a small number of developers (Mockus et al., 2000; O’Reilly, 2002).

### 3.2.3 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[[4]](#footnote-5).

**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 3.2

There are over 180.000 projects registered at sourceforge.net[[5]](#footnote-6). 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, and very often, little thought is given into the repercussions of starting an project. Only 20% of the projects has made it past the point of stable.

## 3.3 FOSS project characteristics

Characteristics of open source projects  
Capiluppi, A.   Lago, P.   Morisio, M.     
Dipt. Automatica e Informatica, Politecnico di Torino, Italy;

From a general standpoint, OS clearly represents a software development model consistent with a less formal approach to governance. OS projects leverage cooperative development, innovation, and informal leadership [40], [47], [80], [99]. OS communities exploit the distributed intelligence of all participants and lack the traditional hierarchical structure and governance roles. These communities promote new coordination mechanisms based on consensus, meritocracy, and the so-called “do-ocracy” [105], i.e., decisions are made by the developers who more actively contribute to the project.

*An Empirical Study on the Relationship among Software Design Quality, Development Effort,*

*and Governance in Open Source Projects*

*Eugenio Capra, Chiara Francalanci, and Francesco Merlo(2008)*

### 3.3.1 Centralization

Social network analyses of bug-fixing interactions was conducted by Crowston and Howison (2006) from three FOSS developer websites: Sourceforge, GNU Savannah and Apache Bugzilla. They found that “the level of centralization is negatively correlated with project size, suggesting that larger projects become more modular, or possibly that becoming more modular is a key to growth”.

<http://floss.syr.edu/publications/ktp2005.pdf>

### 3.3.2 Formalization (formal or informal ties).

### 3.3.3 Complexity

# Radical and Incremental innovation

## 4.1 Definitions of radical and incremental innovation

An innovation is defined as an idea, practice, or material artifact *perceived to be new* by the relevant unit of adoption (Zaltman et al, 1973). Verschil innovativeness en radicalness What their definition did not emphasize was that innovations can vary in the degree of newness to an adopting unit. The notion of radicalness is a way to capture the distribution. Radical and incremental describe different types of technological process innovations. Radical innovations are fundamental changes that represent revolutionary changes in technology. They represent clear departures from existing practice (Ettlie, 1983). In contrast, incremental innovations are minor improvements or simple adjustments in current technology (Munson and Pelz, 1979). The degree of novel technological process content, in other words the degree of new knowledge embedded in the innovation, is the major difference captured by the labels radical and incremental.

Hage (1980) stated that there is a continuum of innovations that range from radical to incremental. An innovation’s placement on this continuum depends upon perceptions of those familiar with the degree of departure of the innovation from the state of knowledge prior to its introduction.

### 4.1.1 Technological discontinuities

Tushman and Anderson (1990) described a cyclical model of technological change (figure 4.1).

