HAAGA-HELIA
University of Applied Sciences

Methodologies Used in Open Source Approach to Developing Software in Companies in Finland

Felicia Gbemisola Jääskeläinen

Bachelor's Thesis
Business Information Technology
November 2013

Abstract

Date of presentation

Degree programme

Author or authors Felicia Gbemisola Jääskeläinen	**Group or year of entry** 2013
Title of report Methodologies Used in Open Source Approach to Developing Software in Companies in Finland	**Number of report pages and attachment pages** 70 + 1
Teacher(s) or supervisor(s) Tuomo Ryynänen	

There has always been methodology since when there has been a need to create solutions to problems. In the case of open source software development, methdology has been essential even though not very visible in creating successful development processes. The Linux and Apache projects are but a few of success stories of open source development projects.

This research project sets out to gain knowledge on methodologies used in open source software development and its theoretical background digs into understanding software development processes and open source holistically. The theoretical background opens up also the factors affecting open source software. These background information is then taken into the empirical phase to establish knowledge in this subject. The result of this research brings to light the current trends in methodologies in use in companies developing open source software's in Finland. All the companies interviewed, uses a methdology either as it is or crafted in combination with another methodology to create a new methdology that functions for the company and brings benefit to the customer too.

The results from this research will be useful for any company implementing open source software development, any company planning / individual interested in methodologies used in open source software development and any company wanting to explore new methdology.

Keywords
Methodology, Open source software, methodology processes, development

Table of contents

Glossary of Terms

OS	Open Source
FS	Free Source: unfettered access to source codes/ free is equal to freedom, not price.
OSS	Open Source Software: unlimited access to source code
FSS	Free Source Software
FLOSS	Free/Libre and Open source Software
GNU	GNU's Not Unix: is a computer system composed entirely of free software.
GNU GPL	GNU General Public License: the most common OSS/FSS license that gives the recipient the free software definition rights.
OSD	Open Source Definition
OSI	Open Source Initiative: are the stewards of the OSD and the community recognized body for reviewing and approving licenses as OSD-conformant.
SF	Source forge: The most commonly used online community with over 2million users for creating open source software development projects and distribution.
OSL	Open Source Licenses: are licenses that comply with the open source definition.
OSSDP	Open source software development process.

1 Introduction

There is a saying that the most contagious resilient parasite is not a virus but an idea. An idea to change the world, an idea to practice freedom and in the case of software development an idea to exercise freedom in source codes development. The software market has been far dominated by proprietary software produces but newer studies and researches shows that open source software's are now gaining sub-stantial popularity in the software market. Newer studies also shows that governments and institutes around the world are adopting open source as an alternative over propri-etary software; countries like France, United Kingdom, Norway, Brazil, United States, even developing countries like India are favoring OS software (Weerawarana & Weeratunga. 2004, 27-36). According to the software industry survey 2011, OS software has a significant role in the Finnish software industry, especially between 2008 and 2010 period (Mikko Rönkkö et al. 2011, 35).

Software development project is a highly collaborative activity that requires teams of developers to continually manage and coordinate their programming tasks (Christoph Treude et.al. 2008, 1). What makes this research even more astounding is the idea that source code may be freely produced, modified and redistributed by loosely organ-ised, ad-hoc communities of programmers from all over the world who may have never seen face-to-face but share a sense of commitment and produce powerful software that are shifting the way software are developed (Eugene Eric Kim,2003, 1).

Developing software entails a great deal of management skills, methods, timing and now considering an IT project making use of open source opportunities; the project management ideology would certainly might be vast and differently approached especially in small open source development project; thus the thesis project title: Methodologies Used in Open Source Approach to Developing Software in Companies in Finland.

1.1 Background

Software development methodologies has evolved over a period of 50 years and within these years there has been factors that have shaped the methodologies to what we have today. Rico stated that there are 32 major classes of software methods that have emerged over the last 50 years. The need to re-invent methodologies to better the way work is done has today shaped the way IT projects are been carried out; thus leading to the evolution of software development methodologies. Factors propelling the shift to more efficient methodologies are: the Internet era, globalisation, recent global depression, the need to create faster and better software's, computers becoming personal, social media and the computer-human communication psychology (Rico, 2005, 1-15).

According to Audris M, et al. (2002, 2), for open source projects, it is said that its development processes are radically difference from the proprietary methods of software development and the main differences are:

- Complex OSSs are built by a large number of developers around the world with a common norm of openness; some of these developers are volunteers while others are supported by companies around the world.
- The workload usually is not assigned to the developers instead the developers choose to undertake whatever work they choose to perform
- There are usually no detailed requirements, design, project plan nor schedule

The author of this thesis has during her studies participated in an agile project using Scrum in developing an application and from then has been interested in software development methodologies. This interest has also driven further studies into IT project management methods, evolution and the research project will cover major software development methodologies widely in OS.

2 Research

When trying to collect data that will give insight into how processes are been generated and used in software project development, the most suitable research design is qualitative. This is so because the researcher hopes to discover how "things are been done" systematically in an organisation (Mark Saunders, et. al, 2012, 171). This research is a descriptive research due to the nature of the research questions and anticipated outcome and since the object of a descriptive research is to gain an accurate profile of events, persons or situation the research is designed as such.

Qualitative research is associated with an interpretive philosophy. It is interpretive because researchers need to make sense of the subjective and socially constructed meanings expressed about the phenomenon being studied (Denzin & Lincoln 2005, 3). Extensive theoretical review of available literatures on the subject matters will be conducted to create a solid background that adheres to the research purpose and answers the research questions.

2.1 Research Purpose and Question

As the title of the project implies, the purpose of this research is to describe methodologies used for open source software development projects in Finland and the term companies in this project refer to OS companies with employee size of 5- 100 employees. The project will focus on identifying and analysing processes and practises in use by the selected companies when developing OS software projects.

The research findings from this project are envisioned to become applicable to identical companies in Finland, can also be useful to other business organizations and any individual interested in OS software. In order to be able to achieve the project goal, the following research questions were crafted:

- What methodologies are used for open source software development in small companies in Finland?
  - Why were these methodologies chosen?
  - How are these methodologies implemented?
- What are the strengths/challenges of developing open source software?

2.2 Scope

The projects scope will be limited to companies with employees of 6 -100 employees, companies will be open source software developing companies and must be operating in Finland.

The thesis project will not be including companies offering services as their major business focus. This project will not be comparing methodologies nor give recommendations as to what methodology is better than the other.

2.3 Research Strategy

The research's theoretical background will cover every aspect concerning open source, extensive information background on software project methodologies and its history and current issues in software development projects.
Theoretical framework is based on existing literature works and these literature works includes relevant articles, blog posts, books, news report, dissertations, publications, interviews, thesis report and journals. These literature works will be analysed and processed to build the theoretical framework for this research and Interview is selected as the research strategy to connect the theoretical framework to the data collection process.
In general, interviews are discussions between an interviewer and pre-selected entity to collect information on a specific set of topics (Harrell & Bradley 2009, 30).). For the purpose of this research, semi-structured interview has been selected as the form of interview best suited for the research.

3 Software Development Methodologies

Methodology in software development is a terminology that umbrella other several terms like method, model, framework and processes and according to Merriam-Webster dictionary; “methodology: 1. is a body of methods, rules, postulates employed by a discipline: a particular procedure or set of procedures. 2. The analysis of the principles or procedures of inquiry in a particular field. Methodology could also mean an approach, a model, framework or a structure on how to accomplish a significant goal” (merriam-webster.com).

Then it can be said that; software development methodology is a structure imposed on the development of a software product; it includes procedures, techniques, tools and documentation aids which will help system developers in their task of implementing a new system and the intent of a methodology is to formalize what is being done and making it more repeatable. As humans, there is always a need to better the way we function especially now that our world have been globalized so is our views on software development methodologies and the following chapters will elaborate more on this subject (Hamid Faridani 2011, 4).

3.1 Common Methodologies

After much research, the study discovered that there are more than 70 different types of software development methodologies, some heavyweight and others light-weighted. For this study, only 5 most common methodologies are chosen and they are:

- The waterfall model
- The Rational Unified Process Model (RUP)
- The Agile Model
    - Scrum
- Open Source Development model
- The Cathedral and the Bazaar

3.1.1 The waterfall Model

According to waterfall-model.com, "waterfall is called as such because the model develop systematically from one phase to other in a downward fashion, like a waterfall and the most probable phases through which it progresses downwards is:

1. Definition Study/Analysis
2. Design
3. Implementation
4. Verification
5. Testing
6. Installation
7. Maintenance

Nowadays the waterfall model is generally taken to mean any sequential model divided into consecutive stages and having the attributes of the original model (Waterfall model.com 2013).

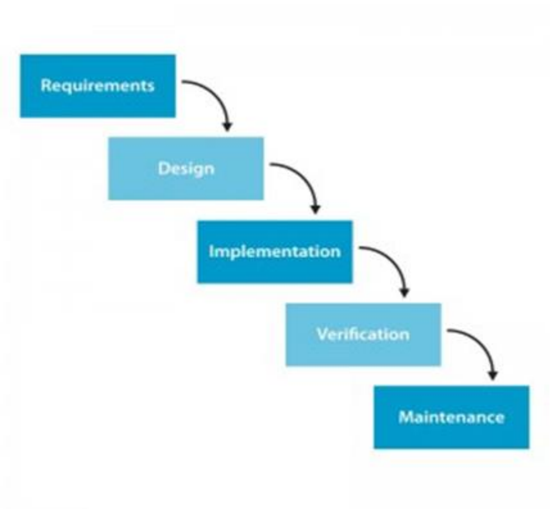

Figure 1: The Waterfall Model

As in Figure 1shows, the model runs from one stage to the other, its flows downward and never backwards. The identification and naming of the stages are not fixed and can be modified to suit particular project characteristics (Cadel & Yeates 2001, 51). This model was developed in the early 1970 and was very much welcomed and taken into use, and is still in use even up till date even though it has been condemned to be inappropriate for some projects and too vast need for documentation. Since the introduction of this model, there has been various modified version of this models. Models like:

Sashimi Model, Aorta Lifecycle model and V Waterfall model are just a few of its modification (waterfall-model.com).

3.1.2 Rational Unified Process Model (RUP)

This is a very iterative software development approach that takes into account the need to accommodate change and adaptability during the development process; that is; a software product is designed and built in a succession of incremental iterations. (Khan et al 2011, 442). The RUP framework shown in Figure 2 depicted as a "hump chart" with the matrix showing the RUP phases and the RUP disciplines.
There are two dimensions for understanding and implementing RUP and they are dynamic or time dimension and static or content dimension as shown in Figure 2 and Figure 3 below.

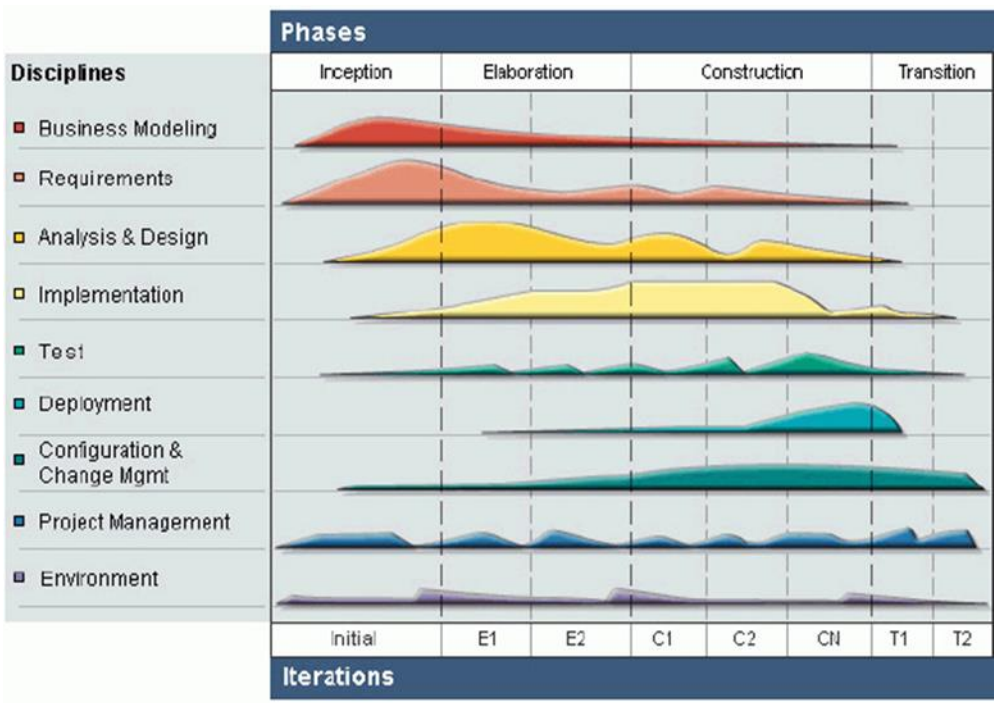

Figure 2: RUP Phases Approach

The dynamic dimension (horizontal) deals with the lifecycle of a project which expresses cycles, phases, iterations and milestones. The lifecycle divides a project into four phases: Inception, Elaboration, Construction and Transition. These lifecycle phases all have objectives as iterations as well as milestones that should be reached at the end of each phase (Per Kroll & Philippe Kruchten 2008, 14).

The content dimension (vertical) expresses how process elements (activities, disciplines, artefacts and roles) are logically grouped into core process disciplines or workflows. A process describes who is doing what, how and when. (Per Kroll & Philippe Kruchten 2008, 14).

