Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 20 Issue 2 Version 1.0 Year 2020
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Comparison between Agile and Traditional Software Development Methodologies

By A.K.M Zahidul Islam & Dr. Alex Ferworn
Ryerson University

Abstract- Agile and Traditional software development methodologies, both are being used in different projects of software development industry. Agile software development technology is an incremental software development process. On the other hand, Traditional software development methodologies or plan-driven software can be explained as a more formal approach to software development. These methodologies come with a fully completed set of systems requirements followed by an architectural and high leveldesign development and inspiration.

This research focuses on the software development life cycle, role and responsibilities of agile and traditional software development methodologies and their technical practices. It performs a comparison between both the software development methodologies. Here a questionnaire is used to collect data from the various experts of different IT related organizations of Bangladesh. In the questionnaire, there are three sections to bring out the individual knowledge from different organization, methodology knowledge of the respondents and software development experience of the respondents. The respondents are mainly software engineer, system analyst, software developer etc. A comparison is also performed between this survey result and a survey done by Ambler.

GJCST-C Classification: *K.6.3*

ACOMPARISONBETWEENAGILEANDTRADITIONALSOFTWAREDEVELOPMENTMETHODOLOGIES

A Comparison between Agile and Traditional Software Development Methodologies

A.K.M Zahidul Islam [α] & Dr. Alex Ferworn [σ]

Abstract- Agile and Traditional software development methodologies, both are being used in different projects of software development industry. Agile software development technology is an incremental software development process. On the other hand, Traditional software development methodologies or plan-driven software can be explained as a more formal approach to software development. These methodologies come with a fully completed set of systems requirements followed by an architectural and high level-design development and inspiration.

This research focuses on the software development life cycle, role and responsibilities of agile and traditional software development methodologies and their technical practices. It performs a comparison between both the software development methodologies. Here a questionnaire is used to collect data from the various experts of different IT related organizations of Bangladesh. In the questionnaire, there are three sections to bring out the individual knowledge from different organization, methodology knowledge of the respondents and software development experience of the respondents. The respondents are mainly software engineer, system analyst, software developer etc. A comparison is also performed between this survey result and a survey done by Ambler.

The analysis demonstrates the effect on software quality and cost from agile methodology and compares it with ambler (2007) survey and tries to find out correlation between the cost and quality of both the surveys. According to the respondents of the survey (Questionnaire) it is clear that what are the facilities and drawbacks of the traditional and agile software development methodologies for different size of the projects of an organization. At the end of the analysis part of this research it shows that for small scale projects more than 90% respondent response for agile methodologies and less than 10% responds for the mix software development technologies which are specific for a organization. For medium scales projects about 50% responds for agile software developments methodologies, more than 40% responds for the traditional software development methodologies and less than 10% responds for the other mix technologies for an organization. For the large scale project less than 10% responds for agile methodologies, more than 80% responds for traditional methodologies and slightly more than 10% responds for the other mix technologies for a specific organization.

The findings of this project research study also confirm the appropriateness of the use of agile methodologies for small scale projects, traditional and agile methodologies for medium scale projects and traditional methodologies for large scale projects of an organization.

Author α: Ryerson University, Toronto, Canada.
e-mail: akmzahidulislam102@gmail.com

I. Introduction

The software development industry is one of the fastest growing industries in the world. By analyzing previous 20 years history of software development it is evident that a lot of brilliant ideas and methods born repeatedly. However, there was no guarantee whether those methods will last long or not though there are a good number of examples to prove this.

The concept of “Agile” is new. When it was introduced there was no agreement or explanation on what precisely it refers to. Despite this doubt agile methods became very popular among the industry within a very limited period. Agile was born after introducing extreme programming also known as XP. There are different methodologies comes under agile such as Dynamic Systems Development Method, FDD, TDD, SCRUM and etc.

