WHICH AGILE SOFTWARE DEVELOPMENT METHODOLOGY SUITS YOU?

– BY APPLYING THE RESULTS ON A PROJECT OF A SMALL Sized COMPANY.
Title: Which agile methodology suits you? By applying the results on a multi-disciplinary project in a small company

Year: Summer 2011

Author: Fatemeh Saadatmand

Supervisor: Bertil Lind

Abstract:

Choosing the Software Development Methodology is the very first step of any project; thus, has been a hot topic among, both, practitioners and academic people. After using plan-driven software development methodologies software development researchers came up with the idea of agile software development methodologies as a masterpiece. Although, failure stories of some teams brought about fading the idea that agile methodologies are the best recipe for any kind of development project. Considering the lack of studies in helping practitioners to select the most appropriate agile software methodology, this study aims at provide the software development manager with a thorough knowledge of agile methodologies and the criteria that should be considered, while selecting one of them. A case study is used as an empirical support.

Keywords: Software development methodologies, Agile, XP, Scrum, Crystal, ASD, OSS, AM, PP, DSDM, RUP, FDD, select agile methodology
Acknowledgements

This study presents the results of a master thesis research in the field of Informatics in Borås University (Sweden). The purposes of the study is to provide a software development manager with a guide to help him/her to select the most appropriate agile software development methodology for the project he/she has in hand.

I would like to thank Hannes Göbel the manager of the Innovationlab who introduced this project to me and helped me to find my desired way through this project. Besides, I would like to thank my supervisor Bertil Lind for his patience during this master thesis and Anders Hjalmarsson, my examiner, who helped me to improve my work with his valuable comments. Additionally, I would like to thank all of my tutors under this master program who helped me to have a better perspective over the Informatics topics and IT world by leveraging my knowledge in this field.

Finally, I would like to thank my family who supported me to make my dream of studying abroad come true, especially my father Ahmad Reza and my mother Mahkhanum for their outstanding support.
Table of Contents

Table of figures .................................................................................................................................... 7

Table of tables ..................................................................................................................................... 7

1 Introduction ......................................................................................................................................... 8

1.2 Background .................................................................................................................................. 8

1.3 Statement of problem .................................................................................................................. 9

1.4 Purpose of the study ................................................................................................................... 9

1.5 Research questions ................................................................................................................... 10

1.6 Target group .................................................................................................................................. 10

1.6.1 Innovationlab: The sample company .................................................................................... 10

1.6.2 Lohorung: The sample project ............................................................................................... 11

1.7 Expected outcome ....................................................................................................................... 12

1.8 The authors’ own experience and background ......................................................................... 12

1.9 Structure of the thesis ................................................................................................................ 13

2 RESEARCH DESIGN ....................................................................................................................... 14

2.1 Research perspective .................................................................................................................. 14

2.2 Research strategy ....................................................................................................................... 15

2.3 Data collection procedures ...................................................................................................... 16

2.4 Data analysis procedures .......................................................................................................... 17

2.5 Result Evaluation ....................................................................................................................... 17

2.6 Result presentation method ....................................................................................................... 17

3 THEORETICAL STUDY .................................................................................................................. 18

3.1 Key concepts ............................................................................................................................... 18

3.2 Subject areas relevant for the research ..................................................................................... 20

3.2.1 Software methodologies ....................................................................................................... 20

3.2.2 Extreme Programming (XP) ................................................................................................. 21
3.2.3 Scrum

3.2.4 Crystal family of methodology

3.2.5 Feature Driven Development

3.2.6 Rational Unified Process

3.2.7 Dynamic Systems Development Method

3.2.8 Adaptive Software Development

3.2.9 Open Source Software Development

3.2.10 Agile modeling

3.2.11 Programmatic Programming

3.2.12 Agiles: Comparison Approach

3.3 Previous researches

3.4 Literature sources

3.5 Agile Software Development Methodologies (Deeper look and comparison)

3.5.1 Extreme Programming

3.5.2 Scrum

3.5.3 Crystal Family of Methodology

3.5.4 Feature Driven Development

3.5.5 Rational Unified Process

3.5.6 Dynamic System Development Method

3.5.7 Adaptive Software Development

3.5.8 Open Source Software Development

3.5.9 Agiles: Comparison approach

3.6 Summary of theoretical findings

3.7 Arguments for an empirical study

4 EMPIRICAL SURVEY

4.1 Purpose
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Sampling</td>
<td>65</td>
</tr>
<tr>
<td>4.3</td>
<td>Interview with the manager</td>
<td>65</td>
</tr>
<tr>
<td>4.4</td>
<td>Empirical research results</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>ANALYSIS AND RESULT</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>Analysis</td>
<td>68</td>
</tr>
<tr>
<td>5.2</td>
<td>Result summary</td>
<td>69</td>
</tr>
<tr>
<td>6</td>
<td>DISCUSSION</td>
<td>70</td>
</tr>
<tr>
<td>6.1</td>
<td>Conclusions</td>
<td>70</td>
</tr>
<tr>
<td>6.2</td>
<td>Implications for Informatics</td>
<td>71</td>
</tr>
<tr>
<td>6.3</td>
<td>Method evaluation</td>
<td>71</td>
</tr>
<tr>
<td>6.4</td>
<td>Result evaluation</td>
<td>71</td>
</tr>
<tr>
<td>6.5</td>
<td>Possibilities to generalize</td>
<td>72</td>
</tr>
<tr>
<td>6.6</td>
<td>Ideas for continued research</td>
<td>73</td>
</tr>
<tr>
<td>7</td>
<td>References</td>
<td>74</td>
</tr>
</tbody>
</table>
Table of figures

Figure 1 Dimensions of Crystal Methodology ................................................................................... 24
Figure 2 Extreme Programming lifecycle .......................................................................................... 31
Figure 3 Scrum Process ..................................................................................................................... 36
Figure 4 One Crystal Orange Increment (Abrahamsson et al. 2002) ................................................ 41
Figure 5 Process of FDD (Palmer & Felsing 2002) ............................................................................. 45
Figure 6 Components assembly in FDD ............................................................................................. 46
Figure 7 RUP phases .......................................................................................................................... 48
Figure 8 Major Milestones of the RUP .............................................................................................. 50
Figure 9 DSDM process diagram (Stapleton 1997, p.3) .................................................................... 51
Figure 10 DSDM feasibility study ...................................................................................................... 52
Figure 11 DSDM functional model ................................................................................................... 53
Figure 12 DSDM design and build ................................................................................................... 53
Figure 13 DSDM implement-deploy-maintain .................................................................................. 54
Figure 14 ASD cycle (Highsmith 2000) ............................................................................................ 56
Figure 15 Comparing methodologies regarding different phases of the project ......................... 61

Table of tables

Table 1 Comparing Agile methodologies ........................................................................................... 60
1 Introduction

1.2 Background

The IT industry owns one of the most dynamic business environments. Such an environment has caused continuous changes in skills, process flows, strategies and policies. As a result, there is a need of evolving in software development industry in order to meet the new requirement. Unfortunately, traditional plan-driven software development methodologies seem to be too disciplined to meet dynamic needs of the development process of these days.

Although the software development industry has seen such mass of varied methodologies during the recent 25 years, only a few could persuade some software people as practical methods. However, Object-Oriented method could beat up traditional methods by bringing about a feasible way to develop information incrementally (Nerur, Mahapatra & Mangalaraj 2005), but undoubtedly, agile methodologies or lightweight methodologies have gone even further in overcoming the complexities that is caused by dynamism. Considering the recent huge "Agile Wave", the reader can clearly see that agile methodologies has been one of the most successful or at least popular- among all the new comers, not only among software developers and software managers but also it has founded its fans among academics these days. On the other hand, the industry witnessed a lot of challenges during the process of migrating from traditional software methodologies to agile methodologies, which brought about writing different books and documents by software researchers.

There seem to be no need to discuss such massive change eXtreme programming (Beck 1999) made to the way of developing software. Such a success leads researchers to design some more models of this series to cover different kind of project structures or company structures e.g., Crystal Methods (Cockburn 2000), Feature-Driven Development (Palmer & Felsing 2002), and Adaptive Software Development(HighSmith 2000). Even some (Agile software development ecosystem) refused to call it Agile Software methodology and called it "Ecosystem" instead in order to fall in to a completely different classification other than traditional methodologies.
1.3 **Statement of problem**

Crystal clear, while choosing the most appropriate method, software project managers face different methodologies for different projects or even different methodologies for different life cycles of a single software project. Among existing methodologies, Agile methodologies have been so popular and have attracted the attention of any IT manager.

The manager should take any factor that play a valuable role in choosing the suitable methodology including the organizations structure, culture and management practices (Nerur, Mahapatra & Mangalaraj 2005) in to consideration.

On the other hand, despite such pervasive use of this methodology series and such high enthusiasm in the subject, in addition to, the existence of a large number of different agile methodologies, it lacks documentation in order to guide software managers to the most proper methodology that suits their company size and their project specifications.

1.4 **Purpose of the study**

In this thesis, the author aims to guide a project manager to choose the appropriate agile software development methodology for any software life cycle of any software project. In order to depict it in more practical way, the author has picked a real project from a small size software company that is in the pre-study phase to show how to choose a development methodology for it.

Finally, the process of choosing the most proper agile software development methodology is proposed to be a part of a pre-study template of any software project in the company.
1.5  Research questions

- What are the different characteristics of different agile methodologies which should take into consideration by the project manager while choosing software development methodology?

- Which factors of the software development project should a project manager focus on while choosing a software development methodology?

1.6  Target group

The target group consists of any software project manager in general. Besides, in specific, it can address the sample company, Innovationlab, and the sample Project, Lohorung, as target group, too.

1.6.1  Innovationlab: The sample company

Innovationlab is a small size company organization inside the University of Borås running varied development projects which can be divided into two principle groups:

- **Action Support:** Handling the affairs related to 15,000 students and 650 employees of the six departments of University of Borås can be challenging for the IT section of the University; thus, Innovationlab is responsible for development and maintenance of some of the tools useful for the university including the website of the university.

