# Abstract

Table of contents

[1. Abstract 1](#_Toc232830964)

[2. Introduction 2](#_Toc232830965)

[2.1 Motivation 2](#_Toc232830966)

[2.2 Problem Field 2](#_Toc232830967)

[2.3 Aim of Research 3](#_Toc232830968)

[3. Free and Open Source Software 3](#_Toc232830969)

[3.1 The Free / Open Source phenomenon 3](#_Toc232830970)

[3.1.1 FOSS developer motivation 4](#_Toc232830971)

[3.1.2 Business models, Involvement of commercial companies 4](#_Toc232830972)

[3.2 FOSS Projects 5](#_Toc232830973)

[3.2.1 Roles of Project Members 5](#_Toc232830974)

[3.2.2 Project/Community Structure 6](#_Toc232830975)

[3.2.3 Project lifecycle 7](#_Toc232830976)

[3.3 FOSS project characteristics 8](#_Toc232830977)

[3.3.1 Centralization 8](#_Toc232830978)

[3.3.2 Formalization (formal or informal ties). 8](#_Toc232830979)

[3.3.3 Complexity 8](#_Toc232830980)

[4. Radical and Incremental innovation 9](#_Toc232830981)

[4.1 Introduction 9](#_Toc232830982)

[4.2 Importance of Radical and Incremental innovation projects 10](#_Toc232830983)

[4.3 Management of radical innovation projects 11](#_Toc232830984)

[4.3.1 Management of incremental innovation 11](#_Toc232830985)

[4.3.2 Management of Radical Innovation 12](#_Toc232830986)

[4.3.3 Managing project risk 14](#_Toc232830987)

[4.3 Nature of radical innovations 15](#_Toc232830988)

[4.3.1 Grote / kleine investeringen 16](#_Toc232830989)

[4.3.2 Veel / weinig tijd 16](#_Toc232830990)

[4.3.3 Crossfunctioneel / autonoom team 16](#_Toc232830991)

[4.3.4 Measurement of radicalness 16](#_Toc232830992)

[4.4 Software development 18](#_Toc232830993)

[4 Data and Methodology 23](#_Toc232830994)

[5.1 Data 23](#_Toc232830995)

[5.2 Methodology 23](#_Toc232830996)

[5.3 Industry Experts 24](#_Toc232830997)

[5 Empirical Results 24](#_Toc232830998)

[6 Conclusions 26](#_Toc232830999)

[7 References 26](#_Toc232831000)

# 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 beta.

## 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

Beperk grotendeels tot management van innovatie projecten

## 4.1 Introduction

In order to define what a radical and an incremental innovation is, it is useful to first define what a ‘technological innovation’ is. Engineering, marketing, management and even economics provide unique spins as to what is considered an innovation (Garcia and Calantone, 2002). A review of this literature reveals that the 1991 OECD (OECD, 1991) study on technological innovations best 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’.

This definition addresses two important distinctions (Garcia and Calantone, 2002):

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, and

2. the innovation process is *iterative* in nature and thus, automatically includes the first introduction of a new innovation and the reintroduction of an improved innovation. 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. ‘Highly innovative’ products are seen as having a high degree of newness and ‘low innovative’ products sit at the opposite extreme of the continuum (Garcia and Calantone, 2002). However, 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’Conner, 1998), new to the market (Kleinschmidt and Cooper, 1991), and new to the consumer (Atuahene-Gima, 1995).

Despite the varying perspectives for ‘innovativeness’ a single consistency does exist; it is always modeled as the degree of discontinuity in marketing and/or technological factors.

Newness to Market

|  |  |  |
| --- | --- | --- |
|  | High | Low |
| High  Technological  Newness | Radical  innovation |  |
| Low |  | Incremental  Innovation |

The notion of *radicalness* is a way to capture the degree 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). Utterback (1996) provides the following definition of a discontinuous or radical innovation: “By discontinuous change or radical innovation, I mean change that sweeps away much of a firm’s existing investment in technical skills and knowledge, designs, production technique, plant and equipment” [p. 200]. 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.

**Invoegen market en technology uncertainty (met figuur)**

## 4.2 Importance of Radical and Incremental innovation projects

Intro

### 4.2.1 Importance of Incremental innovation projects

Less innovative products are more familiar, less uncertain, may have higher synergies, and hence have a higher success rate (Kleinschidt and Cooper, 1991).