“Agile” has the high reputation and interest in the industry but still there is no clear agreement on how “Agile” can be distinguish from more “Plan-driven” methods which are also known as the traditional methods. So it cannot identify any boundaries or limitations (Boehm and Turner).There is no any systematic check on agile methods. However, there are some studies to identify the suitability of agile methods for different software project natures. Due to that there are no current events or guidelines for practitioners to select the best method to bring the maximum benefits to their projects.

“Agile” is becoming more renowned in the software industry. Agile methods are overtaking tradition methods in projects where requirements are changing frequently. In agile software development there is a series of software behaviors which is conventional as well as controversial. As a result, in the near future the software development industry will find ways to carefully use either the traditional or the agile methods or a hybrid of these two methods.

To get highest result and to achieve the goals, a software development team needs to understand and select the most suitable methodologies and techniques for their project. When acquiring the understanding that they can find answers to these questions:

“What natures of project they have in hand the possibility of changes while the project in progress?”

"What is an appropriate balance of effort between documenting the work and getting the product implemented?" (Lindvall et al., 2002)

"When does it pay to spend major effort on planning in advance and avoid change, and when is it more beneficial to plan less rigorously and embrace change?" (Lindvall et al., 2002)

In order to answer properly to above questions and to make the correct decision proper knowledge should be implemented and should be disseminated within the industry. This research aims to develop a set of guidelines to help an organization in their decision making, when selecting the best software development methodology to a given nature of a project or projects, by doing a review on the different traditional and agile methods.

a) *Aims of the Research*

The aims of the research project are:

1. Review a number of different software development methods, both traditional and Agile.
2. "Can agile methods be used in any type of software development project?" find out the answer of this question.
3. Come up with a set of guidelines for a software organization to select the most suitable software development methodology for their software projects.

b) *Objectives of the Research*

The objectives are:

1. Carry out a literature survey on different software development methodologies.
2. Understand the lifecycles, roles and practices of these development methodologies.
3. A comparison for agile and traditional development methodologies to understand the similarities and differences.
4. Carry out a survey in the software industry with practitioners and professional in software engineering.
5. Analyze the gathered data from the survey and summaries them to fulfill the final aim with the help of the knowledge from the literature.

c) *Research Question*

What are the significant factors for a project to consider the most appropriate type of process model, after comparing agile and traditional software development methods?

d) *Structure of this Research*

The first chapter introduces what is the aim and objective of this research and what is the research question of this research. The second chapter introduces the literature review of this research to answer the research question. The third chapter introduces the research design and makes a questioner for the target audience of this research. After a survey from the audience the result of this research is discussed in chapter four. Basically this questioner helps to collect data for this research. Chapter five analyzes the research result and tries to bring out proper methodology for specific software. The final chapter tries to bring out limitation of this study and future aspect of this research.

II. Literature Review

a) *Outline*

The Manifesto for Agile Software Development (MAD) was published in 2001 by a group of seventeen methodologists. This group of experts agreed on a common set of guiding principles and practices around effective software development. The focus was for modeling and documentation of software development projects. The methodologists introduce the guidelines which are: (Fowler and Highsmith, 2001)

- Individuals and interactions over processes and tools

  The main concern in this section is the relationship and communication between the software developers and any other persons involve in the software development process. The dependency on just tools and processes will be minimal.

- Working software over comprehensive documentation

  The main purpose here is to keep the documentation as small as possible and thus concentrating more on building and delivering tested and quality products. Different teams can handle the deliveries differently. Some may deliver hourly or per week while others releases product every two weeks or once a month.

- Customer collaboration over contract negotiation

  The main concern of this section is the relationship between the development team and the client. The relationship has to be very high. However, the importance of having a contract and changing it accordingly is important as well. Agile starts to release functional program modules as soon as the development process starts and thus it effectively minimizes the risk and disappointment of not meeting the actual requirement at the far end of the project.