- **Software Production:** Although, Innovationlab is an organization inside the university it is considered as an independent software company. As a result, it can have its own customers both for producing and consulting.

Furthermore, Innovationlab - as you might have realized out of its name - is responsible about participating in research projects. The tasks Innovationlab would be responsible for can vary in a wide range starting from searching and giving ideas to producing useful tools.
According to the point that staff of Innovationlab consists of both practitioners-developers and academics-PhD students and the fact that Innovationlab resides inside the university which makes it accessible, the Innovationlab has turned out to be the best candidate to assist or run research projects.

1.6.2 Lohorung: The sample project

The goal of this project is to document and vitalize the Lohorung Language that is spoken in the Lohorung Rai community most of which is located in Sankhuwasabha, an eastern remote district of Nepal. Since Lohorung does not have a written language, this project aims to propose a modern language technology to establish a collaborative platform in order to document, preserve and vitalize Lohorung Language.

The project is being currently carried out in joint collaboration of the following partners:
- The Lohorung Community,
- Tribhuvan University, Nepal
- The University of Gothenburg, Sweden
- The University College of Borås, Sweden

Talking about the target group of this project, they can fall in to three different groups:
- The Lohorung community inhabitants: the benefit will be vitalizing their language and prevent it of dying out by means of documenting the language and developing a written format and a platform to vitalize the language. It is estimated that there are 30,000 Lohorungs in Nepal who are mostly concentrated in Lohorung villages that are near the district center Khandbari of the Sankhuwasabha district of Nepal. However, a considerable part of the Lohorung population has also migrated to big cities like Kathmandu and Dharan for better and more job opportunities. There are also many Lohorungs giving their service in the British Army, UK and other sectors in other Countries.
• Linguistics Survey of Nepal project which indirectly will benefit the Government of Nepal and the Nepali people.

• Researchers working on the same area, which can use the outcome of the project or the idea to the same kind of projects or for further studies.

1.7 Expected outcome

The expected outcome will be the proper agile software methodology for the Lohorung project. In addition, the attempt is that the proposed framework of the process of choosing the appropriate software methodology would have the capability of being applied to any software project in order to guide software project managers.

1.8 The authors’ own experience and background

After being suggested to work on the Lohorung project mostly on its pre-study by Hannes Göbel, the technical manager of Innovationlab, for my thesis, I explored the different possible aspects of the project that could be studied. I was looking for any technical points that might look so academic as well, and it took me such a time, I changed my mind between some different subjects after writing my proposal. What inspired me more about selecting methodologies as my thesis main subject was what I found, while reviewing my mails, at the end of Hannes' email that included the information about the project stating "try do something that you think is fun and where you will learn something - it is much more easier to write then...” During my work experience, I always found methodologies fun and also during my studies in bachelor degree- I studied software engineering for bachelor degree- In addition, I found it even more interesting after start working in Innovationlab as a software developer since I think it is the very first and very important step of the project which affect the project and the staff wholly.
1.9 Structure of the thesis

The author will start by defining each agile software methodology, considering the processes, scope of use and practices for each methodology. Next, the assessment criteria will be defined and each methodology will be assessed against all defined criteria and a comparison approach will be presented by means of the generated information.

Finally, a case-study is done in order to use the outcome of the previous part on a project in the real world that is feasible for agile; thus, an empirical support will be generated for the suggested "methodology selection process".
2 RESEARCH DESIGN

2.1 Research perspective

According to Smith (1986, p. 64) in attempting to combine empiricism and nationalism in a way of producing positivism:

“Logical positivism arose as the joint product of two intellectual traditions that conflicted deeply with one another. In attempting to unite these traditions, its adherents created an extremely influential approach to philosophy but one can embodied serious intellectual tensions from its dual ancestry.”

Although Hjørland (2004) claims that “At the beginning of the twenty first century not many theorists defend positivism, which does not imply, however, that the positivist way of thinking is not still influential as a kind of naive or silent philosophy”. He believes a good substitution for positivism that could have a notable position in research has not yet emerged.

Since this study is an amalgamation of theoretical and empirical part (which can be implied as reality and empiricism) the author finds positivism a good choice.

Although, a research that is done under a positivism approach can result in qualitative or quantitative data, under this study qualitative data is produced; In addition, a qualitative approach is utilized, since the most appropriate unit of measurement is not certain while evaluating different methodologies, hence, the researcher can choose between such wide variety of factors, there exists no clear demarcation points to measure methodologies, one of the most major factor involving the decision of choosing the software methodology is the human factor which makes the boundaries of characteristics vaguer and out of numbers. This is why the author has chosen qualitative approach in order to extract the results. Additionally,
among the qualitative approaches, the author has found screening mode presented by (Kitchenham 1996) and (Kitchenham & Jones 1997) the most proper strategy.

2.2 Research strategy

As stated earlier, the purpose of this study is guiding a software project manager to select the most proper software methodology for a given project; the question is which factors of a project should be considered while selecting an agile software methodology and how the selection will process work after selecting the desired features. In order to answer this question screening is used as the evaluation method and to make it more tangible and applicable a case study is used.

The theoretical part guides gives the reader enough information about each software methodology. Next, according to explained characteristics of each agile methodology features are selected to evaluate each methodology against it and, simultaneously, feed the screening mode with them. The analyzing results are presented in form of a comparison approach for agile methodologies. Actually the comparison approach is used to categorize such a high load of data to make it more understandable and usable. Besides, it has been done to make the way of choosing the methodology clearer.

To be more precise, under the theoretical part, the author thrives to shed some light on the subject and answer the research questions, Secondly tries to provide questions with primitive answers. It will be done by narrowing the domain of argumentation and categorizing the data about each method. After establishing the theoretical part, the generated process is used in the empirical part. The author attaches the empirical part to show how to utilize the knowledge that is created in the theoretical part, and also to answer the research questions more thoroughly.

Moreover, the author represents the empirical study to show how to overcome the risk factors that might pop up in projects in reality, since the situation of a company and its projects might not be as isolated as in theory.
To put it in a nutshell, the expected outcome from the theoretical part that is done with screen mode is a thorough knowledge of agile methodologies with a comprehensive comparison to make a good guide for the project manager, while the expected outcome from the empirical part is to show how this guide is useful for a project manager. As a result, the exact answer of the research question is answered in the empirical part while the knowledge in the empirical part comes out of the theoretical part.

2.3 Data collection procedures

The information about each agile methodology is collected among all existing documentation declaring each methodology. Since before starting to find the assessment criteria, it is needed to know each methodology thoroughly and take all limitations and characteristics that are presented from its founder into consideration therefore the documents are selected among original papers and books written by the founder of each methodology. Besides, since the outcome of this study would be used mostly by practitioners, the author has taken all documents extracted from empirical studies of each methodology in to consideration as well. Although, there are some comparison approaches for agile methodologies, the author found them uncompleted and more importantly out of date e.g. “Abrahamsson, Salo, Ronkainen & Warsta 2002”.

For the empirical samples, two important factors have been considered: The company and the project had to feasible for agile and the accessibility of the company. As it is clearly seen this is the most important factor for a company to work with agile methodology is to have the appropriate place for it. The suitable work place for agile is a place that all project members can communicate with each other easily (Undoubtedly, this characteristic will be even more pivotal if pair programming practices are involved with the methodology). On the other side, agile cannot be successful for all type of projects. The project has to be feasible with agile. To make a long story short, five different characteristics affect the agility of the project including size of the team, criticality of the project, the degree of Dynamism of the project, Culture of the company and level of personnel (Boehm & Turner 2003)
2.4 Data analysis procedures

The data for each methodology is collected. The author has presented and defined different criteria to be considered while choosing the methodology; thus, each methodology has been evaluated against the given criteria.

Finally the results are shown as a comparison approach and are interpreted in text, table and diagrams.

2.5 Result Evaluation

Each study would be evaluated with its result evaluation methods. It might be easier and more tangible to evaluate the results when a study utilizes quantitative data, under such a circumstance evaluation can be served via formulas, scoring, and etc, but it is not that obvious and routine strategy for qualitative data because they can be completely relative. Thus, the need of declaring result evaluation for qualitative data is more tangible.

Lincoln & Guda (1985) have suggested a set of criteria in order to evaluate studies, that are as follows: Truth value (truth of the findings are evaluate), Applicability, Consistency, and Neutrality.

2.6 Result presentation method

The author has used text in order to analyze the results while table and diagram are presented later in order to categorize methodologies. Moreover, a case study is done and is presented in the empirical study part in order to illustrate the practical use of the research results.
3 THEORETICAL STUDY

3.1 Key concepts

Below you will find relevant key concepts of this thesis. A brief explanation is given for each concept in order to shed some light over the rout the thesis is going through during upcoming chapters. The key concepts are as follows:

Software Development Methodology: indicates to the framework that is used to arrange, order, and direct the procedure of developing a software-developing project. A diverse set of these methodologies has progressed during recent decades with some diagnosed pros and cons for each of them (Roberts et al. 1998).

Agile: lexically means quick and well coordinated in movement, active. As a result "Agile Software methodologies" are trying to make the dream of IT people of having less complicated an addition to and active software development processes.