Technological discontinuity

Era of incremental change

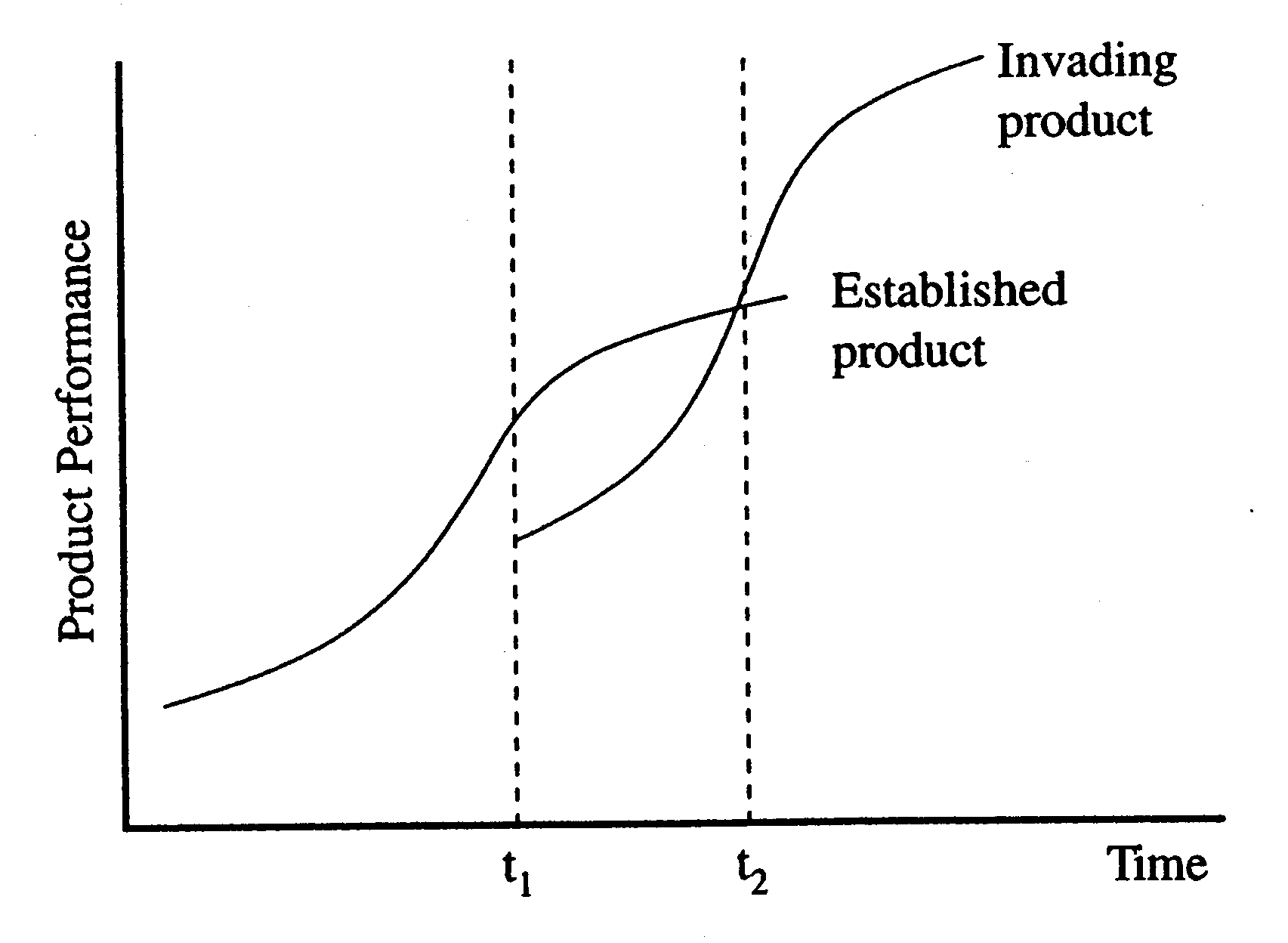
Era of ferment

Dominant design

Figure 4.1

At rare and irregular intervals in every industry, innovations appear that depart dramatically from the norm of continuous incremental innovation that characterizes product classes. These discontinuities either affect underlying processes or the products themselves. The introduction of a radical advance increases variation in a product class. A revolutionary innovation is crude and experimental when introduced, but it ushers in an era of experimentation as organizations struggle to absorb (or destroy) the innovative technology. This era of ferment is characterized by two distinct selection processes: competition between technical regimes and competition within the new technical regime. This period of substantial product-class variation and uncertainty ends with the emergence of a dominant design. A dominant design is a single architecture that establishes dominance in a product class. After a dominant design emerges, technological progress is driven by numerous incremental innovations. Variation now takes the form of elaborating the retained dominant design, not challenging the industry standard with the new, rival architectures. The focus of competition shifts from higher performance to lower cost and to differentiation via minor design variations and strategic positioning tactics.

### 4.1.2 The product life cycle model



Source: Sheets college 8 September.