- Responding to change over following a plan

  The people who are involved in the software development like programmers, clients and any other should be well knowledgeable about the progress and any changes. Any party have the authority to consider possible changes to the product When it is been developed.

The founders of MAD say "while we value the items on the right, we value the items on the left more" (Fowler and Highsmith, 2001), so there are different debates on these values. There are other practitioners including Steven Rakitin (2001) who thinks that items on the left are just an excuse for hackers with no regard for engineering discipline.

Traditional software development methodologies or plan-driven software can be explained as a more formal approach to software development. These methodologies come with a fully completed set of systems requirements followed by an architectural and high level-design development and inspiration. However, during mid 1990's some practitioners found some steps such as full documentation frustrating and unnecessary time wasting (Highsmith, 2002). Due to these heavy aspects, this methodology is known as heavyweight development methods.

Traditional development methodologies all include with the following (Williams & Heckman, 2008):

- Repeatability and predictability
- A defined incremental process
- Extensive documentation
- Up-front system architecture
- Detailed plans, process monitoring
- Controlling and education
- Risk management
- Verification and validation.

The Personal Software Process (PSP), Team Software Process (TSP), and Rational Unified Process (RUP) are the three of the most popular and widely used plan-driven methodologies. Among these plan driven methodologies waterfall model and spiral model are well-known.

According to Davis and Sitaram (1994) waterfall model have the ability to capture the gross state of the project. Using this model therefore a project manager can track the progress through all major phases of development of major intermediate products. On the other hand spiral model captures the iterative nature of software versions and helps the project manager to isolate the key decision points to select a development strategy. They further argue "Neither of these two models, nor any other published model, provides a project manager with a picture of the true state of the project. Project managers who track project status in terms of the major phases have no idea of the status of their projects."

The following table which was published by Abrahamsson et al., (2002) demonstrates the differences of privileged and marginalized methodological information systems development process. These were a collection of views from different authors in the field.

Table 1: Privileged v Marginalized text (source: Abrahamsson, 2002)

Privileged methodological text	Marginalized methodological text
Information systems development is	
A managed controlled process	Random, opportunistic process driven by accident
A linier sequential process	Processes are simultaneous and overlapping and there are gaps in the between
A replicable universal process	Occurs in completely unique and idiographic forms
A rational, determined and goal driven process	Negotiated, compromised and capricious

The marginalized methods have much more things in familiar with the original agile development methods. The privileged method projects use more of a process oriented software development methods. These methods also called plan-driven methods.

McCauley (2001) argues that the underlying philosophy of Traditional methods which is referred to as process-oriented methods in the article, is that the functional requirements of a project is utterly frozen or in other words sealed before move in to the next phases such as the design and development. The article also states that this approach is not feasible for most of the software projects. So the need of a flexible and agile development methods is necessary for developers to make changes or amendments to the specifications while it is been built. Further according to McCauley (2001) there is no software model that suits any nature of software project. It is the project management who should be able to select the best suitable methodology according to the project in hand. There are different other experts in the field who support this argument.

b) *Characteristics of Agile Methods*

"Battlefields are messy, turbulent, uncertain, and full of change. No battlefield commander would say, 'If we just plan this battle long and hard enough, and put repeatable processes in place, we can eliminate change early in the battle and not have to deal with it later on'."(Highsmith, 2002) In this piece of writing Highsmith (2002) explains that a growing number of software projects which are work the same as a battle and they are called 'extreme projects'. This is where the concept of agility becomes important.

The origin of Agile methods go back a long way even though they were properly introduced and started to gain interest in the software industry during the last few years. As mentioned earlier, as a result of built up frustration within the software developers on structured and planed methods in the mid-1990s, development teams started to use early versions of some of the agile methodologies such as Extreme Programming (XP), SCRUM and Dynamic Systems Development Method (DSDM).