Software Development Lifecycle (SDLC): A lifecycle consists of all the different eras of software’s existence from its start stage with requirements definition to plantation and maintenance (Ruparelia 2010). There exist different models of software development lifecycles that according to Ruparelia (2010) they can fall in one of the following three categories as Rupardia (2010) states which includes linear, iterative or a combination of linear and iterative models. Ruparelia (2010) defines that a linear model is a sequential mode where finalization of one step, assuredly causes entering to the next step (Ruparelia 2010). An iterative model, differently, guarantees that every step of the model shall be called in the future (Ruparelia 2010); the philosophy behind these kinds is that development should always be in the process so that the progressing process will be an always- running process during the lifecycle. On the other side, the combined model uses iterative concept except that would recognize the point which iterative process should be terminated (Ruparelia 2010).
Screening mode: Is one of the methods of feature analysis evaluations in order to evaluate Software engineering methods and tools. Actually Kitchenham (1996) addresses qualitative evaluation method by feature analysis. Under the Feature Analysis process users' necessities related to the specific procedure are explored as the basic step of the process; hence, "mapping those requirements to features that a method/tool aimed at supporting that task/activity should possess" (Kitchenham 1996). Under this kind of assessment, users are supposed to be software programmer, software tester and software administrator. An assessor then evaluates how strong the considered features are arranged by a group of methods/tools.

"Feature Analysis is referred to as qualitative because it usually requires a subjective assessment of the relative importance of different features and how well a feature is implemented." (Kitchenham 1996). A single person, who recognizes the necessities, codify them to different features and then evaluate the amount to which method/tool support them by testing the methods or studying sales abstracts, can perform feature analysis. This method is more applicable when screening a huge amount of methods and tools.

**Screening mode Approach**

This form of qualitative evaluation is done by a single person based on documentation and is considered to have the lowest cost among different forms of Feature Analysis. It is recognized as the best approach for screening a huge amount of methods/tools (Kitchenham 1996) and is usually utilized as the first step of more complicated assessment exercises including reducing a huge amount of possible methods/tools to a brief record of one or two entrant methods/tools that is assessed more specifically.

According to (Kitchenham & Jones 1997) under this form of qualitative evaluation, the evaluator is in charge of:
- identifying the candidate methods/tools;
- devising the assessment criteria (i.e., the features to be assessed and the judgment scales) with or without specific
help from potential tools users;
• compiling information about the method/tools;
• scoring each feature for each method/tool;
• analyzing the scores;
• presenting a report on the evaluation.”

Advantages of this method are:
Since this method is designed and considered as the very first stage of the project, it is the quickest and cheapest type of the feature analysis.
On the other hand, the amount of varied roles is attenuated.

While disadvantages are:
The whole assessment is grounded on an individual’s (team’s) assessment; consequently, it can be distorted. In addition, The evaluator may not be representative of potential users of the method/tool. (Kitchenham & Jones 1997)

3.2 Subject areas relevant for the research

3.2.1 Software methodologies

Roberts et al. (1998) believe that a methodology can be referred as a “systematic approach to conducting at least one complete phase (e.g. requirements analysis, design) of system development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of system development and the target system” (Roberts et al. 1998) . Long story short, software development methodology is framework for developing an information system which the author is going to divide them in to two groups of Traditional and Agile in this research.

Agile Software Development Methodologies

Agile - which means nimbleness and active- software development methodologies try to make the development process faster and less hierarchical and complicated. Actually the nucleus point of agile
methodologies are directness and rapidness; hence, within any agile development methodology developers focus on the features necessary for the first step and attempt to deliver them as soon as possible, gather evaluations and comments and will improve by utilizing the gathered information.

Abrahamsson et al. (2002) believe that there are four characteristics that make a development method an agile one: These include: being Incremental, meaning short software releases in fast cycles, being cooperative: close and frequent interactions between customers and developers forms a cooperative development project, being straightforward which lies in the simplicity of the method that turns it to an easy-to-learn one and even easy to alter since it is documented thoroughly, and the forth characteristic is being adaptive that makes the software more flexible to the late changes.

Furthermore, the business requirements of the project and also the design specifications can be altered during the development in Agile methodologies (Lindstrom & Jeffries 2005) Agile SDMs share several features including prototyping, iterative development, and minimal documentation (Holmstrom et al. 2006)

Each methodology is going to be reviewed in a specified format. Within the format four criteria affected by (Abrahamsson, Salo, Ronkainen and Warsta 2002) that are as follows:

- process
- roles and responsibilities: involved in the software development process
- Practices are concrete activities that a method defines to be used in the process
- Adoption and experiences: how the method should be presented in an organization according to existing experience reports
- Scope of use: takes limitations of each method in to consideration

### 3.2.2 Extreme Programming (XP)

Every software developer or even everyone who has been a part of a development cycle undoubtedly has heard about XP, but it is good to know
that XP is way more than pair programming.

The problem and complication of having tiresome and extended cycles in traditional models led in to emergence of Extreme Programming (XP). Such cycles were not suitable for small-size development teams as a result Extreme Programming came in to picture as a lightweight methodology to deal with the special conditions of small-size development teams. It mainly questions different existing customary assumptions e.g. the charge for altering a part of the software undoubtedly increases severely after a period of time.

### 3.2.3 Scrum

Scrum is first presented in Takeuchi and Nonaka (1986) article which presents a rapid, flexible and self-organizing product development process arising from Japan (Schwaber & Beedle 2002). “The term *scrum* originally derives from a strategy in the game of rugby it denotes “getting an out-of-play ball back into the game” with teamwork” stated Schwaber & Beedle (2002).

This method has been presented mostly with the purpose of managing and handling the systems development process. It is an empirical method which has applied the idea of industrial process control theory to systems development that leads to a method that presents the idea of flexibility, adaptability and productivity (Schwaber & Beedle 2002).

On the other hand, it does not include any special method for implementing phase. Instead, Scrum aims at organize the team members so that the system is more flexible in a frequently changed environment. The focal point in Scrum is that systems development includes varied technical and environmental variables such as resources, time frame and technology that probably change during the process. Consequently, the produced system is more flexible and useful when delivered (Schwaber 1995)

Scrum can be a good choice for improving existing practices (e.g. testing practices) in a company. Scrum frequently identifies obstructions and
bottlenecks in the development process as well as current practices by involving consistent management activities (Schwaber 1995; Schwaber & Beedle 2002).

3.2.4 Crystal family of methodology

The Crystal family of methodologies is made by a number of different methodologies in order to make it possible for any single project to choose the appropriate methodology. In addition, this methodology series provides teams with the principles for tailoring the methodologies to be adoptable to different situations of varied projects.

Each methodology of this series has a color depending on how heavy is the methodology, i.e. the heavier the methodology is the darker the color will be. Thus, teams have to select the most appropriate methodology concerning the weight their project affords. One might wonder what the criteria for selecting the most appropriate methodology is. The answer is the size and criticality (figure 1). The larger the project is the heavier the methodology can be since it might need more coordination than smaller projects. On the other side, the more critical the project is the more precision is needed.

The characters shown in Figure 1 describe a potential loss caused by the system failure (i.e. criticality level): Comfort (C), Discretionary money (D), Essential money (E) and Life (L) (Cockburn 2002). Abrahamsson et al. (2002) explains “criticality level C indicates that a system crash due to defects causes a loss of comfort for the user whereas defects in a life critical system may literally cause loss of life”.

[23]
The dimensions of Figure 1 describe the size and criticality. As an example to understand the diagram, C20 represents a project with maximum of forty people working on a project of maximum criticality of Comfort.

3.2.5 Feature Driven Development

Another agile methodology is Feature Driven Development (FDD) which is a flexible one. The point about FDD is that it cannot be the single solution for whole the software development process, since it mostly focuses on the design and building phases (Palmer & Felsing 2002). On the other hand, since it is proposed to be used for software development projects, it does not need any process model. Success stories are mostly found in the industrial and financial projects. The quality aspects are in the center of concerns under the process by thorough observing on the regular and definite deliverables.

Five separated processes that are run in a row forms FDD that helps the
project people to hand over the system with guidance, techniques and approaches. Besides, FDD consists of artifacts, objectives, plans needed for a project. Palmer & Felsing (2002) claims that FDD is suitable for critical systems, which makes it noticeable since it is the only methodology among agiles that support critical systems.

FDD was first discussed in (Coad et al. 2000). Jeff Luca, Peter Coad & Stephan Palmer first applied it for a large lending system in Singapore.

3.2.6 Rational Unified Process

The Rational Unified Modeling abbreviated with RUP has been founded by Philippe Kruchten, Ivar Jacobsen and their colleagues at Rational Corporation in order to accomplish the famous Unified Modeling Language- or UML for short- that is a business-standard approach for software modeling and now is owned by IBM.

RUP is a strategy designed for object-oriented systems just like its father UML and is heavily based on use cases to pattern customer needs and create the framework for the system. The agility of RUP can be seen mostly in the iterative characteristic of the methodology, similar to other agile methodologies. RUP does not implicitly rule our other methods Jacobsen et al. (1994) believe and they think that it is adjusted enough for Object Oriented development.

3.2.7 Dynamic Systems Development Method

Since its origin in 1994, DSDM, short for the Dynamic Systems Development Method, has recently become the popular framework for rapid application development (RAD) in the Britain (Stapleton 1997). Recently, DSDM with its latest version DSDM Atern has turn in to a framework for Agile.

DSDM Atern is the proven and robust Agile framework for effective Project Management (DSDM Atern u.â.).
The mission of the not-for-profit DSDM Consortium is to be the guardian of DSDM, the Agile Project Framework and Full Delivery Method that provides best practice guidance for on time, in budget delivery of projects in any Business Sector (DSDM consortium u.å.).

The main idea behind DSDM is that the time and resources should be fixed at the first place and the next step can be adjusting the amount of functionality.

3.2.8 Adaptive Software Development

Adaptive Software Development, ASD as the abbreviation, was presented for the first time in (Highsmith 2000) by James A. Highsmith III. Highsmith started some research on iterative development strategies and ended up in the keystones of ASD. ASD has affected a lot by a former methodology presented by Highsmith and S. Bayer (Bayer & Highsmith 1994) named “RADical Software Development”.

Incremental, iterative development along with continual prototyping forms the basic ideology of ASD. Besides, thriving to find a solution for development process of complex and large systems is at the center of attention in ASD. It mainly tries to provide teams with a framework that can help them to avoid getting into problems or “chaos” as they call it. On the other side, ASD tries to keep the space for creativities to happen as well.