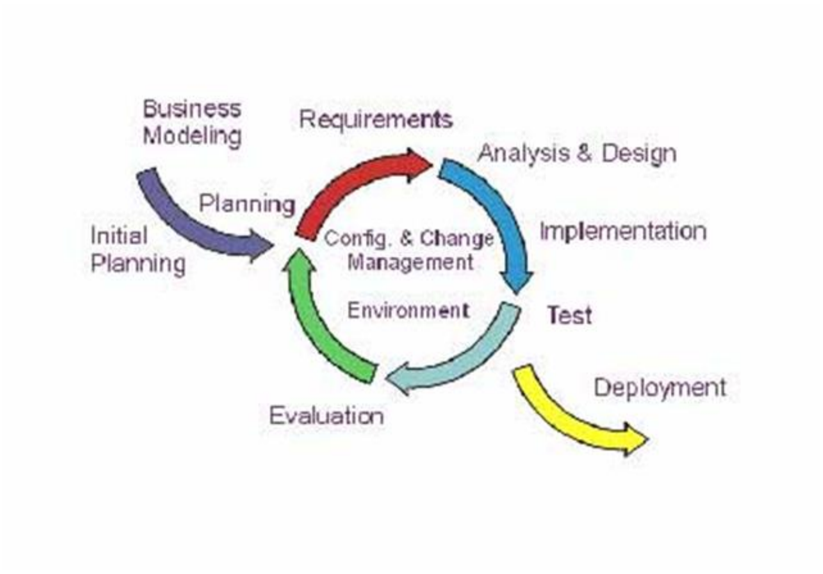

Figure 3: RUP content dimension http://medlem.spray.se/perlin27/rup.htm

The Figure 3 indicates the RUP content dimension; which shows clearly the different roles available in a project and these roles are grouped and exist as workflows. These workflows entails activities that must be carried out to produce artefacts in the right order and iteratively. From the figure the outer sphere consist of the core "engineering" workflows while the inner sphere consists of the core "supporting" workflows. These processes together transposes the incremental and iterative nature of RUP, the vastness and circumstantiates of this model can be as an advantage only if fully adopted and fulfilled its governing disciples. The governing effective deployment 6 best practices according to Rational Unified Process (1998) are:

- Develop iteratively
- Manage requirements
- Use component-based architectures
- Model visually
- Continuously verify quality
- Control change

The Rational Unified Process above six principles to implementing successful projects were developed from Rational's experience while developing large and complex systems; they were also designed to help drive the use of tools offered in Rational's product line but these days Rational is now IBM. The designers of the RUP continue to evolve the process as methods and practices mature through their application. (Pavan Kumar Gorakavi 2009).

RUP is valuable because it embodies many of the best project management practices available and so RUP can be said to be a well-defined and well-structured software engineering process (Charvat, J 2003, 68). It clearly defines who is responsible for what, how things are done and when to do them. The RUP also provides a well-defined structure for the lifecycle of a RUP project, clearly articulating essential milestones and decision points. (Per Kroll & Philippe Kruchten 2008, 3).

3.1.3 Agile

Of all software development methodologies, Agile is the most latest and extreme favoured by the business world. Agile was given birth to in 2001 by 17 software developers' work that became a manifesto. Agile is been said to not only function in the ac-accommodating the business challenges of today's software development it is also said to give to those practising it.
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation; and
- Responding to change over following a plan

And these ideas are in bold because according to the ideology behind the manifesto "that is, while there is value in the items on the right, we value the items on the left more". That is to say, the manifesto recognizes individuals and interaction, working software, customer collaboration and responding to change as invaluable in software development (agilemanifesto.org).

The twelve principles of agile software are:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.

11. The best architectures, requirements, and designs
    emerge from self-organizing teams.
12. At regular intervals, the team reflects on how
    to become more effective, then tunes and adjusts
    its behavior accordingly.

Agile Manifesto http://agilemanifesto.org/principles.html

Agile acts as a framework umbrella for models, structures and methodologies that functions under this principles, different methodologies have risen; methodologies like Scrum; Crystal methodology, Rapid application development (RAD), Kanban and Extreme Programming (XP) and so on. The agile methodologies have been considered as purely lightweight and around the world, very highly spoken of. One of the most popular of all agile is Scrum.

3.1.4 Scrum

According to scrumalliance.org; scrum is a simple yet incredibly powerful set of principles and practices that help teams deliver products in short cycles, enabling fast feedback, continual improvement and rapid adaptation to change (scrumalliance.org). According to Jeff Sutherland 2012, in 2011 Scrum is used in over 75% of agile implementations worldwide. Scrum is not seen as a development method or a formal process, it is rather seen as a compression algorithm for worldwide best practices observed in over 50 year of soft-ware development (Jeff Sutherland, 2012 5). The initial idea behind Scrum was first presented by Takeuchi and Nonaka in 1986 in their white paper "The New New Product Development Game" and in 1995 Sutherland and Schwaber fine-tuned it to what we have today as Scrum. (Ken Schwaber & Mike Beedle 2001).

The following figure explains how scrum functions in a development process. The figure depicts terminologies that are specific to scrum implementation and accordingly have three roles, three ceremonies and three artefacts.
Roles: Product owner, Scrum master, team
Ceremonies: Sprint planning, Daily scrum, sprint review

Artefacts: Product backlog, sprint backlog, burn down chart (Jeff Sutherland 2012, 73).

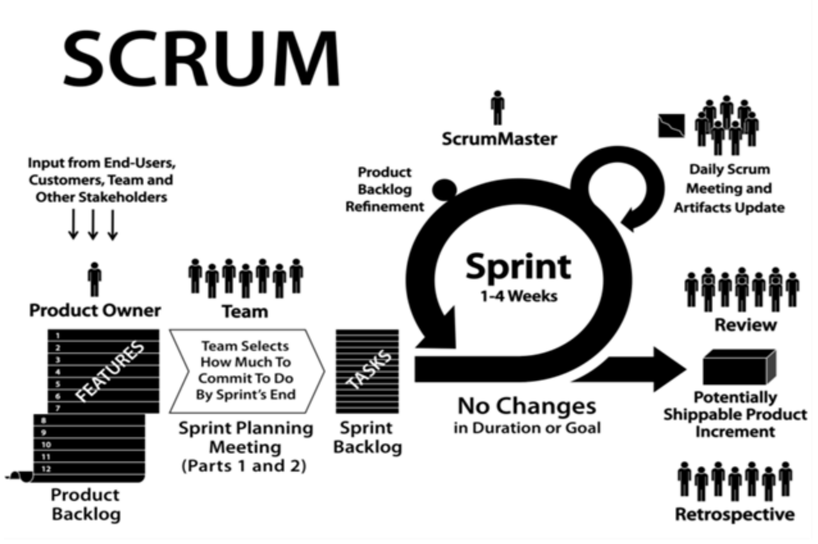
SCRUM
Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
FEATURES
Product Backlog
Team
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting (Parts 1 and 2)
TASKS
Sprint Backlog
Product Backlog Refinement
ScrumMaster
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective

Figure 5: Scrum http://jeffsutherland.com/ScrumPapers.pdf

In the Scrum model, the first activity of each sprint is a sprint planning meeting. During this meeting, the product owner and team talk about the highest priority items on the scrum product backlog. Team members figure out how many items they can commit to, and then create a sprint backlog, which is a list of the tasks to perform during the sprint.

This agile process methodology states that on each day of the Scrum sprint, a daily Scrum meeting is attended by all team members, including the Scrum Master and the product owner. This meeting is time boxed to no more than 15 minutes. During that time, team members share what they worked on the prior day, will work on today, and identify any impediments to progress. Daily scrums serve to synchronize the work of team members as they discuss the work of the sprint.

At the end of a Scrum sprint, the team conducts a sprint review. During the sprint review, the team demonstrates the functionality added during the sprint. The goal of this meeting is to get feedback from the product owner or any users or other stakeholders who have been invited to the review. This feedback may result in changes to the freshly delivered functionality. But it may just as likely result in revising or adding items to the Scrum product backlog.

Another activity performed at the end of each sprint is the sprint retrospective. The whole team participates in this meeting, including the Scrum Master and product owner. The meeting is an opportunity to reflect on the sprint that is ending and identify opportunities to improve in the new sprint (mountaingoatsoftware.com).

3.1.5 Open Source Development Model

The model is said to be a fluid development process characterized by increased intra-team collaboration, continuous integration and testing and greater end-user involvement; this model describes a process and characteristics that would apply to most open source projects but also states that every open source project could have its own development process. (Ibrahim Haddad 2011, 1). This model is developed by Ibrahim Haddad for the Linux foundation. The model presumes that development is apportioned among many teams, working around the world in a mobile structure that is not affected by new arrivals or departures. The model runs under the Feature Development life-cycle and this process consists of:

1. Feature request process
2. Architecture and Design Discussion
3. Collaboration on Implementation
4. Source code submission
5. Continuous testing and Integration
6. Source code release

When there is a need for a new feature, a feature request is create to ensure a common understanding within the development team of what feature has been requested, its priority, development status, associated bugs, blockers and scheduled release. The project team contributors evaluate the request in terms of its need, strength and if it is fit for a future release. This feature request must be communicated and transparent to all team members whether new or old member. The importance of communication comes under the architecture and design discussion phase and the most commonly used form of communication is mailing list, Internet relay chat (IRC) client for design meetings and user support meetings taking into consideration that English might not be the primary spoken language of all participant (Ibrahim Haddad 2011, 4).

After a sustainable communication channel has been created, then a collaborative development phase can ensue. Putting in mind that these team members can be from around the world, a project tool that support effectively code contribution must be involved for example the open source git repository. For a source code to be accepted, it has a life-cycle of its own which is often iterative in nature. The process begins with collaborative development among the team's subset of developers who have taken the responsibility for developing the feature. When the code is functional, it is submitted to the project advocator over the mailing list that then together with other project participant may give feedback on the feature and decide if it can be accepted into the source code of the main software. The smaller the patch that is submitted at an iteration the easier it is to finding bugs and fixes unintended consequences in the source code (Ibrahim Haddad 2011, 5).

Most open source project have automated build suite and test suites that evaluate new code immediately after integrations to identify functional issues during active development, which is why small incremental are favourable to ensure that quality of the source code is assured on all levels. For all the feature codes to be well integrated, a release management is put in place as a development tree to make retrieving the most current stable release available to all users to continue work on (Ibrahim Haddad 2011, 7).
Core characteristics of open source development model are: that code contributors are responsible for development and maintenance of code that is; one or more of the project advocators checks and integrates source codes written by contributors into the body of existing codes (Ibrahim Haddad 2011, 8).

3.1.6 Cathedral and the Bazaar

The Cathedral and the Bazaar is based on the understanding of the success story behind the Linux project. Eric Raymond, 1993 made a statement that the success of Linux project overturned much of what he thought he knew about software engineering and has dedicated himself to revealing the dynamical sphere of how open source projects can succeed by testing his new found understanding with an open source project which was successful(Ko Kuwabara, 2000).

The project tested was to create a POP3 similar-like client that has the ability to hack the addresses on fetched mail so that replies can be sent to the right address, thus the following factors. Factors like:

- Find a problem that is interesting to you.
- Delegate everything you can.
- Be open to the point of promiscuity.
- Get users that act also as co-developers.
- Release early and often.

These building blocks are seen as the basis to a successful bazaar open source project. The Cathedral is seen as the top-down commercial command and control approach to software development while the bazaar represents the Linux model of software development which is a decentralized cooperative approach. The Linux community is likening to a babbling bazaar with diverse agenda and approaches, out of which a comprehensible and stable system could seemingly emerge only by a succession of miracles. (Eric Raymond, 1999).

3.2 Classifying Methodologies

The importance of classifying methodologies has been to acknowledge how these methodologies are implemented, discover similarities in their theories, and identifies their characteristics, their success factors and challenges to further better their applicability (Fritz & Carter 1994, 1).
In classifying methodologies it is necessary to have a list of what needs to be classified, during this research period, there is not available a definite list of software development methodologies. The reason for this may be the revolution and evolutions of methodologies from the historical perspective and a need to create in-house method-

ology that functions in response to the way software's are built in that particular setting.
There are numerous studies on this subject area and some discrepancies as to what specific methodology belongs to what class of methodology. Some other studies even consider a middle weight class of methodologies. But for the purpose of this research, and taking these into account, however most commonly used methodologies are classified as either heavy weighted or light weighted and recently a third classification called hybrids

3.2.1 Heavyweight methodology

Heavyweight methodology works on the assumption that the more rules and coordination there are, the better the project result will be. Heavyweight methodologies are also known as traditional methodologies and every software development methodology considered as heavyweight have the following characteristics.
The characteristics of heavy weight methodologies according to Khan et al 2011, 442 are; "they are based on a sequential series of steps such as requirement definition, solution build, testing and deployment, Focused on detailed documentation, Inclusive planning and extroverted design."

Rothi and Yen (1989) provided a brief review of traditional software development methodologies. In their journal article, they related how the use of traditional software development models is numerous and often regarded as the proper and disciplined approach to the analysis and design of software applications. Examples of such models included the code and fix, waterfall, staged and phased development, transformational, spiral, and iterative models.

Traditional project methodologies are considered beaurecratic or predictive in nature and have been blamed as the cause for many unsuccessful projects. Any project with a team larger than 10 to 20 people who work in multiple locations may be a good candidate for a heavyweight methodology (Charvat, 2003, 104).

In the earliest days of software development, code was written and then debugged (Code-and-Fix model), there was no formal design or analysis and since the approach to developing complex hardware systems was well understood, it provided a model for developing software. These methods approached system development in a requirements/design/build paradigm with standard, well-defined processes. These methodologies are referred to as Heavy Methodologies or plan driven (Faridani 2011, 7).

3.2.2 Lightweight Methodologies

Newer methodologies have started making an appearance in software projects. These methodologies unlike the more classical ones are considered to be more agile and more able to adapt to change. They do not focus on a long development cycle but rather on short iterations, lightweight processes and rely heavily on close customer involvement. These types of methodologies have come to be known as Light or Agile Methodologies. (Faridani 2011, 7).