The Agile methodologies describe a number of principles which in summary put the human factor (customers and developers) first over processes and plans. The highest priority principle is to satisfy the customer through early and continuous delivery of software. According to Miller (2001) there are a number of characteristics of agile methods from a fast delivery view, which ultimately shortens the software project life-cycle:

1. Modularity – This is on the process level of development
2. Iterative – Consider short development cycles which enables to clear error faster and more accurate
3. Time bound – iterative cycles ranging from one to six weeks
4. Parsimony – remove all the unnecessary activities in the development
5. Adaptive – Take faster action against possible new emerging risks
6. Incremental – A functioning application software, build up in smaller steps
7. Convergent – Minimizes risks
8. People-oriented – Agile favour people who are involved over the process and technology
9. Collaborative – Active communication.

In Barry Boehm's IEEE computer article (2002) it is mentioned that according to Highsmith and Cockburn (2001) there are several critical people-factors which agile highlights, such as amicability, talent, skill, and communication. Highsmith and Cockburn (2001) further describes, what is new in agile is not the behaviors or practices they use but the recognition of users or any other people involved as the primary sources which drive the project to a success.

Agile does not require highly-capable people to execute its practices in a software project environment. However, it requires tacit knowledge and lot of expertise to function successfully. Due to this reason agile has a minimum use of fully completed documents. Boehm warned that there is a possible risk that this situation may lead to architectural mistakes, which are hard to find and correct by any external party.

c) *Definition of Agile*

Agile cannot be given with a constant definition. Different practitioners have different wordings according to their experience and understanding. But agile can be explained in few characteristics that are considered as the core characteristics.

- Iterative and incremental process
- Simple and easily adoptable
- Collaboration of all the parties such as users, customers, developers, project managers, etc.
- Produce high quality software within the requirements, budgets and the time scale.

Following are different definitions from different expert practitioners.

"Agile is an iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams with 'just enough' ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders."(Ambler, 2001)

"Agile is a conceptual framework generally centred on iterative and incremental delivery of working software, driven by the customer. The iterative part suggests that we are repeating, or iterating, a complete lifecycle of development over a short, fixed span of time. With each of these iterations, we ship some working subset, or increment, of features." (Langr, 2006)

d) *SDLC for Agile*

According to Ambler agile SDLC composed of four phases Iteration0, Development, Release and production.

Iteration 0:

1. Initial time of the agile project.
2. Modeling and initial architecture of the project.
3. Setting up the environment of the project.

Development Phase:

Incrementally deliver high quality software which meets the changing needs of the use.

Release Phase:

In this phase agile practitioners transition the system into production.

Production Phase:

The fundamental goal of this phase is to keep the system running and help users to use the software.

e) Agile Methodologies

Agile manifesto provides an ideological environment to modern so called "agile" software development with its defined values and principles. A survey conducted by Cutter Consortium with regard to the methods been used in the software development field revealed that 54% of the users use their own in-house development methods, which can be explained within the agile boundaries. Among the defined methodologies in agile the most popular methods were Extreme programming, Feature Driven Development and Adaptive Software Development. The purpose of this section is to introduce few of the widely used agile methodologies identifying the roles, process, responsibilities and practices. The following methods will be included for discussion: Extreme Programming (XP), Dynamic Systems Development Method (DSDM) and SCRUM.

i. *Extreme Programming*

a. *Outline*

Extreme programming (XP) evolved from the frustrations and the problems caused from traditional plan-driven methods, which were the only development solutions in the software industry for a long time (Beck, 1999). XP was developed and brought in to practice in the mid 1990's by Kent Beck, Ward Cunningham and Ron Jeffries (Paulk, 2001) as a result of a project they been working. The main features which XP emphasizes are those that they identify as the prerequisite for effective software development which are improving communication, getting feedback, simplicity and proceeding with courage (Cockburn, 2001). Even though these practices started as just a better ways of development rather than traditional methods with time they showed success. This was the root for XP. XP has widely influence on the principles in the agile manifesto (Kalermo & Rissanen, 2002).