3.2.9 Open Source Software Development

In the 90’s development process of open source triggered forming the strategy of OSS and brought about a creative way to produce software. This paradigm allows the source code to be available for modification and redistribution for free.
Other Agile Methods
Recently the author has discussed different methodology of agile that has a specified framework with identified processes and practices. There are some other methods that are less documented and organized that will be presented in this section:

3.2.10 Agile modeling

Agile Modeling was seen in (Ambler 2002) for the first time and is presented as a “new approach for performing modeling activities” (Ambler 2002). It is attempted to form modeling practices with agility as their main characteristic, besides the cornerstone in designing has been on modeling practices and cultural changes. The aim has been described as below by Ambler (2002):

“The aim is to keep the amount of models and documentation as low as possible. Cultural issues are addressed by depicting ways to encourage communication and to organize team structures and ways of working”.

3.2.11 Programmatic Programming

Programmatic Programming by Andrew Hunt & David Thomas (2000), abbreviated with PP, introduces a set of programming “best practices” (Hunt & Davis 2000). The model is a set of 70 tips, rather short, which can be referred as a guideline for programmers to deal with daily issues. The tips mostly focus on the incremental aspect of design while they cover testing and centering the user.

3.2.12 Agiles: Comparison Approach

Due to the explanation of the items above and considering all limitations and variations, a comparison approach is conducted in two different aspects. Firstly, four different areas are selected to be assumed as assessment criteria named Size of the team, Scope, Team structure and Empirical support.
These four are selected as the most important factors a project manager should be aware of while selecting an agile methodology and match it with the appropriate and matched methodology.

On the other hand, considering life cycles, while choosing the methodology, has always been an issue in this field. Undoubtedly, a project manager should know if a lifecycle is covered by a methodology or not before choose it, let it be agile or not. Thus, life cycle, phases and possibility of having any extra help through cycles are considered as the most important ones through next step of comparison process.

The two above aspects are chosen by the writer based on the available information the author has collected about the methodologies and the importance of them are based on the available documents too and the author’s experience as a developer and project manager for four years.

### 3.3 Previous researches

In order to form the research the author needed to find a presented way with details that has been used before by academics, to compare different methods in IS researches. Kitchenham (1996) presents the screening-mode as the suitable process for this kind of research and explain about it more on seventh part of a research series in Kitchenham & Jones (1997) in details with different step that the author of this study has utilized; although, the author has not utilized it all with completely match details in methods part instead has focused on assessment criteria with the same value.

Bohem & Turner (2003) defines and present different assessment criteria for checking if a project is suitable for agile, otherwise the project can be suitable for traditional methodologies. The recently presented study has affected the author to find the most important criteria of a project as essence of agility and use them after changing the criteria to become more suitable for the current research.

Abrahamsson et al. (2002) presents the useful idea of comparing methodologies by focusing on life cycle and processes and take project management factor in to consideration that has helped the author so much,
but the data about methodologies, hence the results, are based on the information extracted in 2002 while it has happened a lot to the agile during these nine years.

3.4 Literature sources

Roberts, et al. (1998), Factors that impact Implementing a system development methodology, IEEE transactions on software engineering, vol 24 no 8  
This study presents useful information on software development methodologies. The aim of the study is to present factors that affect implementing a software development methodology, SDM for short, and define a list of factors and discuss the importance of each.

Beck, K. (1999), Extreme programming explained: Embrace change, Addison-Wesley  
: Extreme Programming (XP) has first being presented in this study. Beck (1999) explains all about XP in this book from processes to roles and all details.

The reader can find more useful, details and up-to-date information in this book.

Cockburn (2002) explains all about Crystal methodology in this book, he is the inventor of this family and presented it after doing an empirical experience with this methodology.

As the name shows it includes a practical guide to FDD. Moreover, practitioners can find some more up-to-date information about how-to-use this methodology, simply by searching the web.

*Kruchten, P. (2000), The Rational Unified Process, Crosstalk9(7)*

It includes everything about RUP. Practitioners can find useful information about using this methodology simultaneously with some other methodologies including XP by searching the web.


The author has extracted the data about DSDM out of this source. DSDM has been declared firstly in this book by Stapleton.


Adaptive Software Development, ASD for short, is declared firstly and fully in this book that has been the author resource about ASD for this study.

*Sharma, Sugumaran & Rajagopalan (2002), A framework for creating hybrid-open source software communities, Information Systems Journal12 (1)*

Under this study *Sharma, Sugumaran & Rajagopalan (2002)* claim that their research focuses on how organizations can develop an environment resembling to OSS to administer their software development efforts to obtain its numerous advantages. They create a framework that guides the development and administration of a hybrid-OSS community within an organization. As it is clearly seen, they help organizations to utilize open source concept by presenting a methodology (*Sharma, Sugumaran & Rajagopalan 2002*).


Present and discuss Agile Modeling. Thorough studying the book the reader will realize AM as a modeling methodology for XP.

The author has used this literature mostly as the resource for PP, since it is clearer.

3.5 Agile Software Development Methodologies (Deeper look and comparison)

3.5.1 Extreme Programming

Process

According to figure 2 XP includes 5 different phases named: Exploration, Planning, Iterations to Release, Production, Maintenance and Death. Each step is declared in the following part according to (Beck 1999) declarations:

![Extreme Programming lifecycle](image)

**Extreme Programming (XP) lifecycle**

**Figure 2 Extreme Programming lifecycle**

Exploration phase: Customers prepare the story notes. Story notes/cards will include all features the customers like the very first version of the software have. There will be a one-to-one relationship between each story card a each
Simultaneously, the software production staff checks out the tools and method they need for the project. The technology will be checked and the architecture options would be examined by means of prototyping. The time for this phase depends heavily on the familiarity with the technology; meaning, the more developers know the technology the less this phase would take.

Planning phase: manage the aligning of the customer stories and consent about the results and contents of the first release is made. The needed time is estimated by summing up the estimated time for developing each story and this time basically should not be over than two months by knowing that a few days should be dedicated for whole this arranging and estimating time.

Iteration to release: starts with some iteration before the first release. The schedule items are cut down to iteration that will take one to four weeks to develop each. This is customer who chooses the stories for iteration. The final tests formed by the customer would be executed at the final stage of iteration. Finishing stage of the last iteration will trigger the starting of the production phase.

Production phase: more test and performance control is needed in order to deliver an acceptable system to the customer. The flexibility degree of this phase would be discussed and the result would let the changes to be made or not. So the unmade changes are documented for the postponed implementation. From now on the XP should keep the production executing and take care of starting a new iteration; thus, the Maintenance phase needs to do the customer support jobs, too. Combining some other people or changing the structure or both might be needed under the maintenance phase.

Whole the system is near to death when customer does not ask for more stories and is satisfied with the system, or when it turns out to be too expensive for more changes.

Finally, the documentation of the system begins when the system has reached to a stable point where no more changes are required; not to the architecture neither to the code.
Practices

XP attempts to make the dream of successful software development true by solving the problem of ambiguous and frequently-changed customer needs in teams of three to twenty project members. Main characteristics of XP can be listed as:

- Brief documentation
- Pair programming
- Shortened iterations with small releases
- Very quick feedbacks
- Customer attendance
- Collective ownership of the code

Beck (1999) has listed practices of XP as following:

**Planning game**
Frequent communications between developers and the customer leads to estimate cost for the implementation of customer stories by developers. On the other hand, customers will specify the scope and timing of releases.

**Simple Design**
The clearest and easiest possible implementable solution has to be done at the moment. Any sign of unnecessary complicatedness is thrown away.

**Small/short releases**
A simple and clean system is produced in the minimum possible of time in at least once in every two to three months. The developers will start to release new versions with the frequent of monthly to daily.

**Testing**
Test-driven software development is one of the focal points in XP. Before the code, the unit tests are implemented and will run continuously. Functional tests are written by customers.
Metaphors
Among developers and customers, metaphor/set of metaphors is defined by the system. This story describes how the system works which will be a guide for all developers.

Collective ownership
All developers have full access to all part of the code at any time.

Refactoring
Deleting repetitions and duplications, improving communication, shortening and adding flexibility to the system.

Pair programming
Two people write the code at one computer (Beck 1999).

40-hour week
Not working more than 40-hour a week. No two overtime weeks in a row are allowed. If this happens, it is treated as a problem to be solved (Beck 1999).

Continuous integration
As soon as a new piece of code is ready it will be amalgamated into the code. As a result, the system is built many times a day. Changes in the code have to be done after the tests so that they can be accepted.

On-site customer
Customer has to be ready to contact with in a full time manner.

Just rules
Although the team must obey their own rules, the rules can change whenever it is necessary. The changes and the impact of changes has to be discussed and agreed.

Open workspace
It is preferred to have a large room with some compartments in while programmers are located in the center making pair programming easier.

Coding standards
Developers should follow the existing coding rules. It is so important to keep frequent communication between developers through coding.
Scope of use

Beck (1999) reports that XP methodology cannot be used for all type of projects and teams; However, all limitations are not recognized yet. Thus, there is the need of more experimental research on XP from different perspectives.

XP is designed to be used in small and medium scaled teams-not large ones. Beck (1999) believes that number of team members should be limited to three to maximum of twenty people. As pointed before, the physical environment has a focal role in XP since there is a need of frequent communication between developers so the environment structure should not be a hinder for communication. As Beck (1999) says “a scattering of programmers on two floors or even on one floor is intolerable for XP”. On the other hand, geographical distribution is not stated as a limitation in XP documentation. What can be found around this subject in Beck’s book is that “Two teams working on related projects with limited interaction”.

Another characteristic that affect XP is the business culture. There should not be any resistance against XP practices among project members or management or even customers. Since communication and participation of all these people is a focal point in XP to make XP work, any resistance against XP by one of these can lead XP to failure. Furthermore, it is not just about people but about technology, too. The technology must be so flexible against change in order to lead XP to success.