### Add more

### 4.2.2 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).

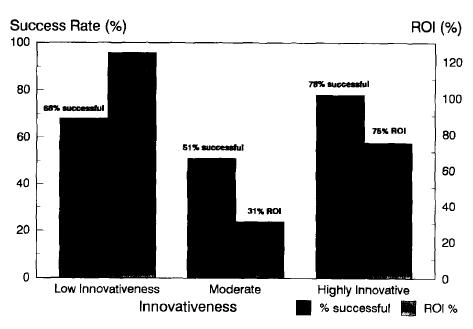
According to Song and Montoya-Weiss (1998) “really new products can often create significant opportunities for differentiation and competitive advantage”. However, the investments that are necessary for radical innovation projects are much higher (Kleinschmidt and Cooper, 1991) and the risks of projects not making it to commercialization are higher than incremental projects (Cooper, 2003). So the risk of radical innovations is much higher but the potential rewards, in term of new product success are also higher.

Watson (2006): 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. > Despite high risks, it pays to invest in radical innovation projects. Elaborate more (what industry measures)

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.

Kleinschmidt and cooper found a U-shape beschrijven.

**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).**

**Portfolio management**

## 4.3 Management of radical innovation projects

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.

### 4.3.1 Management of Incremental innovation

I will mainly use the work of Ernst (2002) as a basis to *describe* 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 for each of these categories.

#### 4.3.1.1 NPD Process

Of specific importance for the success of new products 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 (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 Stage Gate Model (1990, 2002) 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.

#### 4.3.1.2 Organization

Cross-functional team within the organization

Add more

#### 4.3.1.3 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.

#### 4.3.1.4 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**

#### 4.3.1.5 Culture

**Formal culture?**

### 4.3.2 Management of Radical innovation

As discussed earlier, I will mainly base my this section on the findings in the book of Leifer et al (2002). 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.

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). 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 seven managerial challenges and four types of uncertainty that need to be reduced for radical innovation projects to be successful.

I will first address the seven challenges in managing radical innovation and its competencies required to address the challenge and will separate them in the same categories that Ernst used in his 2002 review. I will add relevant literature findings, when I believe this adds information to the findings of Leifer et al.

#### 4.3.2.1 NPD Process

The first challenge is *to capture radical ideas in the “fuzzy front end”*. The required competencies 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 (?????????). Also the development processes are 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).

#### 4.3.2.2 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.

Autonomous team > heavyweight team?

#### 4.3.2.3 Role and commitment of Senior Management

As addressed earlier the role of a *product champion* plays an important role in radical innovation projects. (ELABORATE)

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. Veryzer (1999) also recognizes the importance of a project or product Champion. 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.

#### 4.3.2.4 Strategy

A challenge is *to resolve uncertainty in the business model* (Leifer et al, 2002). 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 identified. Resource acquisition along with establishment and management of internal and externals partnerships are competencies required to address this challenge.

**Portfolio management**

#### 4.3.2.5 Culture

*Managing the transition from radical innovation projects to an operation status*. Required competencies are: accurate assessment of the transition readiness of the project and the receiving unit; development of people, practices an structures for successful transitions; and the ability to build bridges between organizational units.

Also *engaging individual initiative* was identified by Leifer et al. This can be encouraged by effectively define 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.

Furthermore McDermott (1999) stated that *informal networks* inside and outside the company are important to stimulate radical innovation.

### 4.3.3 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, 2002). 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.

#### 4.3.3.1 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 4.? 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.

#### 4.3.3.2 Reduce organization and resources uncertainty

Leifer et al (2002) 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.

MORE?

Conclusion

A lot of research has been done to recognize success factors in incremental NPD. 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. 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.

## 4.3 Nature of radical innovations

|  |  |  |
| --- | --- | --- |
|  | 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. |

### 4.3.1 Grote / kleine investeringen

### 4.3.2 Veel / weinig tijd

### 4.3.3 Crossfunctioneel / autonoom team

(Wheelwright and Clarck (1992).

Concluderen: conclusies literatuur waarschijnlijk niet direct van toepassing op OSSD.

### 4.3.4 Measurement of radicalness

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*.

As the measurement of radicalness in OSS fails in traditional methods, I have come up with a more practical solution for my research. Concluderen of mijn manier van meten wel of niet overeen komt met literatuur, waarom?