There are different theories and arguments about XP whether it is actually a method and how extreme this methods is. Paulk (2001) argues that these practices are actually just commonsense practices that any discipline method would have and not something extreme. Beck (1999) who is one of the founders of XP states that XP is a fresh and new methodology and the term "Extreme" comes from taking these commonsense practices to extreme levels.

XP is based on the following five important values.

- Communication - "Problems with projects can invariably be traced back to somebody not talking to somebody else about something important." (Beck, 2000).
  XP focuses lot on face to face or oral communication and its techniques encourages in maximizing interaction. This is valued on the observation that most project difficulties occur because individuals or teams have not spoken with other parties to clarify questions, to collaborate, or to obtain help.
- Simplicity – Rather than try to capture all features and complicate, Design the project in the simplest way to meets the customer's needs. The value highly stresses on the point, only design and code the current requirements obtained rather than to anticipate and plan for unstated requirements.
- Feedback – The development team(s) obtain feedback from the customers at the end of each iteration and release. The next iteration drives with the consideration of this feedback. There are very short design and implementation feedback loops built into the methodology via pair programming and test-driven development (Williams, 2003).
- Courage - The best thing about XP is that the other three values give the team to have courage in their actions and decision making. The team decides which parts will be done at which stages. Further, this encourages the team to avoid any pressure for unrealistic deadlines or requirements.
- Respect - Team members always have to care about each other and about the project.

b. *XP Lifecycle*

The life cycle of XP consists with five phases. There are Exploration, Planning, Iteration to Release, Product ionizing, Maintenance and Death. The following diagram illustrates how these phases work together in the life cycle.

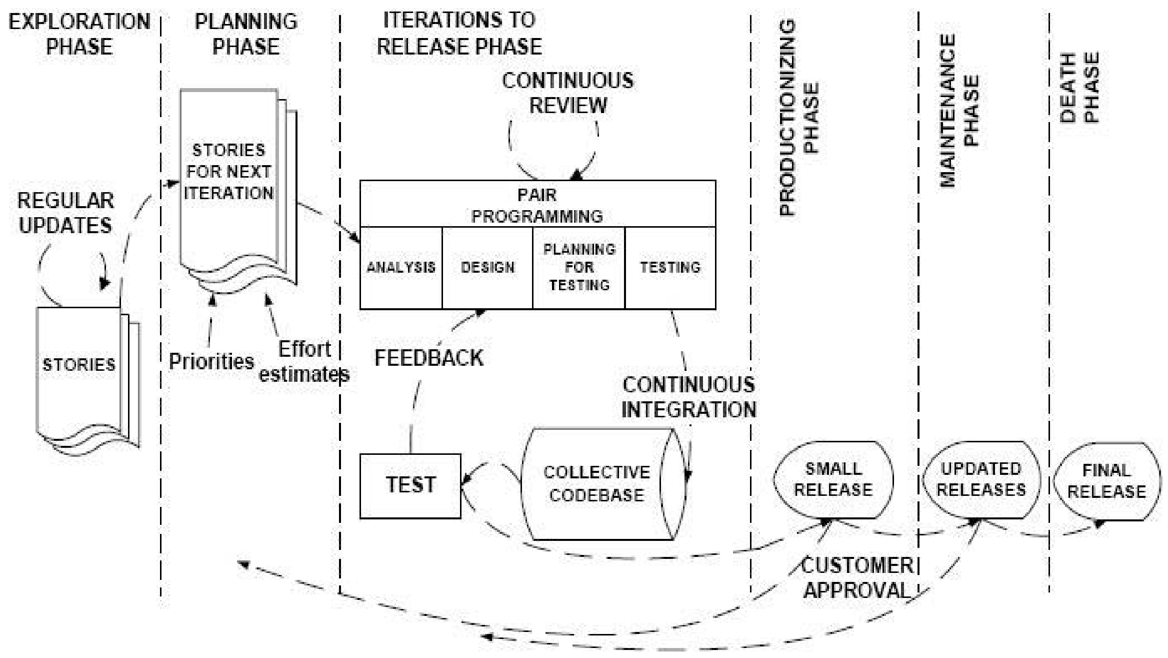

Figure 1: XP Life cycle process (Source: Abrahamsson et al. (2002))