Unlike heavyweight methodologies, lightweight projects have only a few rules, practices and documentation. Projects are designed and built on face-to-face discussions, meetings and the flow of information to the clients (Charvat 2003, 102).

Agile methodologies are based on iterations, that is; projects are divided into smaller easy built phases or iterations and at the end of iteration a functional model or product must be ready. There are significant advantages of adopting agile methodologies. One of the most common advantage of agile has been the speed it has through its process brought to how software are created and released. Other attributes are cost and time effectiveness, improved product quality and higher customer satisfaction (Rico, D.F 2008, 17).

4 Open Source Today

Open source is a development method for software that harnesses the power of distributed peer review and transparency of process (opensource.org). This definition of open source encompasses and showcases the ideology of transparency and by so, cannot be complete without first looking into free software, because before the existence of open source, there was only free software. The following chapter will be looking into free and open source history and ideology

4.1 Open Source Development History

OS software dogma takes behind its development processes the termed collective invention. This proposes that when firms collectively invent, they make available to their competitors the result of new plant designs so that their competitors can incorporate extensions of those designs into new facilities. (Robert C Allen 1983, 1). Although this dogma does not concern open source but the main idea remains that when people of the same interest comes together to achieve a common goal, they can affect substantially the course of events. George Siemens 2003 explains that the same goes for open source software developments which have been said to be fostered by:

- Hackers Culture influence on the history of the Internet
- Transition from open to commercial software development
- Richard Stallman- Free software Foundation
- Linus Torvalds- the Linux Kernel
- Formation of OS Initiative

The internet and the whole computer industry are been said to be based on an open culture right after the World war II funded by agencies, administered with software standards and government purchasing rules, the United States Federal government has helped stimulate open source software and open standards for decades (Nathan Newman, 1999). The steady flow of innovations and free software fuelled the accelerated enactment of the Internet and open source ideology simultaneously. According to George Siemens 2003, the followings timeline shows the evolvement of the Internet:

- 1957 - USSR launches Sputnik. The US government responds by creating the Advanced Research Projects Agency (ARPA).
- 1960 – IBM have free software in the early 1960s
- 1961 – MIT Tech Model Railroad club acquired the first PDP- 1 and invented programming tools, slangs and a culture and first adopt the word "hackers" Kim Johnson (2001 in George Siemens 2003).
- 1965 - 1969 - Development and design of ARPANET (an experimental network for research and development of networking technologies - the birthing place of the Internet) (Nathan Newman , 1999).
- 1972 - ARPANET email
- 1970 - UNIX grows rapidly due to favourable licensing terms with universities (UNIX is an operating system developed at Bell Laboratories with the intent of allowing multiple users simultaneous access to the computer). Multiple versions of Unix were developed - most were proprietary.
- 1991 – Linus Torvalds decided to create a Unix kernel that he could make use alone but later open it to the community for use and further development and named Linux (Eric S. Raymond 1999).
-
- 1983 - TCP/IP adopted by ARPANET as a means of connecting various networks (the Internet is essentially a network of networks communicating via TCP/IP)
- 1984-1985 – Richard Stallman founded the GNU project (GNU not Unix) which is a Unix like operating system
- 1986 - NSFNET - five supercomputers connected to ARPANET allowing for rapid development of connections - particularly universities
-
- 1991 - Tim Berners-Lee creates vision for World Wide Web
- 1993 - Mosaic introduced as "an Internet information browser and World Wide Web client"
- 1994 – Netscape made an attempt to commercialise the internet
- 1995 - Netscape released
- 1995 - Microsoft release Internet Explorer

All of these developments will not have happen if not for the free sharing of codes and programs norms adopted by highly committed programmers.

4.2 Free Source versus Open Source

As the topic indicates, free is equal to freedom, not price, there might still exist some confusion in the use of this term, some studies suggests that "open source" has replaced the term "free software", while others still cannot see the relation between free and open. The ideology behind free software was first introduced within 1984 and 1985 by a system programmer Richard M. Stallman who founded the free software foundation and created the "GNU software system". Freedom according to Merriam-Webster dictionary is said to be "unrestricted use" and "thegnuproject" uses the word "unfettered" as the best suited definition of freedom and says all other alternatives as unfit for the definition of its use of "freedom" in this context (gnu.org).

For software to be labeled as"free software" it must adhere to four fundamental freedoms which are:

- You have the freedom to run the program as you wish, for any purpose (freedom 0).
- You have the freedom to modify the program to suit your needs. (To make this freedom effective in practice, you must have access to the source code, since making changes in a program without having the source code is exceedingly difficult.) (freedom 1).
- You have the freedom to redistribute copies, either gratis or for a fee (freedom 2).
- You have the freedom to distribute modified versions of the program, (freedom 3) so that the community can benefit from your improvements. Access to the source code is a precondition for this. (gnu.org).

While open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria stated down by the OSI which are:

- Free Redistribution

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

- Source Code

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

- Derived Works

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

- Integrity of The Author's Source Code

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

- No Discrimination Against Persons or Groups

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

- No Discrimination against Fields of Endeavour

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

- Distribution of License

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

- License Must Not Be Specific to a Product

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

- License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

- 10. License Must Be Technology-Neutral

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

As Linux and Netscape took to the technology scene in late 1990s, OSI was formed in 1998 to foster and steward the "open" phenomenon of collaborative development while the label "open source" "grew from the realization that the attention around Netscape announcement had created an opportunity to educate and advocate for the superiority of an open development process". At this 1998 session, as agreed, the label was essential to distinguish the open source approach from the free source approach which was said to be "philosophical and politically focused" (open source.org). According to gnu.org, "free software" refers to freedom "something which the term open source does not do" and so for the purpose of this research we will be adhering to the word open source and not free source.

4.3 Software

Software in general whether open or proprietary has today become a major aspect of any business, no matter how small or big, but what is software? The word "software" was coined by John Turkey to describe the programs on which electronic calculators ran, first using it in a 1958 article he wrote for American Mathematical Monthly. (LA Times, 2000). For the purpose of this research we will be using this definition which says that software development is the process of taking a set of requirements from a user (a problem statement), analyzing them, designing a solution to the problem, and then implementing that solution on a computer. This definition takes into perspective all the aspect of software development, not just "the listening to what customer needs" (requirements), "implementation aspect" (Programming), "planning and controlling the processes" (management) and other important processes involved in developing software (John Dooley 2011, 1).

According to Jain 2007, software, in last few decades, has captured a foremost arc of human life. It is now not a product of arbitrary and capricious practices and mere programming activities. Modern software products are engineered under the practice of using selected process techniques to improve the quality of a software development effort. The complexity of the lives we lead, the way business is run, the way society is administered revolves all around software's, and these computer software's systems can be divided into three major types. These three types are: system software, programming software and application software.

System Software

System software is a program that manages and supports the computer resources and operations of a computer system while it executes various tasks such as processing data and information, controlling hardware components, and allowing users to use application software. That is, systems software functions as
a bridge between computer system hardware and the application software. Types of system software's are: device drivers, operating system, window system, servers, utility and one of the most common OS system software is the UNIX system (olemiss.edu).

Programming software

As the name indicates, these groups of software's are for programming as they accommodate tools working together through programming languages. These programming languages assist programmers to create source codes for specific purposes. Tools like compilers, debuggers, dismember, GUI and text editors among others all are integrated into these software's for writing software's. At the moment there are thousands of programming software's in the market, each with its own language for writing source codes. Some of the common open sources programming software are: PHP, Ruby, Python, Unix shell and Perl (osait.com).

Application software

These are set of programs that are designed to carry out operations for a specific application (itsavy.com). These applications allow end user to accomplish one or more specific non-computer –related tasks. Application software ranges from word processing software, database software, educational software, computer games to even medical software(osait.com).

4.4 Open Source Software Today

Software's as we know today, can be large or small depending on its functionality and according to Wheeler 2007 report shows that "many people think that a product is only a winner if it has significant market share. This is lemming-like, but there's some rationale for this: products with big market shares get applications, trained users, and momentum that reduces future risk. Some writers argue against OSS/FS or GNU/Linux as "not being mainstream", but if their use is widespread then such statements reflect the past, not the present". (David Wheeler, 2007).
The netcraft.com survey of 2007, shows that the most popular web server; Apache has 58, 8% of the web server across all domain while its proprietary counterpart Microsoft only has 31.13%, this report clearly showcases the position of OS software today (web server survey 2007).

Among the most successful open source software are Linux and Apache Server as forerunners and others like Python and PHP programming language, Mozilla browser, Eclipse, MySQL, OpenGlass and OpenOffice. Most of these success stories do not provide documentations on what development methodology was applied except documentations on version control and source code itself and for example the Apache and Mozilla project only provide informal narrative descriptions of the overall software development process (Jacques Longchamp 2005). But as OS software becomes more and more adopted, there is a need to have in place a collection of best practices and development processes that can be reused as a method for developing OS software.

This is an eye-opener also in recognising other open source software out there, even major world class business shakers are investing in OS projects, according to IBM's report 2013, 94% of the world's top 500 supercomputers run Linux while in 2012, Linux based Android lead the worldwide smartphone market and even invested one billion dollar into Linux. Open source software not only attract individuals who want high quality software and cannot afford expensive commercial version, but also become good candidates in many businesses or governments' Information Technology plan (Ying Zhou & Joseph Davis 2005 and in Finland the government is encouraging communities to use open source software instead of closed software (Liisa Auer et al. 2011).

4.5 Issues in Open Source Software Development Project

There are issues affecting open source software development as we have come a long way since the 1950 when the word software was first coined, there are of course common factors like finance and project methodology involved in creating the software but other essential factors that are apropos to open source project are addressed in the followings.

4.5.1 The People Factor

The ideology behind open source, has never been monetary but rather people trading their time and their mind in exchange for value, which cannot be measured in monetary value and in other words, "value = mind-sharing –gifting-exchange". (Nicholas Paul 2011). Numerous studies have shown a very large majority of open source project are being abandoned by developers, for instance; SF hosts over 174,333 open source projects and out of these projects, the more or less successful or failed projects were 145,475 and of these only one in six (17%) were successful while the rest of the 83% were abandoned Schweik and English (2012, in Rich Gordon 2013, 281).

Researches also show that for an open source project to be successful, developers should also be users of the software. This give a major boost in motivating developers to create successful software thus a "user-centred need" will likely be met, because a need and interest in the finished product exist (Rich Gordon 2013, 129).

4.5.2 Leadership Factor

Even though open source project usually starts off relatively small, the influence of a well-planned, vision-oriented, structural-communication plan should not be over looked. According to Chorng-Guang Wu (2007) the Linux project was a success because right from the beginning the developers contributing to the project understand the initial developer's vision, and it was made available for anyone interested in further developing and improving the software. A major issue if not well planned that can affect open source software development might be communication and documentation.

Effective communication and sufficient documentation is one of the building block of a successful (Yiyun Shen 2008, 4). An effective leader of an open source software project should make available a strong project communication supporting system which should include an active website, good documentation/conversion system, a bug-tracking system and an intercommunication system (email or forum). (Rich Gordon 2013).

4.5.3 Maintenance and Support

Open source has been perceived right from the beginning as a "leap of faith", that is in the case of maintenances who will be responsible for upgrading open source software. Lately the availability of support for maintaining open source software's are increasing due to the idea that developers of open source software are likely to be the users of that software and the more consumer of the software the likeness of it being also maintained, for example in the case of openOffice which is 99% compatible with its proprietary counterpart. So support is not only available but also simplified (Bill Appelbe 2003, 233). There are also companies giving support and maintenance for open source software, such as RedHat.

4.5.4 Finished Product

For almost every proprietary software out there is an alternative open source software available. Examples like; windows 8 to Ubuntu, Internet Explorer to Firefox, Microsoft office to OpenOffice, Microsoft Visio to Dia, Blackboard to Moodle, Dropbox to Cabos, Microsoft Project to Open Workbench, Adobe Photoshop to GIMP, Microsoft Paint to Tux paint, Skype to QuteCom, Microsoft outlook to Thunderbird, iSong to Songbird, iBackup to Amanda, WinZip to 7-Zip, Salesforce.com to SugarCRM, Google Chrome to Chromium, iOs to Android and this list is almost indefinite (Jimmy Atkinson 2013).

From the above examples, it clearly shows that open source is carving its own market niche and doing it prosperously and just as Schweik and English (2012) puts it "Investment doesn't appear to drive open-source projects. Rather, the need for the software drives development" (Schweik & English 2012).

4.6 Open Source Project Tools

Open source do operate most times on a very different platform but still needs some management tools to function. These management tools make collaborations successful among all the developers and ease their participation, since the developers can be geographically dispersed. (Audris Mockus et. al. 2002, 311), hence the following project tools have been selected as needed for any successful open source project:

- Software Configuration management (SCM) tools
  - Revision control system (RCS)
  - Concurrent versions System (CVS) (Repository)
  - Subversion
- Issue tracker
- Continuous Integrator (CI)
- Testing tools

- Information channel
- Open Source Licences

SCM tools are set of mechanisms for controlling and storing the evolvement processes of any software system, that is, all the changes done in the course of the software project is recorded and stored onto a software configuration manager. Any open source project is ever evolving and change is at its core, so SCM tools help to manage these changes control Vivek Venugopalan (2001). Since open source projects are organized around a central repository from which collaborating developers can retrieve copies of the source code, make changes to them and upload the newer version back to the repository.

The Concurrent versions system (CVS) supports this decentralized software development approach as it gives all the developers the freedom not to be continuously attached to the main repository while developing a function for the project (Andre van der Hoek, 2000, 43). Continuous Integrators role is to build, test codes and build status. The issue tracker is in place as a collaborative hub for project developers when developing new features. Identifying and fixing faults and storing references for maintenances to ease bug or problems tracking noticed during the course of the project life cycle Yuhoon Ki et al, (2009, 526). While according to Walt Scacchi 2007, the most common method for information communication in open source projects are email list, group blogs, discussion forums for example; IRC gallery, news postings, instant messengers, bulletin boards, to-do lists, how-to-guides, project wikis and FAQs. These communication tools are essential for making sure every developer on the project gets the rightly informed at the right time.
Test tools main roles are for validation and verification; these two features help in verifying feature functionality, reliability, security and identify bugs in the software testtools.org (2013).