## 4.4 Software development

|  |  |  |  |
| --- | --- | --- | --- |
|  | Incremental | Radical | Software development |
| Project time line | Short-term – six months to two years. | Long term – usually ten years or more | Usually 6 months to two years before first stable release. Development of updates / improvements and new versions can last for years after release. |
| 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. | At the front end. In OSS ideas are mainly generated or published on OS for a just before they are launched. |
| 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. | Best known in software development are the ‘Waterfall model’ and the ‘Iterative model’. Both are rather formal processes in which tasks and functions can be assigned on forehand. |
| 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. | In commercial software development a detailed plan can be developed at the beginning. A typical OSS project lacks a business plan. |
| 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. | In commercial development a typical project is assigned to a lightweight or heavyweight project team where each member has his responsibilities clearly specified.  Is OSS developers are users. Core members are key players and are essential for the success of the project. They can act cross-functional and one could say they act as Champions. Other players are attracted and can leave throughout the project |
| 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. | Typically a light or heavyweight team is assigned in CSD.  Although often there is no formal structure in OSSD, there is some sort of hierarchy where core members play an important role. |
| 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. | In OSS not all necessary competencies are there at the beginning of the project. Competency resources can come and go during the project. Having the right competencies is critical for the success of the project. |

To meet the challenge of reducing the costs of producing complex software, many companies adopted structured approaches to software development. Cusumano’s study of the “software factory” documents how software design moved from art to routinized tasks manipulating standardized modules (Cusumano, 1991). This approach culminated in an attempt to rationalize the entire cycle of software production, installation, and maintenance through the establishment of

factory-like procedures and processes.

The factory production process is not, however, well suited to all software design processes. Glass’ (1995) view is that software is a creative enterprise that cannot be fully routinized. Methodologies to convert design into a disciplined activity are not suited to addressing new problems to be solved (1995: 41). At the same time, writing of code involves solving the detail-level problems left unsolved in an inevitably incomplete design.

The factory approach to software development applies the Babbage principle of the mental division of labor. In this model, intelligent work is specialized to the design group, code writing is given to a less skilled group, and debugging and maintenance to an even less skilled group.

The interactive approach suggests a production function in which value is maximized, subject to the constraints of threshold quality and time to market. This process will be less structured than a “waterfall” sequence where the design stage precedes coding and testing, but will allow for concurrent design and implementation.

This model suggests that the software production is as good as its most productive member. It is in this sense that open source exploits the intelligence in the community; it provides a matching between competence and task. Open source development permits this resolution of complexity by consistently applying the principles of modular design.

Source: Kogut and Metiu (2001)

Many ideas originate from users, not producers (Von Hippel, 1998).

OSS: concurrent debugging and design, CSS releases beta versions.

Reduce complexity by partitioning the development process into discrete steps.

Agile methods have received positive attention in the software engineering community. Their focus on simplicity, programmers, and products empowers the technical staff and gives a sense of focus on product development basics. Researchers have investigated agile projects in isolation, but not in the traditional stage-gate project management environments where they are often applied. On the basis of interviews and archival analyses, case studies for three different organizations using agile methods showed their feasibility, despite some initial management resistance. The agile methods provided continuous feedback. They give the stage-gate model powerful tools for microplanning, day-to-day work control, and progress reporting. The stage-gate model, in turn, gives agile methods a means to coordinate with other development teams and communicate with functions such as marketing and senior management.

The latest CHAOS study found that application development project success rates continue to improve. The 2004 study reported project success rates of 34 percent (Software Magazine 2004), as compared to 28 percent in 2000, and 16 percent in 1994 (Johnson et al. 2001).

Of course, organizations have implemented the stage gate concept using various life-cycle designs. Just as in IT, product development gurus and practitioners have debated for years the respective merits of waterfall vs. iterative and/or rapid development life-cycle models, including concurrent engineering. Though some critics of Cooper’s stage gate process point to inflexibility of the process, the potential of excessive management control, and negative impact on the speed to market, the approach appears to be effective for many. A benchmarking study conducted by the Product Development Management Association (PDMA), identified the stage gate process as a hallmark practice of many of ‘the best’ performing companies (Griffin 1997).

2004, Bernice L. Rocque and Walter A. Viali At the Stage Gate: Critical Questions for IT Project Sponsors

Originally published as part of 2004 PMI Global Congress Proceedings . Anaheim, California