3.5.2 Scrum

Process

As you can see in figure 3 scrum includes three phases namely: pre-game, mid-game or development and post-game.
The Scrum phases are declared below according to Schwaber (1995) and (Schwaber & Beedle 2002)

The pre-game phase consists of two sub-phases as follows:
   - Planning and Architecture
   - High level design

**Planning**: presents the declaration of the system that is going to be developed. A Product Backlog list is defined as a list including all the requirements that are currently known. Customer, sales and marketing division, customer report or software developers can fill the Backlog list with their requirements. The requirements are prioritized and the cost of implementing is assessed for each. New and more detailed are being added to the Backlog list frequently, so the priority orders and assessments are updated to be more exact as well. Furthermore, planning includes declaration of the project team structure, tools and other resources, risk assessment and controlling concerns, training requirements and verification management approval. For the iteration, the updated product backlog is checked up by the Scrum Team/s in order to know their duties for the next iteration.

**Architecture phase**: Under this phase, the architecture and generally the high level design of the system is arranged based on the current items of the Product Backlog. In case of an improvement to an existing system, the changes that may occur to implement the Backlog items are declared as the
probable problems. In order to review the suggestions for implementations a design review meeting is held. Besides, decisions are made along with preparing plans for the contents of releases in these review meetings.

**The mid-game or development phase:** this phase is the agile section of the Scrum method. This phase is assumes as the black-box where unpredictable might happen. There are some variables in scrum including environmental and technical (e.g. time frame, quality, requirements, implementation tools and technologies and development approaches) which might change during the process, are recognized and checked through different Scrum practices during the Sprints of the development phase. Instead of taking all this considerations in to account at the beginning of the software development project, Scrum attempts to administer them frequently so as to lead in to an adaptable system that can adapt to changes easily.

The system is developed in Sprints under the mid-game phase. Sprints are iterative cycles where the functionality is developed or improved to produce new increments. Sprint can be assumed as the traditional part of the Scrum process since includes the traditional phases of software development: requirements, analysis, design, evolution and delivery phases. Under the development of the Sprint the architecture and the design of the system develop. Each Sprint is scheduled to take one week to one month. There maybe, for example, three to eight Sprints during development of a system before distribution phase of a system (Schwaber & Beedle 2002). Besides, there can exist more than one team involved in implementation (Schwaber & Beedle 2002).

**The post-game phase:** includes termination of the releasing. When it is agreed that the environmental variables (e.g. requirements) are done this phase is initiated. In this level, no more items and variables can be found uncompleted nor can any new one be invented. Thus, since it is the sign of starting the release the preparation for release is done in this phase consisting tasks of integration, system testing and documentation.

**Practices**

The definition of Scrum practices is given below according to Schwaber & Beedle (2002).
**Product Backlog**
Product Backlog describes all the requirements; hence, describes whatever feature that is needed for the final product. It, as a result, includes the work to be done in the project. It has a list of business and technical requirements for the system which is frequently updated.
Each Backlog item includes, for example, features, functions, bug fixes, defects, requested enhancements and technology upgrades. In addition, Backlog list contains unsolved concerns. Different roles might feed the Backlog including customer, project team, marketing and sales, management and customer support (Schwaber &Beedle 2002).

This practice consists of the tasks for creating the Product Backlog list, and reviewing it regularly through the process by adding, deleting, identifying, updating, and prioritizing Product Backlog items. The product owner is in charge of controlling the Product Backlog.

**Sprint**
Sprint is the procedure of adjusting to the changing environmental variables consisting time, knowledge, requirements and etc. The team arranges itself to generate a new executable product increment in a Sprint that takes at least a month. The working tools of the team are Sprint Planning Meeting, Sprint Backlog and Daily scrum meetings.

**Sprint planning meeting**
This meeting is a two-phase meeting managed by the Scrum Master. In order to decide about the goals and functionality of the next Sprint, almost all stakeholders of the project take part in the first phase of the meeting.
Through the second phase of the meeting Scrum Master and Scrum Team are focusing on how the product increment is implemented.

**Sprint Backlog**
Under a Sprint a list of Product Backlogs are implemented. At starting of each Sprint this list which is called Sprint Backlog is used. Scrum Team and Scrum Master and Product Owner select the items in the Sprint Planning meeting based on prioritized items and goals set for the Sprint. In contrary with the Product Backlog, the Sprint Backlog remains unchanged before
Sprint accomplishment (i.e. a month). After completing all the items in the Sprint Backlog, a new iteration of the system is triggered.

**Daily Scrum meeting**
The improvement of the team is kept track of constantly under these regular meetings. During the scrum meetings (also known as planning meetings) that takes about fifteen minutes these topics will be discussed:

- What has been done since the last meeting and what is to be done after this meeting?
- Problems and other environmental variables are discussed.
- Any obstacles in the systems development process or practices are searched, recognized and removed

Leading the Scrum meeting is the Scrum master’s responsibility

**Sprint Review meeting**
An informal meeting will be held on the last day of the Sprint in which the Scrum Team and Scrum master demonstrate the results to the management, customers, users and Product owner. Thus, they will evaluate the product increment and come to agreement about the following steps.

**Effort estimation**
Effort estimation is an iterative activity that the Backlog item estimates are focused on a more correct level when more information is available on a certain Product Backlog item. Performing the effort estimation is the responsibility of the product owner and the Scrum Team/s.

**Scope of use**

Scrum is an approach being designed for team members between 5 to 9 according to Schwaber and Beedle (2002). If more project members are added, different teams should be arranged.
3.5.3 Crystal Family of Methodology

Common rules
There are some rules, characteristics and values that are common among all methodologies existing in this methodology series. These common points are as follows:
- The project uses incremental development cycles which is one to three month or four months at most.
- The stress is on communications and like all other agiles, these are people-oriented
- Any development practices or tools are welcome in crystal methodologies and they also let the approbation of other agile methodologies like XP and scrum.
- The methodology embodies objectives for reducing intermediate work products and shaping conventions within methodology for individual projects and developing the through the project improvement. (Abrahamsson, et al. 2002)

According to (Abrahamsson et al. 2002) the first two of the Crystals including Clear and Orange are experimented in practice and are discussed in this section.

Process
The two members of Crystal Crystal family that have been developed and used are Clear and Crystal Orange (Cockburn 1998, 2002).
As shown in the figure 4 Crystal Clear is presented for small-scaled projects (to maximum six people). Although, with some changes and additions it can even be used for E8 or even D10 systems. The matter in Crystal Clear is the project members should be co-located in order to fulfill communication steps of the methodology (Cockburn 2002).

In contrary with Crystal Clear, Crystal Orange is used for medium-sized projects, with project members of 10 to 40 people working on a project that will not take more than two years. Beside, Cockburn (2002) states that with
some extensions to verification-testing processes, the Crystal Orange can be utilized as E50, too. The difference for Crystal Orange is that in Crystal Orange the project is divided in to several teams with cross-functional groups by means of Holistic Diversity strategy. The balance between thorough deliverables and frequent changes in necessities and design lead in to a controlled amount of deliverables that will cause to reduce cost of maintaining the projects, but still keeping the communication functioning efficiently between the teams (Cockburn 1998).

**ONE INCREMENT**

![Diagram of One Crystal Orange Increment](Abrahamsson et al. 2002)

**Practices**
There are some common practices between all Crystal methods e.g. incremental developments. According to figure 4 each increment in Crystal Orange consists of different activities including staging, monitoring, reviewing, plus parallelism and flux. The listed activities and some other recognized activities according to Cockburn (1998) are described as follows:
Staging
Under Staging arrangements for the next increment is done. This arrangements includes planning to produce a working release in at most each three to four months while a planning of one to three thirty calendar days are the best (Cockburn 2002). Choosing the requirements to be implemented for every increment is the responsibility of the team and they will agree on those requirements they can deliver in the increment.

Revision and Review
There are different iterations in each increment that includes following activities: construction, demonstration and review of the objectives of the increment.

Monitoring
The development is observed concerning what the team has to deliver under the development process regarding their advancements and stability. The improvement is measured by milestones (start, review 1, review 2, test, deliver) and stability stages (wildly fluctuating, fluctuating and stable enough to review). Monitoring is needed in both Crystal Clear and Crystal Orange.

Parallelism and flux
Next step is triggered if the output of monitoring of stability admits the stability for reviewing for the deliverables. In Crystal Orange this means that the multiple teams can get going with the maximum possible of parallelism without failure, but in order to guarantee the success monitoring and architecture teams review their work plans, stability and synchronization.

Holistic Diversity Strategy
In order to divide large functional teams in to cross-functional groups, Crystal Orange has a holistic diversity strategy. The focal point of this strategy is to have varied specialties in one team. Besides, holistic diversity approach let small teams to be casted under the condition of having necessary special know-how, and have an eye on how the team is located and how the communication and documentation and coordination of multiple teams work out.
Methodology-tuning techniques

The methodology-tuning technique can be assumed as one of the principle techniques of Crystal Clear and Orange. It uses project interviews and team workshops for working out a specific Crystal methodology for each single project (Cockburn 2002). One of the basic theories of incremental development is to allow the process to fix and improve. In every increment the project can be trained and use the knowledge it has achieved for improve itself for the next increment.

Using reviews
Cockburn (2002) claims that for each release in Crystal Clear two-user reviewing are suggested while for Crystal Orange, reviews of users are needed three times in each increment.

Reflection Workshops
There is a rule that a team should hold pre- and post-increment reflection workshops in both Crystal Clear and Crystal Orange (with recommendation for also mid-increment reflection workshops)(Cockburn 2002)
One of the characteristics in Crystal Clear and Crystal Orange which worth to discuss is that nor Crystal Clear neither Crystal Orange declare any specific practices, techniques or approach that software developers can use to do their development jobs. Thus, techniques from other methodologies e.g. scrum or XP can replace some of the practices in Crystal like reflection workshops.