- Exploration phase – Story cards are used by customers to express the features they want in the system. In each story card they have to write a feature they wish to have in the system. Mean while the technical teams focus on the tools and technologies they are going to use in the project. They get familiar with those tools as well. They test the technologies and the proposed architecture possibilities by building a prototype of the system. Depending on the project scope and the teams' familiarity with the technologies this phase spans from few weeks to few months.
- Planning phase – Considering all the stories, prioritize the features to be delivered in the first set of the release of the system. The development teams estimate the time required for different features and then agree upon the deliveries for the first release. The first release of the system can take up to two months and the planning phase may take few days.
- Iteration to release phase – The schedule set up for the first release is divided into small iterations before the actual first release. The first iteration builds system architecture for the whole system by selecting and analyzing the stories which includes the features. The customers decide which story to include in each of the iterations. Further the customers can create functional test for the system. These will be used to check the accuracy of the system and may use in the future. Iteration is around one to four weeks each for implement. Once the iterations are done the system is ready for production.
- Product ionized phase – This phase runs faster than the others, which means that the iterations can be reduced to one week instead of three weeks. The system has to be extra tested for performance before release to the customers. New changes found here has to be decided before start working on them. Postponed ideas will be documented to build later.
- Maintenance phase – After the product is product ionized and released for customer use, teams have to make sure that system in the production running and also produces new iterations. This phase need an effort for customer support tasks In order to maintain these operations. Thus, the maintenance phase may require new people into the team and also changes in the development structure.
- Death phase –The project comes to this phase when there are no more requirements from the customers. But there are other concerns such as reliability and performance before reaching this point. Since there are no more requirements to be added to the system all the documents been written at this stage. On the other hand when the project does meet the requirements and it is expensive for further development, it can reach death phase.

c. *Responsibilities and Roles of XP*

There are specific roles in XP for different tasks. This makes work much easier to handle as they are divided with clear roles. The following describes these roles according to Beck (2000) and Abrahamsson et al. (2002).

- Manager – Makes all the decisions and is responsible for the team and its issues. He or she has the right to form the team, obtain and allocate resources, manage people and problems. In order to do all above, he or she communicates with the team to understand the present situation. The manager interfaces with external groups as well including the customers.
- Coach – Responsible for the whole process as a whole. Teaches team members about the XP process as necessary, intervene in case of issues. Keep of track of the ongoing process. A sound knowledge of XP is very important to this role. The coach is typically a programmer and not a manager.
- Tracker – Provides feedback. He or she regularly collects user story and acceptance test case progress and other estimates from the developers and gives feedback on how accurate they are to make better future estimates. Further tracker traces the progress of iterations and evaluate if the project goals are reachable within the allocated time with the current resources. The tracker is a programmer, not a manager or customer.
- Programmer – Writes tests, design, and code and try to keep them simple and definite as possible. They refactor code identify and estimates tasks and stories.
- Tester – Helps customers write and develop functional tests. They run functional test often to broadcast results and they maintain the test tools.
- Customer – Writes stories and acceptance tests. Selects stories for a release and for an iteration. One individual from the customer organization or a group of customers can be involved in the sections, or a customer representative can be chosen from within the development organization that is external to the development team.

d. *Technical Practices*

The initial version of XP had defined programmer- centric technical practices. This was published in 2000 by Beck.