**Radical**

**Incremental**

**FOSS**

**CSS**

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

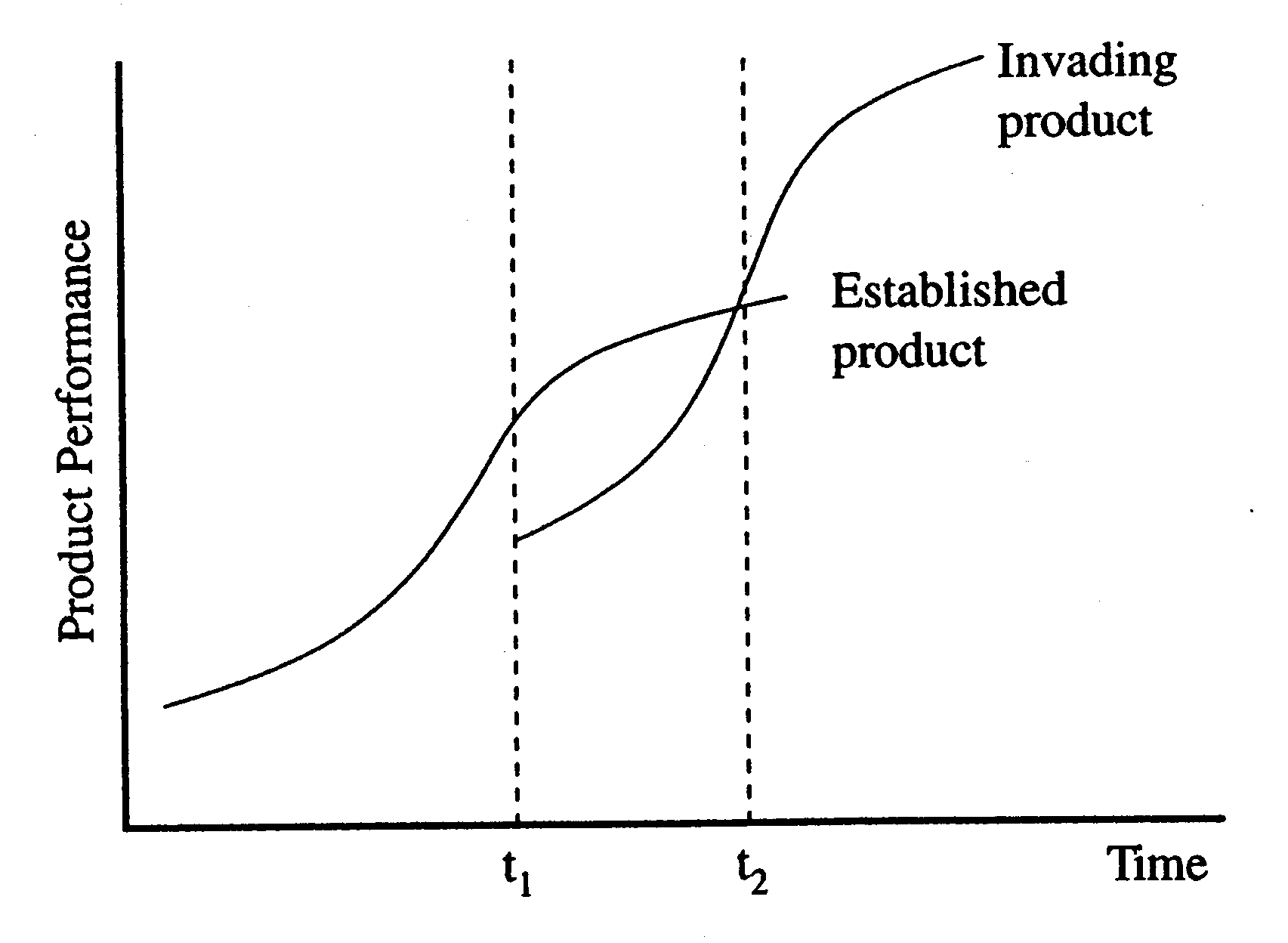
Era of ferment

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

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).

# 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*. USE MARKET AND TECHNICAL NEWNESS + WHY ON MARKET LEVEL

**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.

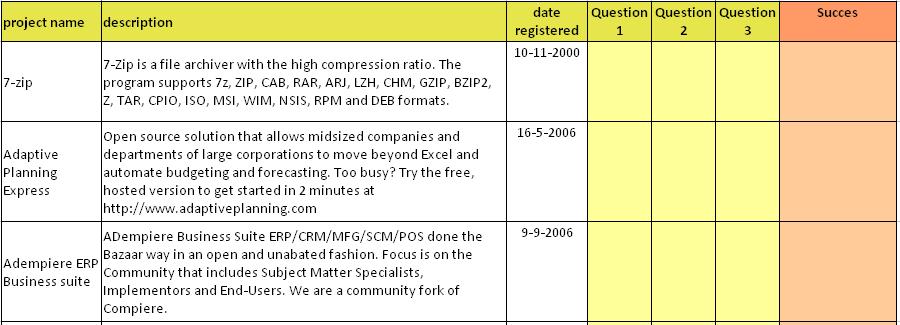


Figure 5. 4‑1

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 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**

**Market newness vs technological newness**

## 5.x 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 1 to several thousand developers for the really large projects. In figure 5.x is shown how many projects there are based on the 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. CONCLUSIE

# 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.

OECD (1991), *The nature of innovation and the evolution of the productive system*, technology and productivity-the challenge for economic policy, Paris: OECD, p. 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*, May/Jun 1998, edition 41 (3).

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)