4.6.1 Open Source Licenses

Licenses are essential in open source software and the reason being that source code must be made legally available for the developing community while making sure that they can be modified, shared and used. (Open source licenses). An open source license in particular gives the "permission to do or not to do something" (Dictionary.com) according to the OSL. Under the OSL, the initial source code developers still holds the original copyright (or patent) on the software but at the same time grants permission to other developers to use the source code and prevent others from claiming your work as their own . (Cameron Chapman, 2010).

The most popularly used OSL according to opensource.org are:

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License

Among others, even though there are more than 70 types of open source licenses among others. The above listed licenses all have different usages and grants as well different permission for different usage. The rise in open source software's has also led to the rise in the amount of open source licenses available. The ideology of these licenses is that, the software's source code is been written by someone or group of people, which is/are the proprietor of the software and almost all open source licenses contain specific clauses as to how authorship has to be credited (Gwyn Firth Murray, 2009, 1). Companies that choose to publish the source code and engage in open development hope to gain external contributions from other stakeholders such as customers, partners, and voluntary developers using a variety of business models to gain revenue. For other stakeholders, the legal relationship is based on the open source license, under which the initial source code is published for the community (Juho Lindman, et. al. 2011, 31).

OSI has a list of 10 requirements needed for a license to be called an OSL and in choosing the right license for software; the company must first decide how it is willing to make profit off the software by considering whether it's going to be selling support, selling connected hardware or commercial software (Juho Lindman et. al 2011, 34).

5 Empirical Implementation

The interview was semi-structured to permit participants to freely discuss subjects within a mid-controlled atmosphere and by so; enrich the ensued discussions outcome between the interviewee and interviewer. From the theoretical research, selected topics to be covered in the interviews were created to structure and guide how the interview will be carried out. The semi-structured interview questions were well designed to have major questions with interval questions to cover extensively the research purpose and answer the research questions.
Since the companies involved in the project are selected based on their business activity, the sampling is purposive and deliberate. A short introduction on all companies participating in the interviews will be included in the context of the project. The companies will be contacted before the interview to get help on getting a substantial sample of the companies' employees to participate in the interviews. All interviews were recorder, important memos taken, transcribed and analyzed and discarded after the project is ended.
There were a total of 6 companies that participated in the empirical data collection stage of this thesis project. The companies' names will not be published to protect their businesses from any unintended important business information leaks, instead the companies will be named as Company A to Company F and all companies are regis-

tered and operate in Finland. The interviews were designed to connect the theoretical background of the thesis and by so doing answer to the research questions

5.1 Company A

Company A is Devlab and it is located in Helsinki and it was a spinoff from the Finnish wholesale company in 2006 and has develops one open source ERP software system which has been developed over 10 years. Their customers are small and medium sized Finnish companies, some of which have subsidiaries in Estonia and Sweden. The company also provides support services for these subsidiaries. The company has 10 employees and about 30-35 customers with business turnover up to 1 million Euros in per year. The interviewees present are the company's CEO, Timo Sundell and Johan Tötterman a developer. The CEO do have a developer background and his role in the company is more on administration and sales, high level specifications and describing customer's business needs. The developer has been in the company from the beginning and has been participating in the development of the software right from the beginning and he reviews all the codes written by other developers that goes into the software and sometimes participate in coding also.

5.1.1 Methodology Used in the Company

From the interview analysis from company A; it was very clear that the chosen methodology is waterfall model and this model for the company is modified to be able to accommodate their customer's needs and how business is being done in the company.

> I would say that we mainly use waterfall and that the whole development is mainly done for customer's need. The customer pays to get that specific new feature or new functionality. Our customers are used to knowing in beforehand, how much this cost. They come up with an idea and then we do the high level specification, we tell them that this will cost a specific amount. And then we would implement it and deploy to them and that's the model that they like (Sundell Timo, 2.Oct. 2013).

The ERP system by the company dictates what type of business the company can provide to their customers and even though the rigidity of waterfall is taken known, the company is very confident in its choice of methodology.

> Our business model is to sell implementation project and support and development of our ERP system. The waterfall model, even though it's a strict sort of way of working gives both us and the customer the knowledge after the specification. I would say that in a smaller project the way we work is still quite fast, even though it has a lot of steps because you know it could be that the customer today is complaining that he don't like how this one functionality work and tomorrow he already have the new version of it and we have done all the steps even though there is a lot of steps you can be quite fast with the waterfall model (Sundell Timo, 2.Oct. 2013).

Here the interviewer presented the waterfall phases form the thesis theoretical background to the interviewees to give a direction of how waterfall model has been implemented. The interviewees agree with the proposed phases of waterfall model but also added some loops that showcase how the company implement waterfall.
This is the phases presented:

1. Definition Study/Analysis
2. Design
3. Implementation
4. Verification
5. Testing
6. Installation
7. Maintenance

According to the interviewees;

> I would say it's similar, pretty close and we do the step one and two. The Definition study/ analysis and Design phase but we call it the Specification phase. It describes the customer's problem and it has the technical specification which is more like instructions for the programmers. The Implementation phase is the actual programming while the Verification phase is the code review phase and we are testing in two levels before verification. The programmer's responsibility is to test the unit test. He tests the functionality that he has changed or created and there is a project manager who knows what the customer's problem is and then he does the final testing also. Then putting it into the production is installation and then it goes to maintenance. That is pretty much correct but I would also add a loop into that:

Because unfortunately the customers don't always are not able to tell us exactly what they are looking for, or we don't understand what they are looking for. We think we are doing exactly what they want and then we install it to their environment and then they say that, Yes this is nice but this is missing or this is not at all what they meant. So then, it becomes a second round, sometimes this happens and then it goes back to the specifications (Sundell Timo, 2.Oct. 2013) .

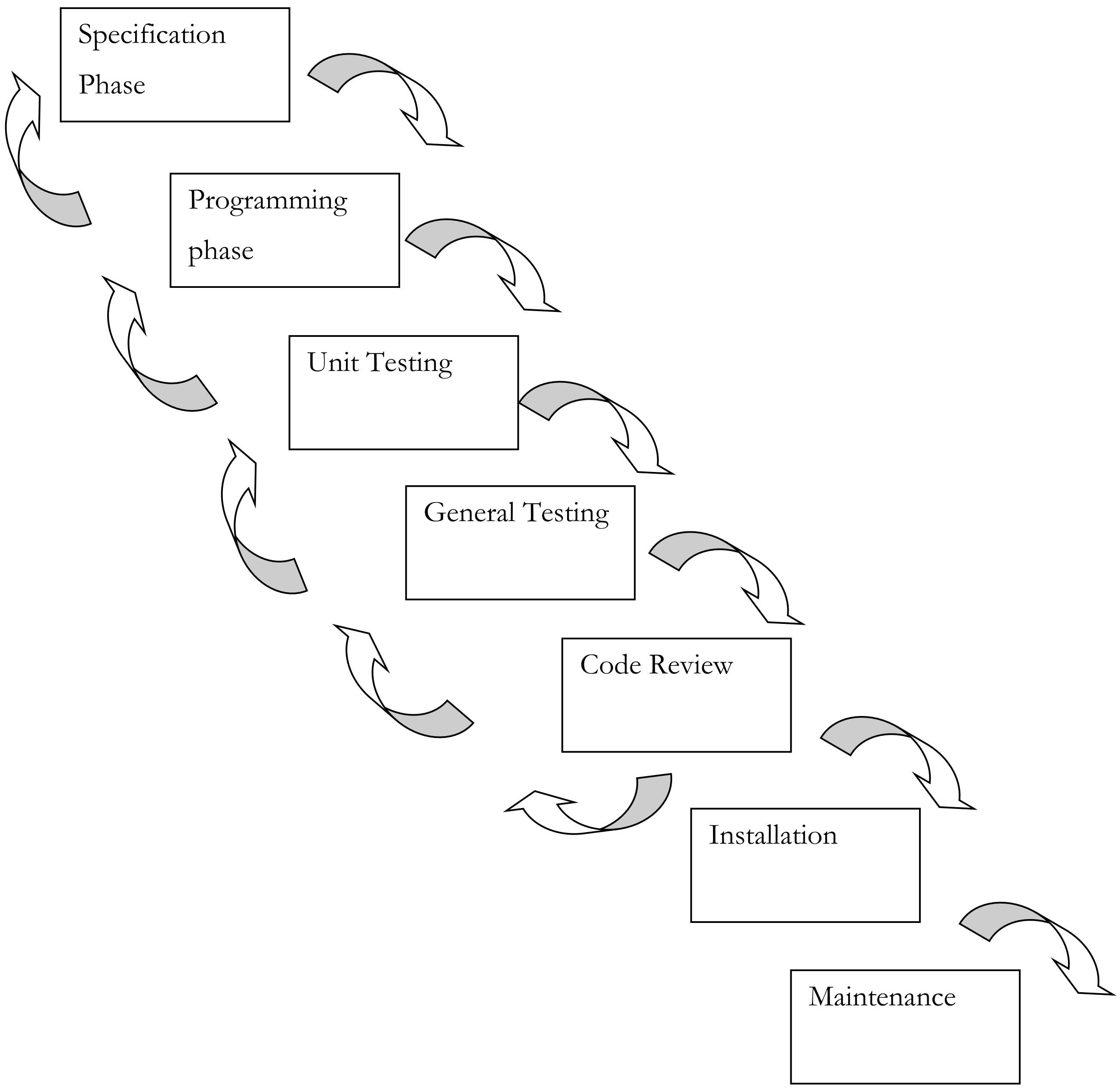

Figure 6: Waterfall model in Company A

This is the basic structure of the water fall model implemented at company A. From this basic structure, there can be loops that can start from the first phase of specification to phase code review before installation and maintenance phase.
These phases depending on the customer needs;

> the shortest implementation one can be done in from step 1 to step 7 in 1 day or like in a couple of hours if it is a really simple feature but then sometimes bigger projects are like 3-4 months (Sundell Timo, 2.Oct. 2013) .

The advantages for selecting waterfall as the choice of methodology for the company is very vast as the interviewees gave despite recognizing the heaviness of the methodology. The followings are the advantages of this methodology;

1. It gives a good structure to the whole project.

   > I think it gives a good structure to the whole process. I mean we have these specific steps that need to be done every time and we have the specific documentation, rules for every step. It's really important for us to know what the customer wants and being able to deliver just that (Sundell Timo, 2.Oct. 2013).

2. Captures the importance of effective budgeting.

   > The customers are paying for the development of software and that means that our budget should be the one that meets the amount of time that we have used for that specific task in developing the software. At least we should know where the goal is and also its easier to estimate how much labour hours we have to work to reach the goal (Tötterman Johan, 2 Oct. 2013).

3. Support good practises in documentation.

   > For us it is important to been able to document and know what the customer wants and do the testing and verification because if we skip some of the steps; in a small project you would maybe would like to skip alittle bit like cut corners here and there but almost every time you run into problems if you cut corners (Tötterman Johan, 2 Oct. 2013).

4. Familiarization with the model.

   > It's a familiar process, so everyone knows what that the next step is testing and after that verification. The customers do like that and think that they get very good end solution this way (Sundell Timo, 2 Oct. 2013).

5. Supports the continuous development of the software.

> There is only like one version of our software and all our customers are using that version which is always the newest version and the software is living all the time (Tötterman Johan, 2 Oct. 2013).

6. Provides better opportunity for testing.

> Because the software is very big and if you don't do all the things proper way they might have side effect. The thing itself looks fine but actually because the same program is used in 10 different places, so if one place was broken and doing the proper testing and code review every time would decrease these side effects (Sundell Timo, 2 Oct. 2013) .

The interviewees can only find one specific disadvantage to implementing the waterfall model the way they do, which is that it can feel heavy sometimes.

> I think small things to be developed sometimes feels heavy because you know, we have internal rule that even though it's a tiny little thing for example customer want to change one word; it would be a 2 hours project. Because you know you have to go through the different phases so for small pieces it is a little bit heavy and even for this small ones we don't cut corners because it would create problems. Because the software is very big and if you don't do all the things proper way you might have a side effect (Sundell Timo, 2 Oct. 2013).

The company uses PHP as their programming language, mySQL open source database, Apache web server and the Linux operating system, Version control is in Git and have a repository on GitHub while testing is done manually. Choice of license is GPL version 2

> Everything we do is GPL version 2, the developers know it and they know that they can take other GPL projects and embed them into our software. This license choice has been pre-decided, it's the GPL version and it's the same license for everything we do (Tötterman Johan, 2 Oct. 2013) .

5.2 Company B

Company B is located in Tampere and they develop and design open source web services software's for a whole range of customer needs including Linux support. Interviewee was Kari Lehtomaa, web developer a web developer for the company and his role includes developing plug-ins using Word press and node-js programming and also providing Linux support, at the time of interview, this employee no longer works for the company.

5.2.1 Methodology Used in the Company

According to the interviewee, the company would say they implement agile but from his perspective, it is more a mix of Scrum and open source development model and this mixed model in use is not so effective due to poor project management.

> There was problem with the project management because there were no good definitions for the software and changes occur too often (Lehtomaa Kari, 8 Oct. 2013).

From Scrum, the company implements different ceremonies: sprint planning, daily scrums, sprint review and sprint retrospective. There is the product owner, scrum mater and a team of 4 employees, but then even at sprint planning's, sprint tasks sizes were not realizable within the given time and the scrum itself is very loosely implemented.