Tushman and Anderson (1990)

Henderson and Clarck (1990)

### 4.1.2 (Economical) effects of radical and incremental innovation (?)

### 4.1.x Differences radical and incremental innovation

Radical innovation involves the application of significant new technologies or significant new combinations of technologies to new market opportunities and is therefore a major driver for growth (Rice et al, 2001). Radical innovation is defined 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). Radical innovations create such a dramatic change in products, processes or services that they transform existing markets or industries, or create new ones. *Dahlin and Behrens (2005) defined a radical invention as:*

*•novel (dissimilar from previously available inventions),*

*•unique (diverging from current interests of other inventors),*

*•having an impact on future technologies (encouraging imitation).*

## 4.3 Measuring innovation

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.

Jordan and Segelod (2006) state that software innovativeness can be related to several aspects of the product, such as its *features*, *the impression of its newness* and *the novelty of architectural structure*. Uitbreiden

## 4.4 Previous research on innovation in FOSS

Previous research on innovativeness in the FOSS sector was conducted by Klincewicz (2005). He evaluated 500 of the most active projects registered at SourceForge.net. He used theoretical sampling approach to analyze the large scale sample, using “tech mining” software. Klincewicz found relatively *low* levels of technical newness, alongside a *high* interest of developers and users in the innovative projects.

Lorenzi et al. (2008) tried to answer the question whether programs based on FOSS solutions are more innovative than proprietary ones. Using a sample of 134 software solutions produced by Italian firms and an expert panel they found that FOSS solutions seem to be more innovative. Iets verder uitbreiden

Measuring radicallness

## 4.4 Organizational structure and types of innovation

In their 1984 study, Ettlie et al. evaluated a general model of the innovation process in organizations that is differentiated by radical versus the incremental outcomes and found strong support for this theory. Their framework suggests that the strategy-structure causal sequence for radical innovation is different from the strategy-structure sequence for incremental innovation. “Incremental innovation processes that lead to new product introduction appear to be dependent on more traditional structural arrangements and market oriented strategies.” Structural effects are persistent when the other variables including size are controlled. Even the more traditional structures are differentiated by innovation type. “*Centralization* and *informal structures* tend to support radical process adoption, which suggests that regardless of size, organizations match their structure for the innovating situation. A *market dominated growth strategy* tends to reinforce the structural arrangements for incremental innovation-complexity, *decentralization* and *formalization*.”

<http://www.jstor.org/stable/pdfplus/2631748.pdf>

Approaches studying the degree of radicalness of innovation attempt to determine the organizational characteristics that predict radical and incremental innovation (Damanpour and Gopalakrishnan, 1998).

Dimensions:

Centralization

Formalization

Complexity

Samenvattend:

FOSS: hoe groter het project, hoe decentraler de hierarchie wordt > geschikter voor incrementele innovatie.

*Hypothesis 1:*

The higher the developer count on a FOSS project is, the more likely it is to induce incremental innovation.

Hypothesis 2:

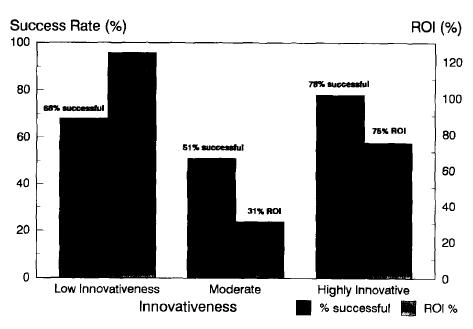
The FOSS environment is better suitable for incremental innovation.

## 4.5 Innovation and success rate

More innovative products should create more opportunities for differentiation and competitive advantage, hence impact positively on performance.

Conversely, less innovative products are more familiar, less uncertain, may have higher synergies, and hence have a higher success rate.