Scope of use
The most obvious limitation that can be addressed here is based on Cockburn (2002) the Crystal family of methodologies does not cover life-critical projects shown in figure1. Besides, Cockburn (2002) claims that only co-located teams can find this family of methodologies useful.
Having monitored the Crystals individually, Cockburn (2002) has presented Crystal Clear with a limited communication structure that makes Crystal Clear applicable only for one team that are working at the same office. In addition, what make Crystal Clear inappropriate for life critical systems is its insufficient validation components. The team has to be co-located and is able to work on a system which its value is not higher than the money in the level of criticality in Crystal Orange. Crystal Orange has no sub-team structure which makes it suitable for projects involving up to 40 people. It also lacks activities for design and code verification that makes it unsuitable for life-critical systems.

3.5.4 Feature Driven Development

Process
As you can see in figure5 FDD has five sequential processes that handle designing and building the system. The agile scent is smelled from the iterative section in FDD processes comprising Building and Designing with changes that varies from rapid to late ones in customer and business requirements. The iteration of each feature is limited to one to three weeks (Abrahamsson et al. 2002).

Practices
Palmer & Felsing (2002) claim that the amalgamation of existing practices in order to form the five processes of FDD makes these processes so individual for each project. Besides, they suggest to use all defined practices since no single practice dominates the entire process.
The practices are as follows:
Domain Object Modeling: Defines a framework for features to be appended later by exploring and declaring the problem domain.
Developing the feature: Developing and tracking the improvement process by means of a list of trivial functionally broken down and client-valued functions.

Individual Class (Code) Ownership: Consistency, performance and conceptual integrity of each class has to be guaranteed by a specified person.

Feature Teams: small-scaled and actively arranged teams

Inspection: points to application of best-known defect-detection strategies

Regular builds: In order to guarantee the existence of an always running, usable and stable system from sketch to recently added features.

Configuration Management: Makes tracing the system possible based on the latest version of each finished source code.

Progress reporting: According to the completed work to all necessary organizational levels the progress is reported.

In order to get the most benefits of FDD, the team has to use all the defined practices. Although, Abrahamsson et al. (2002) claims that the FDD is flexible enough to be adoptable to the team experience level.
Scope of use
Not having varied and enough experiences and success stories on FDD can be addressed as one of the basic shortcomings on FDD, since it makes it more difficult for practitioners when it comes to detail level. On the other hand, existing practical experiences turns FDD in to a considerable candidate for large-scaled industrial projects. Besides, FDD seems to be a good choice for any other projects after amalgamating it with XP or Scrum.

3.5.5 Rational Unified Process

Process
The lifetime in a RUP project consists of four different lifecycles including Inception, Elaboration, Construction and Transition as you can clearly see it in the figure7.
Figure 7 RUP phases
In RUP a project is defined in terms of four phases in which exist iterations. Iteration can take between two weeks to six months. At the finishing phase of iteration a demonstrable part of the software must be ready.

The RUP phases are declared as follows according to (Kruchten 2000)

**Inception:** This phase mainly consists of pre planning and pre studying of the project.

**Elaboration:** The architecture of the project is being illustrated in this phase. The use cases are done along with their estimated risks. The software architecture model and development plan can be named as the most important results of this phase.

**Construction:** As the name shows the coding is done under this phase.

**Transition:** The system transits to production and goes under quality check.
Below (figure8) you can find the major milestones of RUP and the transmission states.

**Practices**

The foundation of RUP can be summarized in to six different practices that play a very primitive nucleus role in RUP. You can find a list of practices below:
Develop software iteratively, Manage requirements, Use component-based architecture, Visually model software, Verify software quality, and Control changes to software.

Scope of use
Abrahamsson et al. (2002) claims that “one of the major drawbacks of RUP is that it fails to provide any clear implementation guidelines in the vein of, for example, Crystal, which lists the required documentation and roles with respect to software criticality and project size”. Besides, it seems that tailoring can be a challenging part for the RUP team since there are not still enough documents and guidelines about it.

3.5.6 Dynamic System Development Method

Process
DSDM consists of five phases: feasibility study, business study, functional model iteration, design and build iteration, and implementation as it is shown in figure 9.

![Figure 9 DSDM process diagram (Stapleton 1997, p.3)](image-url)
The architecture of DSDM is declared briefly below according to DSDM Atern e-book:

The DSDM development process consists of 7 phases which include Project studies phase, iterative cycles and maintenance phase. As you can see in the figure above the project flow moves between phases as indicated with arrows. Arrows with different colors have different meanings, the blue ones show a forward flow, the greens show a flow that might be taken if circumstances are normal and reds represent the flow that’s taken if the project is not under normal circumstances which happens when the project has not met the requirements.

Pre-Project

The pre-project phase consists of simply the pre study steps of the project.

Figure 10 DSDM feasibility study
Feasibility Study
Can the project be done within the constraints of time and resources?

Business Study
Studying the business aspects of the project

Functional Model

![Figure 11 DSDM functional model](image)

Functional prototype of the system is made at this phase, functional prototype is declared as:
“Functional prototypes of the system are made and reviewed. A functional prototype is a prototype of the functions the system should perform and how it should perform them.” (DSDM Atern e-book u.â.)

Design And Build

![Figure 12 DSDM design and build](image)
The result of this phase has to be the design of the product, it has to be designed in a way that it will be implemented in iterations.

**Implement/Deploy/Maintain**

![Figure 13 DSDM implement-deploy-maintain](image)

Finally, under the last phase, the documents are finalized and a review document would be done as well. On the other hand, users confirm the product functionality after they learn how to use the system.

**Post-Project - Maintenance**

**Practices**

The practices that form the cornerstone of DSDM are listed as below:

- Eager communication with the customer (like almost other technologies of agile)
- Decision making should not be a monopoly and only in the hands of management
- Delivery should be on a regular basis
- Ensuring that the product matches the business needs
Iterative and incremental process is needed to make sure the customer is fulfilled
Changes during the development process should be revocable
Principle requirements should be kept unchanged during the process at a high level
Both developer test and user test are simultaneous with developing
Frequent communication between the organization and customer and also between different departments of the organization is a must

Scope of use
The team size for DSDM can be between two and six people, and there is the possibility of having more than one team in one project. The DSDM has so many success stories which can varies from small-scaled to large-scaled projects according to their website. Stapleton (1997) believes that DSDM is more suitable for business systems than engineering or scientific applications.

3.5.7 Adaptive Software Development

Process
A software development project is described in three phases named Speculate, Collaborate and Learn as shown in figure 14 below.
“Speculation” is the arrangement phase, in this phase uncertainty and deviations are supposed to be a recipe for failure. “Collaborate” – as its name might show– focuses on communication and teamwork. The need to have a system that learns from its mistakes and improve itself is expressed as “Learn”.

The characteristics of ASD are as follows according to (Highsmith 2000)

- Mission-Driven
- Component-Based
- Iterative
- Time-Boxed: having deadlines on a regular basis
- Change-Tolerant
- Risk-Driven

**Practices**

ASD presents few practices named “iterative development, feature-based (component-based) planning and customer focus group reviews” as Highsmith (2002) claims.
Scope of use
The main difference of ASD and any other agile methodology is that in ASD the project members are not forced to be co-located, since in ASD the dominant idea is providing teams with a framework to build a complex system and in order to build a complex system a wide knowledge is required; thus, experts from different places and different teams are required.

3.5.8 Open Source Software Development

Process
Any OSS project can consist of phases as follows (Sharma, Sugumaran & Rajagopalan 2002)
- Problem Discovery
- Finding Volunteers
- Solution Identification
- Code development and testing
- Code change review
- Code commit and documentation
- Release management

Practices
OSS development is characterized as follows according to (Mockus et al. 2000)
- Large numbers of volunteers build the system
- People accept the task they choose to (no one assign the tasks)
- Any formal high level or detailed sketch exist

Scope of use
Most of documented successful results are in large software tools and Internet based products. The process can be slowed down since project members can be so different in skills and points of view. On the other hand, OSS can be good choice to discuss for multi cultural companies that are geographically dispersed.
3.5.9 **Agiles: Comparison approach**

Below four different areas are chosen to compare agile methodologies against. The four areas are: size of the team, scope, team-structure and empirical study. The following section is dedicated to reviewing each area.

**Size of the team**
Agile methodologies revolve around the human-factor. Besides, communications between people including project members themselves and project members with customers play such a substantial role in all agile development methodologies; thus, affect the methodology undoubtedly. A methodology that works best for 10 people might have completely inverse affect with 100 people. Size of the team is considered number of project members in this study. Besides, it will be checked if the methodology supports more than one team simultaneously.

**Scope**
Each methodology might be able to show the best of itself in one or many field according to the existing documentation of the methodology or published empirical study reports. Besides, some methodologies might not work out in some type of projects while work for any other ones. Although, no limitation can be pointed for some methodologies that can have different reasons e.g. lack of empirical reports or the generality of the methodology.

**Team Structure**
There can be different structure of the team/s working on a project. Some projects can run by single teams including limited number of people co-located in a one-floor firm or by a team with a so hierarchical structure located in different branches a company in different corners of the world. Clearly, such varied type of teams cannot go through the success rout with the same map. The team structure mostly implies the hierarchical form of the team, if it is a
single team or can have sub teams. On the other side, the geographical distribution is considered.

**Empirical support**
In the way to success, it is recommended to choose the ways that are more likely to guide you to the goal. Such recommendations come from experiences.
A success story makes the use of methodology clearer and provides teams with more details. Furthermore, empirical support for each methodology guarantees the success of the methodology to some degree especially for similar cases.