> The sprints last from 1-2weeks and the team consists of 4 employees. It was quite hard to do the sprint because a team member might commit to do something for the sprint and not get it done by the end of the sprint and once a month we have inputs from the client (Lehtomaa Kari, 8 Oct. 2013) .

The open source development model is practiced in that, there are the architecture and designs discussions phase, collaboration on implementation phase and source code submission. The continuous testing and integration was very loosely used. At the start of the project, the project manager have a the meeting with the customer where the customer informs of the goal f the project, then the project manager forms team where he discusses ;

> the client's issues in a meeting based on client's needs. Get the tasks needed for the sprint divided and after the meeting, sprint begins and everybody starts to code. We use GitHub for collaboration on Implementation , live chat as information channel, to

tell each other what they are doing for example I'm working on this please don't touch. Continuous testing and integration: Starts when the software is 60- 70% ready. GitHub used for the process of testing and integration (Lehtomaa Kari, 8 Oct. 2013).

The interviewee stresses that the use of this mixture of Scrum and open source development model is not well implemented and does not see any advantage of implementing this model.

> I don't think it's a good idea to mix them. Open source development model techniques are more for commercial software, and then scrum is the one to use with good project management. In loose team's projects, where you have developers around the world, then the open source model is better (Lehtomaa Kari, 8 Oct. 2013).

Developing tools used in the company are Git with GitHub, Issue tracker was trac. Java was the programming language; there was a continuous integrator in use but not with what he was developing for the company while testing tools was JavaScript. The GPL license was in use in the company.

The Scrum / open source development model in use in this company could have function better because open source development model supports greater end-user involvement, intra-team collaboration and also accommodates other development processes which in this case could have been Scrum, if the models have been well managed.

5.3 Company C

Company C is Encompass Solutions Oy, which is located in Oulu and focuses on developing and supporting start-ups. Their business model functions in developing start-ups ideas into products. They work directly with idea owners and products are usually developed in about a week and they have a low and efficient flat rate. The interviewee is the CEO of the company and also contributes to the company as a developer. They develop anything from web-based solutions, mobile web pages, web-tools and robotics. The company participates in a wide variety of project and use open source on almost every level of the company. There are 6 employees working in the company.

> We work with robotics; our bigger money maker right now is web applications. Web applications is a big thing, everybody wants a web page and most of the time we can create a product faster in a web version than we can in any other version.

The major methodology used in this company is Lean startup mixed with Agile.

> We do Agile of course with Lean startup. Let's say that you're using my product and you don't like some aspect or some features that needs to be improved, so you email me, I will read it, I don't ignore it, but I won't write it down and put it somewhere and file it away as some official document, I won't put it in a spread sheet anywhere and the reason is that the features that bug us the most will stand out to us already and we as humans would remember the problems, so we take those and we work on it. The Lean startup approach is used because we are involved with startups (Brower Jason, 14 Oct. 2013).

Lean startup is a system for developing business or product in the most efficient way possible to reduce the risk of failure. (Lean startup circle wiki, 2013). This methodology functions bearing in mind some characteristics;

> The idea with Lean startup is that you create small changes to an application, release it as soon as possible, as soon as you think it will make you money and then see what the customer does and we have to be so agile, develop so flexible that we can adapt to anything that comes along (Brower Jason, 14 Oct. 2013).

Not only for startup businesses but also other form of business development, this is methodology seems implementable, even though this methodology is quite fresh as it came into being in 2008. (Eric Ries, 2008).

> The way it works is that, Eric Ries the author of the lean startup, he started many companies and he realized that there is a certain pattern in the successful businesses and there is a way to develop a company and an idea in a way that's its able to adapt to what is really needed. It's very Lean startup style to just take one little change at a time and we develop it and see what happens
> The idea is that I have this things that I want you to buy it, the problem is that you as a customer will complain about the product , you can say I want it to be more like this or like that , so why don't you make it more like the best email service out there, whatever you are using at the moment and if I would have followed that idea I would actually be following a competitor and I would be behind , so whatever best email service does and everyone is saying why don't you do this like that and so I just put myself as a company behind them. I can't think like that, I can't think like you do as a customer. So the cus-

> tomer is not always right. In this case, very much so, we can't do that, or we put ourselves behind our competitors. Now if we listen to ourselves and we ignore you that is also a problem because we tend to develop in a way that caters to us only. So we diverge in a different direction and if we are lucky we are better than the best email service out there (Brower Jason, 14 Oct. 2013).

The ideology behind this mixed method is to be able to give to the customer the best possible development experiences faster than and as empowering as possible. The principles guiding this mixed of Lean startup and Agile are:

- Finding better innovative in developing business
- Doing only what is important to the developing of the business at one specific time.
- Development must be fast and releases should be small and effective
- Lean way of development should be empowering to use small workforce become effective.
  (Eric Ries 2008).

The advantages of Lean startup mixed with Agile are according to the interviewee;

> It really lays down what is the right way to focus on a business idea. It cut the product into the right batch like 1 at a time, we can many times create a better, more efficient product, so when we do software development, we take small steps to find the result. We work on such small models which have small steps so if we do that, which is very agile and we do it in a small style, then the architecture is very straight forward. We know exactly how to work the application to fit that particular feature, so it's very easy to do (Brower Jason, 14 Oct. 2013).

The company uses tools like; continuous integrator (CI), some basic testing tools, Version control, Git and a little bit of subversion. They also use GitHub and Bitbucket as a DVCS hosting site and language of programming is python. Licenses in use are LGPL, GPL and are planning to take the new Affero into use in the nearest future

> The reason is GPL has a conservative approach to the licensing that it still always remains open and anyone that uses it must use it open software. But that's also the problem with GPL, but it's also the advantage because I can keep it open until when I understand the advantage of where it's going as a product and then I say I will now make it LGPL(Brower Jason, 14 Oct. 2013) .

The company has also earlier used Bazaar and the cathedral methodology in an open source software development project, which is still today active. The result of the project is software called Memaker which is an avatar maker for Linux which is still in use even today.

5.4 Company D

Company D is is Lumolink, a project oriented company located in Oulu and does online marketing and web solutions including web applications that are mobile friendly and scalable on all mobile devices for customers from all over Finland and Russia. There are 5 to 7 employees working in the company depending on the project. The interviewee is the CEO of the company, Artem Daniliants and his role in the company is doing marketing for clients; also participate in internal product development. The interviewee teaches scrum and open source development model.

5.4.1 Methodology Used in the Company

Company D choice of methodology is Lean approach mixed in Scrum. As the interviewee is a scrum teacher himself, the reason for this type of methodology is;

> When it comes to software development we use more Lean approach and abit of scrum. The problem with scrum is that, you have a set period of time, for example we get 2-3 weeks, it depends on the team and then of course on the time of the project but the goals are set, you are not allowed to inject additional task, remove task and so on. You prioritize the tasks but the time is set for 2 weeks and like 28 tasks and that's it. But in a small company like ours, the problem with that comes that we have to have ability to insert additional tasks, and remove tasks and so on and scrum doesn't really do well in that situation, when there are constant distractions . So we use Lean, in that sense that we consider ourselves like a factory and we work on one piece at a time, the pieces are arranged in a backlog and they are prioritized and so on and each piece go through certain stages and then its ready but we don't have a set amount of pieces for a certain amount of time so what happens is that once, one piece is ready, you can take another one and that could be from another project. It could be that it arrived today in the morning. So we focus on one piece of one task at a time but we don't set fixed

> amount of task and we consider it in our process a better model than scrum (Artem Daniliants, 14 Oct. 2013).

The processes involved in this methodology are:

Client requirement gathering;

> First of all, it starts with client requirements. And clients are horrible at making requirements but that's a known fact (Artem Daniliants, 14 Oct. 2013).

Functionality mock-up phase

> We do functionality mock-ups, so it's easier for clients to say that, I want this button to do this, this is much simpler considering that in most cases, our clients are not technically savvy, so in that sense it's very easy for them to relate when there is something visual. We basically try to vacuum as much data from the client as we can, for example would you want a weather widget here(Artem Daniliants, 14 Oct. 2013) .

This is then followed by the process called information architecture phase;

> Then its information architecture, so once client is satisfied with mockups, we send them to be approved and mockups are accommodated with the functionality description under every single screen, so if we have for example application first screen, that will list what each elements does and what is the goal of this view, and then we send it to the client, then client approves and says yes, that is what I meant, that's how it should be and if it is not approved then we iterate until it is approved (Artem Daniliants, 14 Oct. 2013).

Once the information archicture has been approved by the customer then technological specification is then documented.

> Once its approved we arrive at the technological specifications and that usually takes a few days up to a week, if it is a really big project. So what we do technical specifications and that's a must for us, but we try to only focus on things that are not obvious. So things that are implicit we don't mention them. Only things that are explicitly need to be mentioned and known ahead of time. Once technical specification is done, we don't even send it to the client because in most cases he doesn't have the technology savvy to understand it, so what we do after that is that we break the specifications into larger tasks. For example we could say that user's authentication, this is just one task and then we break it down to for example backup functionality and so on and we are left with maybe 20 to 30 tasks (Artem Daniliants, 14 Oct. 2013).

The next phase after technical specification is work time estimation;

> After that is done, what we do next is that we estimate roughly how long it would take. What we do nowadays is that we estimate in terms of weeks, so that allows us for huge error margin but it is also easier for programmer to say, No, it's not going to take a week, but if you ask him to estimate in days, he might be like maybe in 2 or more days but at this level the only thing im worried about is how long will it take in general more or less, so week is a pretty good number because it very easy to say that it would take over a week or less than a week because if it's less than a week we don't really look at accuracy but if it is more than a week then we say it seems like a huge thing, so let's think why is it so big. So in terms of programming, week is a huge amount of time (Artem Daniliants, 14 Oct. 2013) .

This phase is then followed by the Monday planning's where the importance of information channel is essential. The Scrum's backlog ideology with having tasks in different columns that support how tasks are been done is incorporated very much into this phase.

> So once we have 20 tasks and each estimated that it would take less than a week. After this we start the week, so every Monday we have Monday planning, so Monday planning is where everybody meets on Skype or other meeting methods and for our processes we use Trelo. Trelo.com is a free project management tool that adheres to the Lean and scrum models, awesome product. It has just columns for backlog, To-do and so on. So in the Trelo, what we have is, we have a backlog column with all the tasks, then what we do in Monday planning is that we take some of them and we split them in much smaller tasks (Artem Daniliants, 14 Oct. 2013). .

These smaller tasks are then taken into the week objects, which are then the tasks that will be completed for the week. The week contains the daily sit downs where the scrum questions are asked and answered.

> We then have the next column after backlog, called week objects . We split the tasks and we agree that ok, these tasks will be done in this week and we classify them In terms of priority and after that we begin working and the person who takes the task will move it to the Doing column and then once its done, you move it to Review and then every morning we have don't have daily stand up but daily sit down and the idea there is that we go over the task that everybody does , we answer the scrum questions, what have you don't yesterday, what are you going to do today , are they any blocks(Artem Daniliants, 14 Oct. 2013) .

After a task has been considered as "Done " it is moved to the Review column.

> We go over Review Column, we review the tasks and we decide that can we approve it or should it be moved back to the doing and the person should be advised on how to proceed. And if we agree that it can be moved to the Done column, we drag it to the done column and we agree that its reviewed. It's a very good way for tasks to be reviewed by many people at once, so in this way it kind of a decisional quality control mechanism (Artem Daniliants, 14 Oct. 2013).

Below is the company's methodology in phases presented diagrammatically in figure 7.

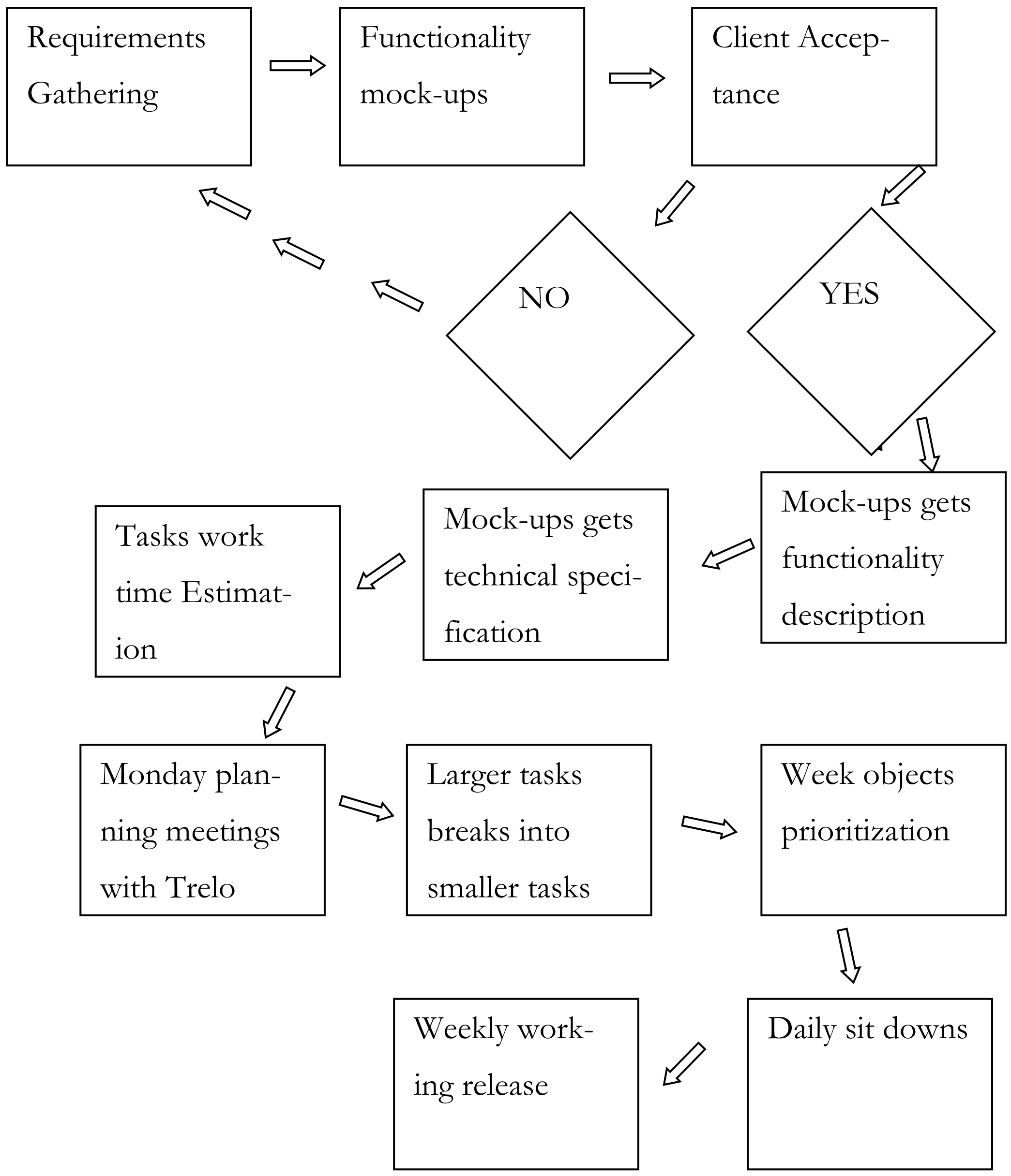
Requirements Gathering
Functionality mock-ups
Client Acceptance
NO
YES
Mock-ups gets functionality description
Mock-ups gets technical specification
Tasks work time Estimation
Monday planning meetings with Trelo
Larger tasks breaks into smaller tasks
Week objects prioritization
Daily sit downs
Weekly working release