- Planning game
- Short releases
- Metaphor
- Simple design
- Testing
- Refactoring
- Pair programming
- Collective ownership
- Continues integration
- 40 hour week
- On-site customer
- Coding standards
- Open workspace
- Just rules

XP practices were changed to include 13 primary practices and 11 corollary practices in 2005 (Beck, 2005). The primary practices are intended to be useful independent of each other and the other practices used, though the interactions between the practices may amplify their effect (Williams, 2007).

ii. *SCRUM*

a. *Outline*

"The relay race approach to product development may conflict with the goals of maximum speed and flexibility. Instead, a holistic or 'rugby' approach – where a team tries to go the distance as a unit, passing the ball back and forth – may better serve today's competitive requirements." (Takeuchi and Nonaka, 1986)

SCRUM is also a member from the agile development processes family. Scrum is a process skeleton that includes a set of practices and predefined roles. It provides you a set of guidelines to develop software from its design stage to its completion. Scrum is best suited for the projects with rapidly changing or highly emergent requirements. It is a Simple and scalable method which means easily combined with other methods and doesn't prescribe engineering practices. According to the article on scrum by Clifton and Dunlap (2003b) there are few software development issues scrum addresses for a better software production.

- Chaos due to changing requirements - The real or perceived requirements of a project usually change drastically from the time the product is designed to when it is released. Under most product development methods, all design is done at the beginning of the project, and then no changes are allowed for or made when the requirements change.
- Unrealistic estimates of time, cost, and quality of the product - The project management and the developers tend to underestimate how much time and resources a project will take, and how much functionality can be produced within those constraints. In actuality, this usually cannot be accurately predicted at the beginning of the development cycle.
- Developers are forced to lie about how the project is progressing - When management underestimates the time and cost needed to reach a certain level of quality, the developers must either lie about how much progress has been made on the product, or face the indignation of the management.

b. *SCRUM Lifecycle*

Scrum has a process which has to be followed by any organization or team that adopt this methodology. As figure 2 illustrates the projects development happens via a series of month-long iterations called Sprints. Scrum is ideally suited for projects with frequently changing or highly emergent requirements. The Product Backlog lists the work to be done on a Scrum project. It lists all desired changes to the product. A Sprint Planning Meeting is held at the start of each sprint during which the Product Owner prioritizes the Product Backlog and the Scrum Team selects the tasks they can complete during the coming Sprint. These tasks are then moved from the Product Backlog to the Sprint Backlog.