Table 1 illustrates the comparison approach of the agile methodologies concerning the classification title above:

<table>
<thead>
<tr>
<th></th>
<th>Scope</th>
<th>Size(#of people)</th>
<th>Team structure</th>
<th>Empirical Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>XP</td>
<td>In any project with non-resistance culture against XP</td>
<td>Between three and maximum twenty project members for more than this quantity multiple team should be performed</td>
<td>Geographical distribution is not out of scope but Communication and coordination between the project members should be enable in all of the time</td>
<td>+</td>
</tr>
<tr>
<td>Scrum</td>
<td>Can be adopted to manage whatever engineering practices are used in the organisation</td>
<td>Between five to nine project members for more than this quantity multiple team should be performed</td>
<td>Not suitable for complex team structures</td>
<td>+</td>
</tr>
<tr>
<td>Crystal</td>
<td>Does not cover life critical project (lack of system validation elements).</td>
<td>Up to 40 people and they must be a single team. It lacks sub-team structure</td>
<td>Only co-located teams can be addressed by these methodologies.</td>
<td>-</td>
</tr>
<tr>
<td>FDD</td>
<td>Embodies iterative Development. Best practices are effective in industry. Success stories are business-oriented</td>
<td>Is applicable for large scale projects mostly</td>
<td>No limitation is declared</td>
<td>-</td>
</tr>
<tr>
<td>Methodology</td>
<td>Key Features</td>
<td>Limitations</td>
<td>Suitability</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>RUP</td>
<td>Is applicable for Object-Oriented systems- Can be applied for large-scale projects and is applicable for small projects if some parts are omitted (is supposed to be a good complex with XP)</td>
<td>No limitation is declared</td>
<td>Suitable for critical systems</td>
<td></td>
</tr>
<tr>
<td>ASD</td>
<td>Focuses mainly on the problems in developing complex and large systems</td>
<td>Possible to not have co-located personnel so makes distributed development possible</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSDM</td>
<td>Is more easily applied to business systems than to engineering or scientific approaches</td>
<td>Teams of seven +/- two people, Solution development team(s) should contain skilled people</td>
<td>There can be many teams. Most of them are in large projects</td>
<td></td>
</tr>
<tr>
<td>OSS</td>
<td>Suitable for geographically and culturally dispersed organization, most success stories are about large software tools and internet products</td>
<td>No limitation</td>
<td>No force to have co-located people</td>
<td></td>
</tr>
<tr>
<td>AM</td>
<td>Suitable for both simple and complex projects. you are much more likely to succeed if you do adopt all of AM.</td>
<td>Same as XP</td>
<td>Although communication is as important as XP, some project members/teams can be not co-located</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Comparing Agile methodologies
Another important approach in comparing approaches that is important for project managers while selecting a methodology for developing a software project is to know which phases of the lifecycle of a software project are affected by the methodology. In other words, it is highly needed to know if a single methodology covers all needed practices through a software project. As to serve this objective, the diagram below (figure15) is given as to provide managers the tool to be able to analyze different agile methodologies easier.

Each method is separated in to three bars. The uppermost bar illustrate whether a methodology provide the needed tools for managing the project. The bar in the middle declares if the process of developing the software is described. The length of the bar indicates the phases of the software development that are covered by the concerning methodology. Lastly, the bottommost bar describes whether a method gives more guidance to the process or it is shortened in the main ideology.
Project management

Regarding covering the project management practices agile methodologies varies in such a wide domain. Currently, AM and PP do not point to any managerial tools or points. Although, AM has improved in recent years by presenting Agile Model Driven Development (am website) it still does not cover any management practices. XP has recently added some guidelines for management but still has a long way to go. ASD, Adaptive Management Model, thrives to make some changes in the development culture. FDD presents features in order to plan the project and keep the project on its own way and prevent any chaos. DSDM, and its latest version, is designed to make any organization equipped against the changes of the Market and requirements. On the other hand, Crystal family provides a manager with different versions to be used for different degree of criticality.

Phases and the Process

All agile development methodologies are not standalone ones as vividly shown in figure 15 the useful information for a project manager is to know which methodology is suitable for which phase of the development process.

ASD does not have adequate information for the project in the phases of acceptance test, project inception and system in use. Crystal family covers phases from design to integration test. DSDM and ISD cover all the phases of the development. DSDM is an independent method since it tries to have complete support over all life-cycle phases, but mostly at a management level.

XP, FDD, Scrum and PP covers requirement specification, design, testing up until the system test and implementation (but not Scrum).

In ASD, Crystal, DSDM, XP, FDD and Scrum the development process has been clarified thoroughly.

Practices and Guidance

A methodology needs to have some practical reviews, tools and clarification in details not just in high level to be useful. Some abstract guidance and principles are not of much help when it comes to using the methodology on a project in practice. A level of detail is needed in a way not to be so abstract
and high level and not to be so detailed that block the creativity of the project members.

Scrum defines the practices and the requirements but not for all phases. As it is seen, ASD focuses on concepts and cultures. AM, XP and PP have been directly derived from practical settings.

### 3.6 Summary of theoretical findings

As stated before the method of evaluating agile methodologies, in order to provide a project manager of a software development project with a guideline to find the most appropriate methodology for the project she/he has in hand, that has been chosen in this study is screening mode. In this part, the screen mode was defined clearly to help the reader to follow the deducting path. Different steps of screen mode were declared and the author thrived to fulfill each item of screening mode within next sections. Each methodology was defined as identifying the methods and they were declared base on three different areas, which served compiling information about the method, called: Process, Practices and Scope of Use. Within Process part whole the process of the methodology was discussed mainly with pictorial illustration. Practices includes all the needed practices within the methodology and Scope of use focused on different fields that the methodology has been used while discussed its limitations according to the existing documents and empirical experiences.

Under comparison part, different assessment criteria were presented and described and were presented in table and diagram in order to make analyzing easier. In addition, the reported results in the form of table and diagram were analyzed later.

According to the comparison part, the project manager can find the most probable methodology for a software development project. Although, he / she can use the information above as a reference of comparison between different agile methodologies and look for any other criteria that is so important for his/her company among the declaration part in order to make less error-prone and less risky decision.

Finally, the comparison part provide primitive answers to the research question. Actually, the presented assessment criteria are the characteristics
that must be considered about each methodology, and the project manager should consider each characteristic presented in table 1 for the software development project he/she has in hand.

3.7 **Arguments for an empirical study**

The result of the theoretical study may partly answer the research questions, but there may be questions that have not been answered or that need further illumination. Identify what an empirical study may contribute to give answers to the research questions and use that as a basis to argue for an empirical study. The theoretical part presents the principals and provides the project manager the declaration and the comparison of methods as to shed light on the path she/he is taking; nevertheless, having empirical support is more helpful and makes all vague points about the process clear. To serve the aim of having empirical support for this study, the Lohorung project is selected. (For more information about Lohorung project and the sample company- Innovationlab- refer to Target group section)

4 **EMPIRICAL SURVEY**

4.1 **Purpose**

This empirical study is used as the way evaluation is organized. Empirical survey is presented as a strategy to be a very helpful and even necessary part of any study that suggests any engineering or industrial practices. The practical essence of such areas of studies calls for any tangible result that can be a usable material for practitioners. Thus, to provide the addressed readers of this study with a practical guidance this empirical study is attached.

The selection process of the desired methodology under this empirical study is done on the Lohorung project and it is expected to find the most appropriate software methodology for this kind of project.
The empirical survey is added to this study in order to monitor the difficulties the project manager might have to consider the assessment criteria. Moreover, the empirical survey might produce additional knowledge that is not covered in the theoretical section.

### 4.2 Sampling

The criteria that are needed to find the project for the empirical study are:

- Software development project.
- Not started state: so it means that the IT manager is looking for an appropriate methodology to start it.
- Feasible for agile

Lohorung project has the both criteria so is chosen as the qualified case. One might challenge Lohorung being a software development project but a linguistic project. It must be said that, since during this project the aim is to help linguistic people to survive the language by means of IT the project is known as an IT project.

Besides, according to the Bohem and turner radar chart (Boehm & Turner 2003) The Lohorung project is feasible for agile. The process of feasibility for agile is out of the scope of this study, for more information and how to use the mentioned chart refer to Bohem and radar chart (Boehm & Turner 2003).

The information about the project and the company is extracted out of an interview with the manager of Innovationlab (the sample company).

### 4.3 Interview with the manager

1. How many people are estimated to participate in the project?
2. What is the percentage of each type of project members if we divide them in to three groups of beginners, intermediates and experts according to their skills?
3. What are your criteria to choose the staff for this project?
4. Will the staff work in Innovationlab? (Will Innovationlab be their work-place)
5. Sub-team structures are not seen in other projects of Innovationlab, Do you consider plain structures more successful?

**Interview transcript**

1. It is not decided exactly since it depends on completion of requirement collection process and the needed time but the amount of project members will not be more than ten.

2. Beginners: 20  
   Intermediates: 20  
   Experts: 60

3. independent  
   ambitious  
   technically  
   competent  
   reliable  
   careful  
   Be open to new technology  
   helpful

4. Yes for the most part, but it is needed to use some experts as consultants for this kind of project.

5. Yes, I consider plain structures to be more successful for a company like Innovationlab, level of employees and project models.

**4.4 Empirical research results**

According to the theory section, as to find out the suitable methodology among all other agile methodologies, the project should be assessed under the presented assessment criteria in the theory section declared in table 1. As stated before, these criteria are the factors that a project manager should consider while choosing a methodology for the project.
characteristics of the methodologies are considered in theoretical part and a comparison result is given in form of a table and a diagram)

Size of people: The estimated number of people is not more than ten people.

Besides, the scope of the project can be defined as a research project; thus, the criticality of the system is not so high. On the other side, the customer still does not have a pre-defined sketch for the project which make the project a communicative one because the project members has to be in frequent contact with the customer to gain more information and make sure the customer is satisfied with the result which lead the project to have an iterative process.