Figure 7 : Lean approach with Agile in Company D

So this way, at the end of the week, it has gone through a vigorous quality control system that is still very Lean in terms of a small company size . The process is not burdening and every week you push working release that has been in the staging server, passed unit testing, past code review, passed peer review and obviously we proceed until there are no more main tasks in the backlog (Artem Daniliants, 14 Oct. 2013) .

The advantages of this methodology support the business and its mode of developing software;

> In our environment we cannot say No to additional tasks, if clients wants to get something done, we have to get it done. In scrum there is this holy team that is not allowed to be interrupted and if you ask the scrum master that how do you deal with that situation, well he might answer that there is the sprint on at the moment, and he can kind of be the buffer and this just doesn't work in a small company. So for us in our company, the advantages of mixing methodologies to make the methodology work for us (Artem Daniliants, 14 Oct. 2013).

The disadvantage identified is the difficulty of introducing a new team member into the team since this is not entire Scrum neither is it entirely Lean.

> It's hard to bring people in, so if we were like hard core Scrum, then somebody who is Scrum certified could come in and understand our method but when you have your mix, you have to document it , you have to have some kind of agreement, in terms of we agree to work like this(Artem Daniliants, 14 Oct. 2013) .

The tools used in company D are; Programming Language, Python, content management system, Drupal and Django. Git is used as revision control and an issue tracker is also in place. Continuous integrator is also used and some testing tools. For information channel an online chat called Hipchat is used. Balsamiq software is used as as the functionality mock- up software. BSD and MIT license are in use.

5.5 Company E

Company E is part of Europe's largest Drupal shop Wunderkraut, which is present in 9 countries and has 35 employees in Finland and located in Helsinki. The company builds web services, intranets and other internal tools that customers use in information sharing. The interviewee was Cristian Andrei and his roles in the company are developer/ project manager, takes care of customer communication, planning and creating project estimates.

5.5.1 Methodology Used in the Company

Scrum methodology is the chosen methodology chosen in the company and this methodology drives the way the business functions and even dictates what type of customer the business can work with.

> Agile is our main focus when it comes to doing projects and scrum is the method. We actually turned down a couple of customers because we do not work that way. We do not accept requirements, 50 pages of requirements and layout and someone coming to give that to us and then coming back like in 6 months and asking where is his product don't work for us because we like to be in constant contact with the customer (Cristian Andrei, 23 Oct. 2013).

The scrum methodology building blocks are implemented in the company late until after the project has been launched and delivered to the customer then the business relationship continues as a free mode agile principled business relationship that is sustainable.

> We start off with scrum and then we do sprints as much as it is rationally possible. Because once we get to the phase when we don't have any more stories to fulfill, then we just go into some sort of free mode where we do performance improvement and fine tuning of the system and in that case we don't really need sprints (Cristian Andrei, 23 Oct. 2013).

The end goal of the project is the first phase of the project then methods based on constraints are analyzed

> ….we sit down and discuss what is the end goal of the project, what is the business need that the project will feed and then we device methods on how to get there, based on time constraints, financial constraints and our own developers constraints (Cristian Andrei, 23 Oct. 2013).
>
> .

A generalized deadline for ending the project will be set but not a set of functionalities to be delivered.

> We identify what is the crucial part, what is the most haves of the project and once we do that then we start working. And then we know that Yes, we will launch when we have that particular set of functionalities fulfilled. And then we launch and then we just take it from there. So then the customer knows that they will not get all the bells and whistles from the very beginning mostly because if they are not business critical we will not choose to implement that in our first iteration (Cristian Andrei, 23 Oct. 2013).

The scrums building blocks in use by the company are:

> …..do iterations and we also use sprints, which are taken from scrum. To us sprint, is a set-time frame when we just develop and usually we have a one week sprint and we kick it off with sprint planning, where we set the goals for that particular sprint and we then identify the dependencies and the blockers for future sprints as well and after that we just start working trying to fulfill the sprint goals and at the end of the sprint we have a meeting where we demo and or analyze our sprint work and then we sign off stories or we take them to the next sprint and usually the customers is with us at our sprint planning and sprint retrospective meetings (Cristian Andrei, 23 Oct. 2013).

The team size on average is 5 employees and if there is need for more employees, then the team is split into two.

> …..Usually our project teams consist of scrum master/ developer, one or two developers, an employee which is the person which handles the graphics of the website and makes everything pretty and then the product owner (PO) which is somebody from the customer always and that is our scrum team (Cristian Andrei, 23 Oct. 2013).

Below is a representation of the methodology in figure 8.

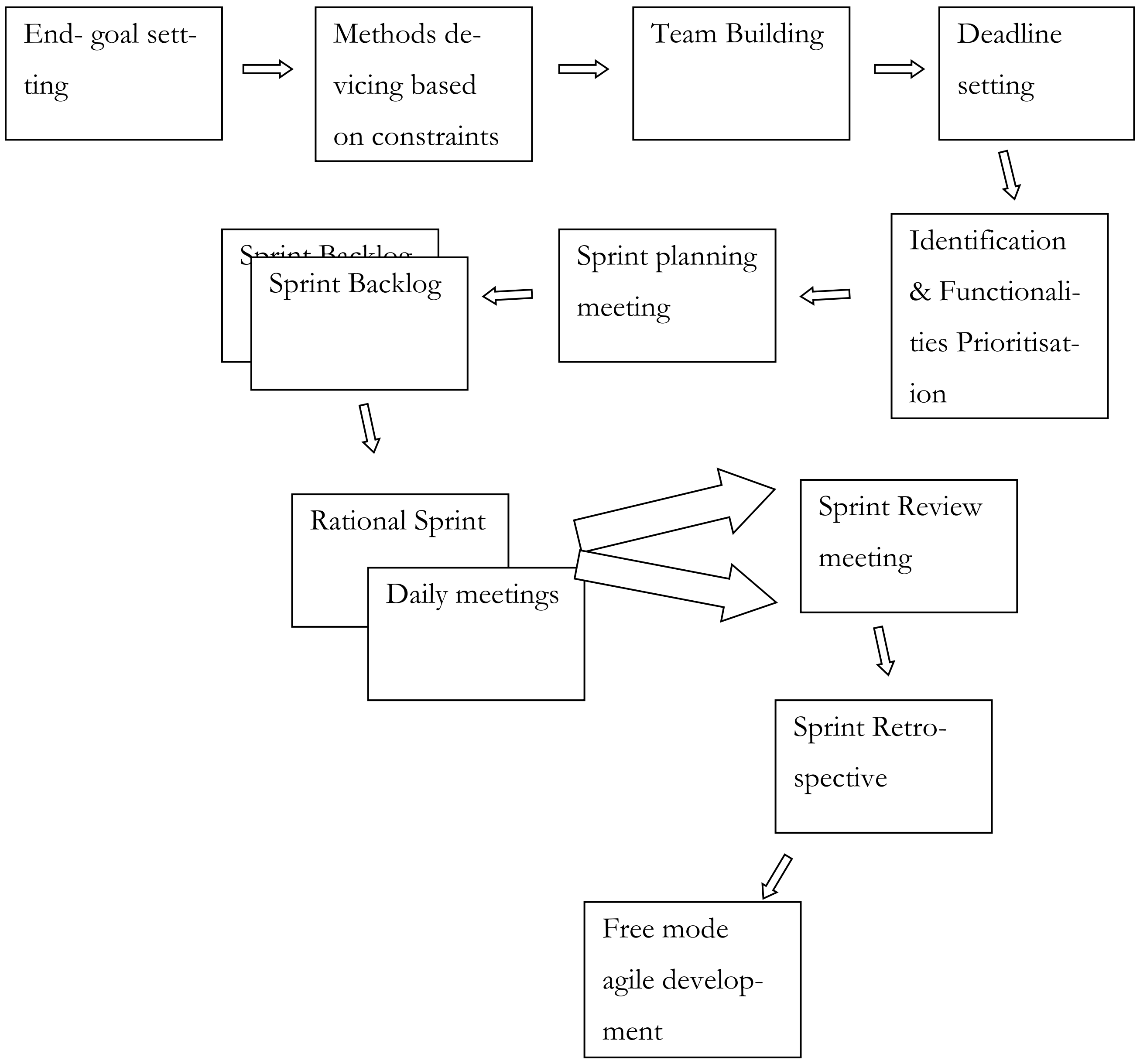

Figure 8: Scrum methodology in Company E

Then after scrum, the methodology becomes free mode agile principles adhering.

> It is hard to say when a project is finished or really ready because it's never really ready. So when we have the big launch or the main project has ended then we go into continuous development mode. And that varies from customer to customer, either fixed term packages or fixed budget packages and we just let starts and usually the customer gives out a list that says for example, we noticed that this is not working, can you fix them and based on the details we make it work and try to fulfill the list (Cristian Andrei, 23 Oct. 2013).
>
> .

The advantage of this methodology for the company is that it gives better time for reacting to customer needs.

> It gives you a better chance to react. . Let's say that we have a company which has an online business who sells shoes, the market is really big for shoes right now but it could be that in 2 months time, it would go to hoodies, so we need to adapt faster (Cristian Andrei, 23 Oct. 2013).

The disadvantages of this methodology are insignificant in comparison with the advantages it provides.

> All the disadvantages that I may think of are not so important due to the nature of small iterations and short timeframes and react on whatever environmental changes there may be. And at some point this doesn't have to be environmental changes, it can also be the market, or that the customer has some other problems and the project needs to be put on hold (Cristian Andrei, 23 Oct. 2013).

For the open source software development project, the company uses tools such as Git, GitHub, and Beanstalk. Subversion tool is also Git, for Issue tracking the customer has their own custom made called Case tracker. Drupal is used for content management. Basic functionality testing tool is Behat , other testing is done manually and Skype is the main mode of information channel. Since the company uses Drupal and Drupal uses GPL license, so GPL license is in use.

5.6 Company F

Company F is Cybercom; a Swedish company that operates mostly in Sweden, Finland, Poland, and India but also have offices in many countries. There are over 1200 employee and in Finland round 300 and out of these 100 of the employees are in the data centre. The interviewee is Ilkka Tengvall, chief system architect and advisory consultant in the company. Their customers are wide spread into many business areas; they are into web development, web hosting for the most visited Finnish sites, 2 out of the top3 and several out of the top 10, embedded solutions for industries, data centre services and other types of services.

5.6.1 Methodology Used in the Company

The interview information on methodology is mostly about his knowledge in the data centre because he works at the data centre of the company. The company functions under Agile and the data centre uses Scrum methodology, which has been in use for over a year. All the scrum principles and characteristics are implemented in the development projects; including the roles; artefacts and ceremonies. The size of the scrum team varies between 3-8 employees plus the product owner, the project timeline varies depending on the development project itself.
From the perspective of the interviewee, the advantages for implementing Scrum are:

> I think it's a decision to do more effectively. Develop more effectively. Have information flowing better, like the daily meetings help and force u you to share information, know what the others are doing and the other reason for having scrum is having visibility into project to know what's been delayed, what's progressing well (Tengvall Ilkka, 07 Nov. 2013).

Defining scrum characteristics to better suit project implementation is seen as the most problem faced while implementing Scrum.

> We had a lot of difficulties and here it's not that bad and its working but it also needs time for everyone to get accustomed to scrum method so in the beginning it doesn't work as well as it should. People don't use the tools similar ways and sizes of the work items are not matching each other. So we are doing a lot better than we did ½ a year ago (Tengvall Ilkka, 07 Nov. 2013)

Tools used in software developments are; Jenkins for Continuous integration, Celenium as a testing tool. Git is used and GitHub. For issue tracking proprietary software is being used and the IRC gallery and mailing list is used as the information channel. The licenses used are Apache, GPL, MIL among others which are dependent on the open source code pulled from the community. The customer decides finally what license to use of his/her ready produce.

5.7 Advantages of Open Source

Based on the interviews the advantages of open source approach to developing software's are effective enough to keep open source software's developments a growing trend in the business information technology sphere. These are what the interviewees gave about the advantages of open source approach:

1. No license fees, (Sundell Timo, 2 Oct. 2013, Brower Jason, 14 Oct. 2013).

2. No vendor locking; for the customers, this means freedom, they can always change their vendor whenever they want to (Sundell Timo, 2 Oct. 2013).