**Success** Rate (%)



**Figure 2. Impact of innovativeness on profitability.**

**ROls significantly different (ANOVAs;**

**Duncan Multiple Range Tests). Success rates**

**significantly different (t-test on means).**

<http://www.sciencedirect.com/science?_ob=MiamiImageURL&_imagekey=B6VD5-45YDN5G-1K-1&_cdi=5973&_user=499884&_check=y&_orig=search&_coverDate=12%2F31%2F1991&view=c&wchp=dGLzVtb-zSkzV&md5=56562e59398337254149bcdadc4dedb7&ie=/sdarticle.pdf>

# Data and Methodology

## 5.1 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.

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 3.2.3 *‘Project lifecycle’*). This list was reviewed by two FOSS experts who both pointed out that some of the larger FOSS projects 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.

## 5.2 Methodology

A panel of 4 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*.

**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?

**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.

**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, etc?

[Scores bij elkaar optellen]

The final score which defines the *radicalness* for each project, will be the average score of the first three questions. So this score will be a number between 1 and 5.

**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)*.

## 5.3 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.

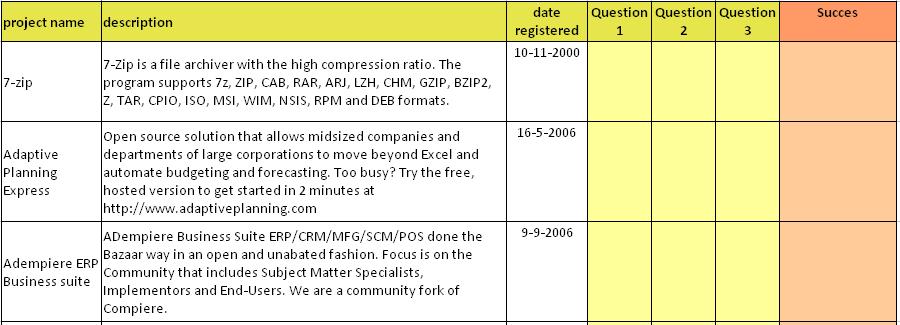


Figure 5. 5‑1

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 Open Source since 10 years.

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

# Empirical Results

Number of projects: **115**

Average project radicalness: **3.07**

Average project success: **1.74**

Number of projects rated ‘success = 1.0’: **54** Median radicalness: **2.73**

Number of projects rated ‘success = 1.5’: **6** Median radicalness: **2.50**

Number of projects rated ‘success = 2.0’: **24** Median radicalness: **3.00**

Number of projects rated ‘success = 2.5’: **8** Median radicalness: **3.72**

Number of projects rated ‘success = 3.0’: **23** Median radicalness: **4.00**

# Conclusions

# SimonsLeversOfControl

# References

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

Anderson, P. and Tushman, M. (1990). “Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change”, Administrative Science Quarterly, edition 35(4): p. 604-633.

Castelluccio, M. (2008) “Enterprise Open Source Adoption” *Strategic Finance,*  edition *90* (5),p.  57-58.  Retrieved April 10, 2009, from ABI/INFORM Global database. (Document ID: 1596941681).

Crowston, K. and Howison, J. (2006), “Hierarchy and Centralization in Free and Open Source Software Team Communications”, Knowledge, Technology, and Policy, edition 18(4): p. 65-85.

Crowston, K. and Scozzi, B. (2002), “Open Source Software Projects as Virtual Organizations: Competencty Rallying for Software Development”, *IEE Proceedings Software*, 149, p. 3-17.

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: p. 85-95.

Ettlie, J. (1983), “Organizatioal Policy and Innovation Among Suppliers to the Food Processing Sector”, *Acedemic Management Journal*, edition 26, p. 187-224.

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: p. 263-272.

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’Reilly, T. (1999), “Lessons from Open-Source Software Development”, *Communications of the ACM*, edition 42, p. 33-37.

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).

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.

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: p. 459–468.

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.

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

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

1. www.fsf.org [↑](#footnote-ref-2)
2. A major Linux distribution vendor, http://www.redhat.com/ [↑](#footnote-ref-3)
3. www.opensource.org/docs/osd 19-3-2009 [↑](#footnote-ref-4)
4. Derived from the sourceforge.net database [↑](#footnote-ref-5)
5. Information retrieved from http://ossmole.sourceforge.net/, 22-04-2009. [↑](#footnote-ref-6)