The project members are not going to be dispersed geographically. In addition they will be co-located in Innovationlab which is a large room with some semi-cubical for each person. The existence of sub-teams is not decided, yet. However, there must be some consultants to help the team members but they are not supposed as project members; thus, do not have such main role in selection process.

Moreover, the culture of the team will be so open to new technology and new experiences. On the other hand, the project members should be communicative as the adjective „helpful“ states. Although, it seems that being communicative is not in the center of the manager’s attention and values while selecting the project members for the team which is something that should be assumed as a risk in the project that needs a planned program to work on it for all the project members in order to remove this risk factor.

When it comes to exploratory project, the project manager can be unsure about the factors stated above. The more unsure the answers are, the more risky the project will be.
ANALYSIS AND RESULT

5.1 Analysis

The project members are not decided to be more than ten people. Thus, the projects that are designed for large-scaled projects are omitted from the selection process including FDD, RUP and OSS according to the declaration provided in theoretical part for each methodology. On the other hand, DSDM can be considered as a “border” methodology; although, it is documented to have 5 to 9 project members, all the empirical are for large-scaled projects. Consequently, DSDM is supposed to be 50% useful for this kind of project.

Furthermore, the scope of the project has to be considered as the second mentioned criteria earlier in table 1 in theoretical section. As the criteria for employing/using personnel in this project shows, the team will be open to new technology and will be communicative, thereupon, makes XP a good choice for the project. Obviously, the project is not industrial, but a research project, thus, DSDM and RUP do not fulfill the needs. In addition, the project is not critical, either, according to the description about the project before.

As stated above, project members are not geographically dispersed and the work place provides project members with a perfect environment for communication.

Finally, the first phase of the process is done according to the comparison part of the study and it is time to start the second phase of the study and considering phases and lifecycles. Thus, phase two starts with the remaining options to match to the project domain.

The remaining are XP, Crystal, AM and Scrum. The key to solve the problem lies in process phases. XP covers the bottom two bars named process and activities or guidance. Crystal covers the two upper bars: project management and process. AM covers the lower bottom bar and for just 2 phases of the project And Scrum covers the upper bar for all the phases, and the two bottom bar for two phases.

Finally, XP is a good candidate but it has to be blended with one of other methodologies, since it does not cover all aspects of a project. On the other
side, Crystal shows such high potential for being the next choice, because it covers the two upper bottoms and can be used simultaneously with XP.

Scrum is a good choice, too. The good point for Scrum is that it has empirical support which makes using it easier while Crystal does not support empirical support but is designed for exactly the same team structure as Innovationlab team.

The author individually believes that XP along with Crystal are a good choice for this kind of project. Lack of empirical study in Crystal can be categorized as a risk, considering the Lohorung project as a research-low risk project, the project can tolerate the risk and on the other hand, considering the culture of the Innovationlab and the team that are defined as open to new technology and techniques the team can find its way.

5.2 Result summary

The research questions are:

- What are the different characteristics of different agile methodologies, which should take in to consideration by the project manager during the choosing software development methodology process?
- Which factors of the software development project should a project manager cogitate about while choosing a software development methodology?

After monitoring and reviewing the existing agile methodologies, a thorough comparison of all methodologies is given in the theoretical part. Simultaneously, a two-phase comparison process is suggested to the software development manager to use while choosing a methodology.

The first phase, is to monitor the project in four aspects of size of the team (number of people involved in the project), the project scope (by considering all the possible limitations), team structure (e.g. sub team structures and etc.) and empirical support (indicating if there is any empirical for the project) that represents the factors the project manager should consider about the given project.
On the other hand, the factors represented above are presented to be considered for each declared methodology, too.

After matching the project with some methodologies and delete some other options from the option list, It is time to consider different phases that the methodology covers. Besides, extra help for running the project and life cycle of the project is taking in to consideration.

By means of the comparison material that expressed above, the evaluation process of methods executed for the sample project, Lohorung, and the result was choosing XP along with Crystal as appropriate agile methodologies for this project. The extra knowledge being produced in empirical part can be pointed as existence of mixture of methodologies can be the result.

6 DISCUSSION

6.1 Conclusions

All the existing agile methodologies are reviewed based on existing documents and literature. Having reviewed all the existing methodologies a set of assessment criteria is given as factors that a project manager should consider in a software development project while choosing the methodology. On the other side, all the methodologies are examined under the same criteria and the result is given in form of a table (table 1) just to help the project manager to match the project with one or more methodology. The next phase consists of considering the second part of characteristics of each methodology that has passed the first phase. Under the second phase other important characteristics of the methodologies including the covered life-cycles are suggested to take in to consideratio by the manager.

Finally, an empirical study is done to consider the usage of the suggested process in practice to find out the probable difficulties a project manager might face considering the presented factors of the project. Moreover, it is monitored if extra knowledge produces during empirical section.
6.2 Implications for Informatics

Finding the most appropriate software methodology is seen as one of the hottest topic among software practitioners. The software methodology is the very first step of any development project that will affect all project members and customers. The affect on customer can obviously be seen in the frequency of relationships with the customer. Although, the knowledge of the customer of the system can highly affect the decision of selecting the methodology, since the surer and more knowledgeable the customer is about the system the less Dynamic the project is. There is no need to say that Dynamism is one of the principles in choosing the methodology.

6.3 Method evaluation

The evaluation method, as stated earlier, was screening that is known as a standard evaluation approach to evaluate methods or features in software industry. Although, the features are chosen out of existing description, documentation, reviews and experiences, the lack of documentation for agile methodologies especially earlier ones is tangible. Thus, it can make the evaluation results more fragile. It calls for much more empirical reports of using agile methodologies and more detailed and precise limitations.

6.4 Result evaluation

According to the Lincoln & Guba (1985) these four criteria should be considered in evaluation, as stated before:

Truth value: which include how one can establish confidence in the truth (Lincoln & Guba 1985). The author has established her findings heavily based on the initiative reviewing that presents the existing information about each agile methodology using the original documents for each methodology. On the other hand, the information about the samples, both the sample company and the sample project are extracted from the interview from the company manager and some unfinished documents about the project.

Applicability: It does not contain only applicability in other contexts but also
contains applicability in other subjects (Lincoln & Guba 1985). The author’s findings can be applied in any reviewing of agile methodologies contexts and subjects, since the author presents a brief review of all existing agiles. Besides, the comparison part cab applied in any context related to comparing agiles or related to any of the presented assessment criteria.

Consistency: Lincoln & Guba (1985) believe that consistency is: “how can one determine if the findings of an inquiry would be repeated if the inquiry were replicated with the same subject in the same context” (Lincoln & Guba 1985). Undoubtedly, the results of this study can be repeated for the same or similar studies about agiles because the assessment criteria are chosen from the documents and definition of each agile methodology that would not change even if a new agile methodology is added. Clearly, more assessment criteria might be added in similar context, actually the results can be a base for further studies on agile methodologies comparisons.

Neutrality: Lincoln & Guba (1985) believe that Neutrality is: “if the finding is determined by the subject not by biases, motivations or interests of the researcher” (Lincoln & Guba 1985). The author has attempted to remain a neutral viewer and analyzer to the subject through whole the process. Since the author is not in benefit of the results, achieving the aim of neutrality has not been a difficult aim to her.

Finally, the research questions are answered via the process above. Besides, a sample of utilizing the created knowledge is given to organize and evaluate the given result when it comes to reality.

6.5 **Possibilities to generalize**

The comparison of the methodologies is given so generally and thoroughly. Every kind of project can use the comparison results and method declaration in order to select the most appropriate methodology for the project.

Clearly, Nepal project can be replaced by any project that is feasible for agile methodologies.
6.6 **Ideas for continued research**

This study has completely focused on finding a suitable methodology for a new project, while another important challenge that software project managers face is to adopt a methodology to an already started project. In order to find some facts in this area for making some comparisons in form of tables or etc. researcher has to have an eye on adopting information of each methodology. Some of methodologies provide information about adopting in their documentation.

Method tailoring can be addressed as the best feature that any methodology can offer to its users. Obviously, each project has its own different requirements and characteristics that make it unique. So using different methodology simultaneously can guarantee the success of the software project in a higher degree. As a result, considering method tailoring information about each methodology e.g. if there is any method tailoring experiences on the methodology can be so helpful for project managers and make the selection result more precise.
References


Beck, K. (1999), Extreme programming explained: Embrace change, Addison-Wesley


Cockburn, A. (2000), Writing Effective Use Cases. The Crystal Collection for software Professionals, Addison-Wesley Professional


Highsmith, J. (2002), Agile software development ecosystems, Boston, MA., Pearson Education


Ruparelia, N. B. (2010), Software Development Life Cycle Models, Software Engineering Notes vol 35 no 3


University of Borås is a modern university in the city center. We give courses in business administration and informatics, library and information science, fashion and textiles, behavioral sciences and teacher education, engineering and health sciences.

In the School of Business and IT (HIT), we have focused on the students' future needs. Therefore we have created programs in which employability is a key word. Subject integration and contextualization are other important concepts. The department has a closeness, both between students and teachers as well as between industry and education.

Our courses in business administration give students the opportunity to learn more about different businesses and governments and how governance and organization of these activities take place. They may also learn about society development and organizations' adaptation to the outside world. They have the opportunity to improve their ability to analyze, develop and control activities, whether they want to engage in auditing, management or marketing.

Among our IT courses, there's always something for those who want to design the future of IT-based communications, analyze the needs and demands on organizations' information to design their content structures, integrating IT and business development, developing their ability to analyze and design business processes or focus on programming and development of good use of IT in enterprises and organizations.

The research in the school is well recognized and oriented towards professionalism as well as design and development. The overall research profile is Business-IT-Services which combine knowledge and skills in informatics as well as in business administration. The research is profession-oriented, which is reflected in the research, in many cases conducted on action research-based grounds, with businesses and government organizations at local, national and international arenas. The research design and professional orientation is manifested also in InnovationLab, which is the department's and university's unit for research-supporting system development.