3. The customer can also participate in the development processes (Sundell Timo, 2 Oct. 2013).

4. As our software is open source and have GPL license. That means that we can use other GPL and open source components in our software without any trouble (Tötterman Johan, 2 Oct. 2013).

5. The price tag on our software is 0 and you can fork it or download it anytime you want and if you have technical skills and you can use it without any of our services. We are not aware of how many companies in Finland are using our software without us even knowing about it but maybe an handful at least and some of them have been contributing also to the code (Tötterman Johan, 2. Oct. 2013)

6. More secure, better information security because if there is security holes in the product, it can be found much faster because the code if free because everyone can read it (Lehtomaa Kari, 8 Oct. 2013).

7. Active forums; for example; python programming have active forums where you can get help and it's free (Brower Jason, 14 Oct. 2013).

8. Better Resource Pulling
   From the customer's point of view, if you think about it that the city wants to develop an It system and one of the biggest IT systems are patients-doctors information systems. Those take millions to develop. If you think about it, city goes and creates a tender, some big company wins and then it's a black box, the city first of all doesn't get to see the codes so often as they would like and then they are doing something, paying millions of dollars after this city, another city needs also a patient-doctor information system and they are putting 2 million dollars and then a third city needs the a new patient-doctor information system. From the big company point of view, this makes business sense, because they lock their clients in. They are not allowed anymore by license or virtue of mass of code, they are not able to develop it anywhere else. But if the first city came and said, we are not doing this alone, hey, other cities can join in, lets pull our cash together and make an open source system and anybody in the world can use and then other cities from around the world also say that we want to have the same system, what will happen is that, transparency, that is very big in open source (Artem Daniliants, 14 Oct. 2013).

9. Transparency
   The big company could get away with really bad code but if it was an open source project, some guru sitting at home can say, oh mine, that is horrible , how can you allow something like this, ok, here is a patch. And having a huge system like that, some security guru could look and say oh mine that is really bad (Artem Daniliants, 14 Oct. 2013).

10. Power of community
    You have something at your disposal that thousands of people use, it goes through rigorous security checks and there is so much functionally built in, and it would make sense for us to commit to the bigger project and commit to their code to profit everybody (Artem Daniliants, 14 Oct. 2013).

11. Quality Control

    Transparency brings quality control in itself but in open source definitely I would say process is usually better, because in open source there is no management and there are no code talks. If it's broken, commit a patch or don't complain. It is a different methodology; there is no room for politics. We either create good codes or we keep quiet (Artem Daniliants, 14 Oct. 2013).

12. The power of forking

    In open source the guys can say that we are going to take this and make our own thing. And there are hundreds of projects in open source world that happens due to forking. For example, an operating system Free-BSD and there are so many forks that came in from Free-BSD that wouldn't be possible today if it wasn't for the ability to fork and open source license models allow and encourage forking. Ofcourse forking is expensive in terms of resources because you are splitting resources but still it's the last resource weapon against the tyrants. So if the big company said, we are going to do our own version, you guys can leave, then they can say we can do our own thing and the most important thing is that they can still get the patches but they can have their own mixes. So in that sense, it a powerful model that allows for better results and less politics and better transparency (Artem Daniliants, 14 Oct. 2013). .

13. Community way of working

    Easiness of getting thousands of users/ developers around world in developing, testing, reporting bugs, no need for reinventing already models (Tengvall Ilkka, 07 Nov. 2013).

14. The freedom in open source

    Personally I love the freedom to be able to modify things to our need and do whatever we ant with it. Lets say with openstack we are on the driver's sit, we deside what to do with the software, how we use it if we want to modify them and we are totally free to do so and we dont need to pay someone to do the changes or someone blackmailing money from us to implement something (Tengvall Ilkka, 07 Nov. 2013).

5.8 Disadvantages of Open source Software

These are excerpt from the interviews on the disadvantages of open source.

1. Sometimes the software is in Alpha stage, there is a lot of problems with it and to make translations, for example to translate to Finnish language can be problematic. I have done some CRM installations and finding Finnish translation can be quite hard (Sundell Timo, 2 Oct. 2013).

2. Sometimes it's hard for people to get into the community, because you are beginners, so if you are a beginner, it's hard to get there and be part of that, and it takes a lot of work to be part of that community (Brower Jason, 14 Oct. 2013) .

3. Its programmer driven, there are not enough designers, User experience professionals. Usually open source software has great code, bad documentation and bad user interface because it's pure headed by programmers.Its almost impossible to join it because you don't know where to begin, you don't know how things are working but if it has appealing design and good documentation it can be amazingly successful and a good example of that is Ubuntu (Brower Jason, 14 Oct. 2013).

4. I think everybody accepts open source but not a lot of people accept open source license, so if for example if you want to create a student community and you use open source tools but let's say that you are a company that develops some sort of product that is based on open source, if the license of the open source software used to create a product is for example GPL, you have to release the source code. There are of course open source projects that have for example BSD license that we like alt, because BSD allows you to still keep it in a black box (Brower Jason, 14 Oct. 2013).

5. It's scary when you don't have somebody to hold responsible. For example you use open source software for mission critical systems and then all of a sudden its breaks and you are willing to pay a lot of money to somebody to fix it but its open source project, peoples do it maybe part time but that is been resolved now a days by consultancy companies but it scary to use for some mission critical, something that is developed as an hobby doesn't feel the same especially for a bigger company (Artem Daniliants, 14 Oct. 2013).

5.9 Factors Affecting Open source

1. Pre assumptions of open source means.

   Maybe there's alittle less nowadays. But there has been companies that are like scared of open source. They are like they don't understand it at all, so they are like a little bit scared of it. They don't know how it works. Maybe we have gotten an answer from somebody that sometimes that we don't have the guts to do this because we don't understand what open source is (Tötterman Johan, 2 Oct. 2013).

2. Less knowledge of our open source operates

   With our customers we see that for them its difficult to understand what areas are we different to the proprietary software. Because for example; Is there some kind of maintenance, let's say that there is a change in law in Finland that affects the software. Do we need to do that for current customers for free or not? It's not clear to them but for us it is clear because we know the software doesn't have any maintenance because it is "as it" (Sundell Timo, 2 Oct. 2013).

3. Trust Issues

   Peoples don't trust that open source code can be better than commercial software codes but it is also changing little by little. Big companies use money to push their commercial software to schools and government (Lehtomaa Kari, 8 Oct. 2013).

4. Lack of knowledge of the presence and use of open source

   Lots of people don't know that they are every day using open source software. For example when they are browsing the web they are using Apache web server, Face book is developed using open source software (Lehtomaa Kari, 8 Oct. 2013).

5. Marketing Issue

   I think the biggest problem with open source is that there is no marketing team. The people that are using it like me aren't really marketing people. There are people who program all day and they have no clue how to market it, in fact they think they can just ignore them entirely and the only thing that open source have to show for itself is for the amazing software but it has nobody to step forward and tell about it, so what you get is a community that's really amazing. In a world where there is such heavy marketing, we are entirely invisible, we sits in the back (Brower Jason, 14 Oct. 2013).

6. The open source Image

   When I go and tell somebody I work with Linux, have you heard about it, they are like No, I haven't heard of it, so I'm like you've heard of Mac, you've heard of windows, it's just like that, so they may say.. is it like generic brand, so I say, no and you don't pay anything for it so it must be the cheaper version and that's the philosophy that some people get and that's really sad. Linux is used more than any other operating system worldwide, all of the other operating system combined still do not get to the level of which Linux is been used(Brower Jason, 14 Oct. 2013).

7. Problem solving

   When you create a software for example in Lean startup, you are working towards what people don't see that they need, you have to be persuaded that you need something that the idea of marketing, so in Lean startup you develop the open source software to solve a need while open source software itself is not built to solve a need rather to support a general infrastructure (Brower Jason, 14 Oct. 2013).

5.10 Why the Open Source Approach?

The interviewees views on why they choose to do open source is an added bonus to why open source business sector is growing and this is what they have to say.

1. It was never a business decision. It was always a decision based on a feeling because as the CEO mentioned the project was like an internal software project for the wholesale company, that IT department and it was never meant to be business, so it was just based on a feeling that lets do it open source cus the whole platform that we are building it on is open source and that's it (Sundell Timo, 2 Oct. 2013).

2. We don't get any license fee and we don't want any license fees , our business model is based on service , so that is why we are talking about money in the software development because all the development is paid by customer . There is no

new feature coming on the system if someone is not paying for it (Tötterman Johan, 2 Oct. 2013).

3. Because it makes more money and money is a big value and it's easier to get, it's easier to start, I can say I use web2py framework and it is open source and people just use it, they can just get on and start using it and they come back and say I like this web tool and its really cool (Brower Jason, 14 Oct. 2013).

4. In terms of developing environments, the leading proprietary developing environment are big and you will need DVDs to install them and it takes time and have bugs in them. With the framework I'm working on, are very compact, that includes the framework that I'm developing in, the version control, the IDE and the server that runs it. This is a question of time and hours means a lot (Brower Jason, 14 Oct. 2013).

5. I didn't have to pay for anybody who specializes in deployment , in bigger companies they have a deployment guy, all he does is installs everybody application to other people's computers, that's insane, I don't need to do that in my company. The code is open, if someone needs it; it's easy to reach, so for me the open source philosophy is much better (Brower Jason, 14 Oct. 2013).

6. We do open source projects to learn, to provide something back to the community. We don't have enough momentum to create a project that would be taken by others, so what we do is, we release code and hope that someone might find it useful so they don't have to reinvent the whole thing again and because we use open source software, so we feel it's kind of take a little, give a little (Artem Daniliants, 14 Oct. 2013).

7. It is also a nice feeling to use the community of the open source. As stated before we use Drupal and Drupal has more than 150,000 modules which we can just download, put into our project with very little work to be done on it as the customer wants it and then move on (Cristian Andrei, 23 Oct. 2013).

8. It brings value to us, since we don't have to invest the time to reinvent and it brings value to the customer because we don't have to spend time to reinvent the function and make it work with the site and then who doesn't want to pay less for a lot of functionality (Ilkka Tengvall, 7 Nov. 2013).

6 Conclusion

In today's highly globalized business scene, the ability to adapt and innovate is one of the major keys to survive. Methodologies play an essential role in every software development process as indicated as from the thesis project results and without methodologies, project would certainly become chaotic, unrealistic and can become be unpurposeful. This study has proven that the role of methodologies in implementing open source software developments is immeasurable. From trying to improve the image of open source development processes, better the rate of open source businesses spring up to empowering the whole world in shifting toward becoming more open sourced.

This thesis project has achieved what the research questions have set out to accomplish. The research question has wanted to gain knowledge on what methodologies are used in open source software development and the six companies that have participated in this research has given a thorough descriptions as to what methodologies are being used , why this specific methodology and how are these methodology being implemented in the development process. This thesis shows the complexity of how methodologies function in development process. Most of the companies in this research have a sort of hybrid methodology in use, that is; a methodology that has been crafted to suit how that specific company do open source software development. One company still implements a modified waterfall method, which is an heavy methodology and it is functioning for them in that environment, others are into agile method or a mix of agile and another methodology which is seen as light weighted that also functions in that environment.

This thesis project has also brought to light the important role that Git and Github plays in the software development as an invaluable tool. Since projects success also depends on chosen tools that support the development processes; Git and Github and other open source development project tools

The choice of what methodology and tools is been used in open source software development is essentially driven by what type of development activity does the company implement and methodologies are not meant to be rigid but rather meant to support, guide and empower to capture and delivering the best possible customer solution possible.

6.1 Future Research

This thesis research stands as the building block for a larger project research that covers more companies is needed to get a broader view into methodologies used in open source software development. Also in the future, looking into what roles does methodology play in the success/ failure of open source software development processes can contribute in understanding this phenomenon.
It would be a very added bonus to have a research on powerful has open source been supporting start-up businesses around the world.

6.2 Personal Learning

This thesis project was very personal to me as I have always been fascinated with methodologies generally. Now at the end of the project, reflecting on areas that i could have done better, I think the scope of the project could have been abit wider and the sample size could have been larger but there is time frame restriction that is limiting in accomplishing this.

Going through this whole process of doing this research, has intensely broaden my view on the business opportunity lurking in open source software, empowered me in understanding why does the world need open source and what are the impact of open source in the nearest future. I believe that as human being, one of the powers we still possesses today, is the power in making choices and open source software has proven to be as competent, maybe even better than the leading proprietary software out there. The power to choose is in our hands and I personally believe that open source has just

started to evolve in a way that makes the world listen.   One of the most empowering experience for me during this project, is getting to meet open source enthusiasts out there and even more so , some of them are alumni of Haaga-Helia that are so passionate about open source, i want to say thank you, i was really inspired by them.
Finally, a saying a heard from an open source enthusiast; information is out there, it does not depreciate and the more it is out there, the more becomes better.

References

Christoph Treude, Margaret-Anne Storey & Jens Weber 2009. Emphirical Research on Collaboration in Software Development: A Systematic Literature Review. Technical Report DCS-331-IR. Department of Computer Science, Univerity of Victoria. URL : http://ctreude.files.wordpress.com/2009/12/tr-collaborationsoftwaredevelopmentsystematicreview.pdf. Accessed: 13 Oct 2013.

Sanjiva, Weerawarana & Jivaka, Weeratunga 2004. Open Source in Developing Countries. Sida URL : http://www.eldis.org/fulltext/opensource.pdf. Accessed 21 Jul 2013.

Eugene Eric. Kim 2003. An Introduction to Open Source Communities, Blue Oxen Associates, Technical report, BOA-00007. URL : http://flosshub.org/system/files/blueoxen.pdf. Accessed: 11 Sept.2013.

Rico D.F 2005. Short History of Software Methods. URL: http://davidfrico.com/rico04e.pdf Accessed: 14 March 2012.

Audris Mockus, Roy T Fielding & James Herbsleb 2002. Two case studies of Open Source Software Development : Apache and Mozilla. ACM Transactions on Software Engineering and Methdology. Vol. 11, No 3, July 2002. 309-346. URL: http://mockus.us/papers/mozilla.pdf Accessed: 26 Sept 2013.

Mark Saunders, Philip Lewis & Adrian Thornhill 2012. Research Methods for Students. 6th ed,. pp 163-176.

Denzin, N.K & Lincoln, Y.S 2005. The Sage Handbook of Qualitative Research. 3rd ed,. pp 3. Sage Publications. London.

Margaret C Harrell & Melissa A . Bradley 2009. Data Collection Methods. Semi-Structured Interviews and Focus Groups. National Defense Research Institute.

RAND. pp. 30-47. URL:
http://www.rand.org/content/dam/rand/pubs/technical_reports/2009/RAND_TR718.pdf . Accessed: 12 Sept. 2013.

Merriam-webster dictionary online. URL: http://www.merriam-webster.com/dictionary/methodology. Accessed: 24 July 2013.

Hamid Faridani 2011. A Guide to Selecting Software Development Methoddologies. pp 4- 11. URL :
http://www.gtislig.org/HamidFaridani_GuideToSelectingSWMethodologies_SOC_PDD_20110305.pdf. Accessed: 12 June 2013.

Robert C. Allen 1983. Collective Invention. University of British Columbia, Vancouver, Canada. Journal of Economic Behaviour and Organization 4 (1983) 1-24 North-Holland. URL: http://www.nuffield.ox.ac.uk/users/allen/collinvent.pdf Accessed 27 Sept. 2013.

George Siemens 2003. Free and Open Source Movements. URL :
http://www.elearnspace.org/Articles/open_source_part_1.htm Accessed 27 Sept. 2013.

Nathan Newman 1999. The Origins and future of Open Source Software. A NetAction White Paper. URL: http://www.netaction.org/opensrc/future/oss-whole.html Accessed: 27 Sept. 2013.

Kim Johnson 2001. A Descriptive Process Model for Open-Source Software Development. URL:
http://dspace.ucalgary.ca/bitstream/1880/41007/1/2001_Johnson,%20Kim.pdf. Accsessed: 10 Oct. 2013.

Cadle J. & Yeates,D. 2001. Project Management For Information Systems. 3rd ed. pp. 51. Prentice Hall, UK.

Khan A.I., Qurashi R.J., & Khan U.A, 2011. A Comprehensive Study of Commonly Practised Heavy and Light Weight Software Methdologies. IJCSI International Journal of Computer Science Issues, Vol 8, Issue 4, No 2, July 2011. ISSN (Online): 1694-0814. Pp. 442-444. URL: http://arxiv.org/ftp/arxiv/papers/1111/1111.3001.pdf Accessed 22 Jun 2013.

Per Kroll & Philippe Kruchten 2003.The Rational Unified Process Made Easy: A Practitioners Guide to the RUP.Object Technology Series. Addison-Wesley

RUP content dimension URL : http://medlem.spray.se/perlin27/rup.htm. Accessed: 14 Aug.2013.

Rational Unified Process 1998. Best practices for Software Development Teams. Rational Software White Paper TP026B, Rev 11/ 01. URL: http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf Accessed: 10 Aug. 2013

Pavan Kumar Gorakivi 2009. Build Your Project Using Rational Unified Process. IPMA. URL : http://www.asapm.org/asapmag/articles/A7_AboutRUP.pdf. Accessed: 10 Aug. 2013.

Charvat, J. 2003. Project Management Methodologies. John Wiley & Sons Inc. United States. pp. 9-127.

Agile Manifesto. URL : http://agilemanifesto.org/ Accessed 14 Aug. 2013

Jeff Sutherland. The scrum papers: Nut, Bolts and Origins of an agile Framework. 2012: URL : http://jeffsutherland.com/ScrumPapers.pdf#page=1&zoom=75,0,80 Accesses: 24 May 2013.

Ken Schwaber & Mike Beedle 2001. Agile software Development with scrum (Series in Agile Software Development).

Mountaingoatsoftarew.com URL:
http://www.mountaingoatsoftware.com/topics/scrum Accessed 14 Aug.2013

Ibrahim Haddad 2011. Understanding the Open Source Development Model. A white Paper by the Linux Foundation. URL:
http://home.deib.polimi.it/bellasi/lib/exe/fetch.php?media=docs:lf_os_dev_model.pdf Accessed. 11 Sept.2013.

Ko Kuwabara 2000. Linux: A Bazaar at the Edge of Chaos. First Monday,Peer-reviewed journal on the Internet. Volume 5, Number 3. URL :
http://firstmonday.org/article/view/731/640 Accessed: 01.Oct. 2013.

Eric S. Raymond 1999. Open Sources. Voice form the Open Source Revolution. O reiley online Catalogue. First edition. URL :
http://oreilly.com/catalog/opensources/chapter/ch01.html Accessed: 27Sept. 2013.

Opensource.org. URL: http://opensource.org/about Accessed 12 Aug. 2013.

Gnu.org URL: http://www.gnu.org/gnu/thegnuproject.html. Accessed: 12.Aug.2013

Open source.org URL: http://opensource.org/osd Accessed: 12.Aug.2013.

John Turkey; Coined the Words Software and Bit 2000.URL :
http://articles.latimes.com/2000/jul/29/local/me-61253. Accessed: 12.Aug. 2013

John Dooley 2011. Software Development and Professional Practices. Writing Great Programs from Start to Finish. Apress 2011.

Jain, D, 2007. Importance of Processes and Standards in Software Development URL: http://www.codeproject.com/Articles/17121/Importance-of-Processes-and-Standards-in-Software Accessed: 12.April.2012.

System Software. URL: http://home.olemiss.edu/~misbook/sfsysfm.htm Accessed: 01.Aug.2013.

ChinUsa URL : http://www.osait.com/event/197/txt/Types-of-Software-Systems.html Accessed: 01.Aug.2013.

Wheeler D.A., Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers! 2003 URL: http://www.dwheeler.com/oss_fs_why.html. Accessed 15.Aug.2013

Web server survey 2007. URL: http://news.netcraft.com/archives/2007/04/02/april_2007_web_server_survey.html Accessed 15 Aug.2013.

Jacques Longchamp 2005. Open Source Software Development Process Modelling, S.T. Acuna, N. Juristo (Eds), Springer, 29-64. URL: http://www.loria.fr/~jloncham/OSSChapter-Revised.pdf Accessed: 11 Sept.2013.

Ying Zhou & Joseph Davis 2005. Open source software reliability model: an empirical approach. Open Source Application Spaces: Fifth Workshop on Open Source Software Engineering (5-WOSSE) May 17, 2005, St Louis, MO, USA. URL: http://isr.uci.edu/~wscacchi/Papers/WOSSE-2005/ZhouDavis.pdf. Accessed: 24 Sept. 2013

Auer Liisa, Juntunen Jouni & Ojala Pekka (2011). Open Source Project as a Pedagogical tool in Higher Education. Mind Trek 11, September 28-30 2011. Tampere Finland. URL : http://dl.acm.org/citation.cfm?id=2181073 Accessed: 12 Sept 2013.

Rich Gordon 2013. 6 Things to Know about Successful (and Failed) open source software. URL: http://www.pbs.org/idealab/2013/08/6-things-to-know-about-successful-open-source-software Accessed 16 Aug.2013.

Charles Schweik and Robert English. 2012. Internet Success: A Study of Open-Source Software Commons. The MIT Press, Cambridge, MA 02142.

Chorng-Guang Wu 2007. An Empirical Analysis of Open Source Software Developers Motivation Using Expectancy- Valence Theory. University of Colorado at Denver and Health sciences Centre. URL: http://dl.acm.org/citation.cfm?id=1354430 Accessed: 16 Aug. 2013

Yiyun Shen 2008. Software Engineering Challenges in Small Companies. Helsinki 04.04.2008. Seminar Report. University of Helsinki. Department of Computer Science. Pp.4. URL: http://www.cs.helsinki.fi/u/paakki/Shen.pdf Accessed: 17.Sept.2013

Bill Appelbe 2003. The Future of Open Source Software. Victorian Partnership for Advanced Computing. Carlton South, Victoria 3053, Australia. Journal of Research and Practice in Information Technology Vol, 35 No 4, November 2003. Pp.233. URL: http://ws.acs.org.au/jrpit/JRPITVolumes/JRPIT35/JRPIT35.4.227.pdf
Accessed 16 Aug.2013.

Jimmy Atkinson 2013. The Top 50 Proprietary Programs that Drive You Crazy — and Their Open Source Alternatives. URL: http://whdb.com/blog/2008/the-top-50-proprietary-programs-that-drive-you-crazy-and-their-open-source-alternatives/ Accessed 16 Aug. 2013.

Vivek Venugopalan 2001. Software Configuration Management for Open Source Projects. URL: http://www.ibiblio.org/gferg/ldp/SCM-OpenSource/index.html Accessed: 27 Sept.2013.

Andre van der Hoek 2000. Configuring Management and Open Source Projects, Third Workshop on Software Rngineering over the Internet. Pp 41-45. URL:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.5292&rep=rep1&type=pdf Accessed: 27 Sept.2013

Yuhoon Ki and Meongchul Song 2009. An Open Source Based Approach to Software Development Infrastructures. IEEE/ACM International Conference in Automated Software Engineering. Advance Softwrae Research Center. URL: http://www.computer.org/csdl/proceedings/ase/2009/3891/00/3891a525-abs.html Accessd: 30 Sept.2013.

Walt Scacchi 2007. Free/Open Source Software Development: Recent Research Results and Emerging Opportunities. Institute for Software Research. University of Califfornia Irvine. ESEC/ FSE 07. Cavtat near Dubrovnik Croatia. Pp 459- 468. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.2442&rep=rep1&type=pdf Accessed: 30 Sept.2013.

Testtools.org. URL : http://test-tools.technologyevaluation.com/ Accessed: 30 Sept. 2013.

Cameron Chapman, 2010. A short guide to Open Source and similar Licenses. URL: http://www.smashingmagazine.com/2010/03/24/a-short-guide-to-open-source-and-similar-licenses/ Accessed: 16Aug.2013.

Opensource.org 2006. Report of License Proliferation Committee and draft FAQ. URL : http://opensource.org/proliferation-report Accessed 16 Aug.2013.

Juho Lindman, Matti Rossi, and Anna Puustell, (2011). Matching Open Source Software Licenses with Corresponding Business Models. Focus: Software Business Aalto University School of Economics. July/August 2011. IEEE Software (31-35). URL : http://search.proquest.com/docview/873310168/1402498BC4710BEFC97/4?accountid=27436 Accessed: 27 Aug.2013.

Gwyn Firth Murray (2011). Categorization of Open Source Licenses: More Than Just Semantics. The Computer and Internet Lawyer. Volume 26, Number 1. January 2009. URL :
http://search.proquest.com/docview/222875590/fulltextPDF/1402498BC4710BEFC97/1?accountid=27436 Accessed: 27 Aug.2013.

Tristan Kromer (2013). Lean Startup circle wiki URL:
http://leanstartup.pbworks.com/w/page/65946049/Intro%20to%20Lean%20Startup Accessed 10 Nov.2013.
Eric Ries. (2008) Startup Lessons Learned. URL:
http://www.startuplessonslearned.com/2008/09/lean-startup.html Accessed: 10 Nov.2013

Catherine A. Fritz & Bradley D. Carter 1994. A Classification and Summary of Software Evaluation and Selection Methodologies. Computer Science Technical Report No. 940823. Department of Computer science. Mississippi State University. Mississippi State, MS 39762. URL:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.55.4470&rep=rep1&type=pdf Accessed 14 Aug.2013.

Khan A.I., Qurashi R.J.& Khan U.A, 2011. A Comprehensive Study of Commonly Practised Heavy and Light Eeight Software Methdologies. IJCSI International Journal of Computer Science Issues, Vol 8, Issue 4, No 2, July 2011. ISSN (Online): 1694-0814. Pp. 442-444. URL: http://arxiv.org/ftp/arxiv/papers/1111/1111.3001.pdf Accessed 22. Accessed: 10 Aug. 2013.

Rothi, J & Yen, D. 1989. System Analysis and Design in End User Developers Applications. Journal of Information Systems Education. URL:
http://www.gise.org/JISE/Vol1-5/SYSTEMAN.htm Accessed: 24.April 2013

Rico D.F 2005. Short History of Software Methods URL:

http://davidfrico.com/rico04e.pdf Accessed: 14.March 2013

Lehtomaa Kari. 8 Oct. 2013. Ex-employee in an IT company. Interview

Sundell Timo & Johan Tötterman. CEO and Developer 2 Oct. 2013. Devlab. Interview.

Brower Jason. 14 Oct. 2013. CEO Encompass Solutions Oy. Interview.

Artem Daniliants 14 Oct. 2013. CEO Lumolink. Interview.

Cristian Andrei 23 Oct. 2013. Developer. Wunderkraut. Interview.

Tengvall Ilkka 7 Nov. 2013. Chief Architect . Cybercom. Interview

Attachments

Attachment 1. Interview Question

1. A short introduction of the company.

2. Type of software do you develop

3. Have you used any of the following methodology in an open source development project? If YES, any of the following
    i. Waterfall Model
    ii. The Rational Unified Process Model (RUP)
    iii. Agile
        1. Scrum
    iv. Open Source Development model
    v. The cathedral and the Bazaar

4. What are the advantages of this chosen software development methodologies?

5. What are the disadvantages of this methodology?

6. What open source tools do you use for developing software?

7. What are the advantages of doing open source?

8. What are the disadvantages of open source?