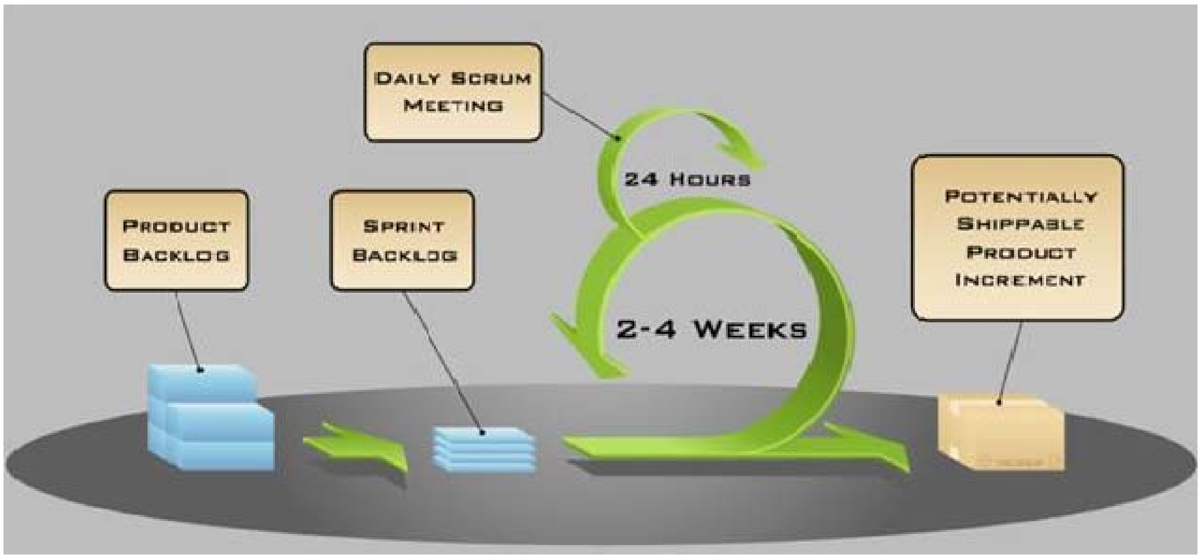

Figure 2: Scrum lifecycle (Source: www.davenicolette.net)

In order to help the team stay on track, a brief daily meeting, called the Daily Scrum, is conducted each day during the sprint. At the end of each sprint the team demonstrates the completed functionality at a Sprint Review Meeting (Mountain Goat, 2008).

c. *Responsibilities and Roles of SCRUM*

Scrum implements its iterative and incremental process through three roles. All management responsibilities are divided between these three roles (Schwaber, 2007).

- Product Owner – The product owner is responsible for the project, managing, controlling and creating and prioritizing the Product Backlog. He or she is selected from the other parties such as management, customers and the scrum master. Product owner selects what will be included in the next iteration/Sprint, and reviewing the system (with other stakeholders) at the end of the Sprint and makes the final decisions related to the product backlog (Abrahamsson et al., 2002).
- Scrum Master – Scrum master makes sure that the project runs according to the plan. He also makes sure that the team follows the practices and rules in scrum. It is his responsibility to reinforce the product iteration and goals and the Scrum values and to conducts the daily Scrum Meeting. Scrum master interacts with the management and the customers during the project and also responsible in the iteration demonstration (the Sprint Review), listens to progress, removes impediments (blocks), and provides resources. The Scrum Master is also a Developer. He takes part in product development as well (Schwaber, 2007).
- Developer – Member of the Scrum team. The Scrum Team is committed to achieving a Sprint Goal and has full authority to do whatever it takes to achieve the goal. The team may consist of developers between 5 and 10.
- Customer – Involves in the tasks of creating the product backlog. They provide ideas and other information for feature to be developed in the system.

d. *Technical Practices*

SCRUM does not mention any particular practices like other methodologies. Instead Scrum focus on some management practices and tools to avoid chaos in different stages of the process. Following are the practices used in scrum development (Schwaber and Beedle, 2002, cited by Abrahamsson et al. (2002)).

- Product backlog
- Effort estimations
- Sprint
- Sprint planning meeting
- Sprint backlog
- Daily Scrum meeting
- Sprint review meeting

Throughout the life cycle of SCRUM these practices are been carried out. Each and every role has their duties towards the success of the project during these practices.

iii. *Dynamic Systems Development Method*

a. *Outline*

The Dynamic Systems Development Method (DSDM) was first developed in the United Kingdom around the mid to late 1990s by a group of people from a business background. It was totally not related with technical perspective. This can be said as one of the heavier Agile approaches available (Coffin and Lane, 2007). It was initially developed as an addition to Rapid Application Development (RAD), incorporating best practices from the business-oriented environments.

DSDM is a well ordered, commonsensical process focused on delivering business solutions quickly and efficiently. It has similarities to SCRUM and XP in many ways, but it has its best uses where the time requirement is fixed (CliftonandDunlap, 2003a). DSDM focuses on delivery of the business solution, rather than just team activity. It ensures the feasibility and business sense of a project before it is created. The cooperation and collaboration between all interested parties is an important fact in DSDM. This method makes heavy use of prototyping to ensure all the involved parties have a clear picture of all aspects of the system.

Unlike in traditional development methodologies where functionality is fixed, and time and resources are variable, in DSDM, time is fixed, and functionality is variable (CliftonandDunlap, 2003a). The following figure best illustrates this scenario.

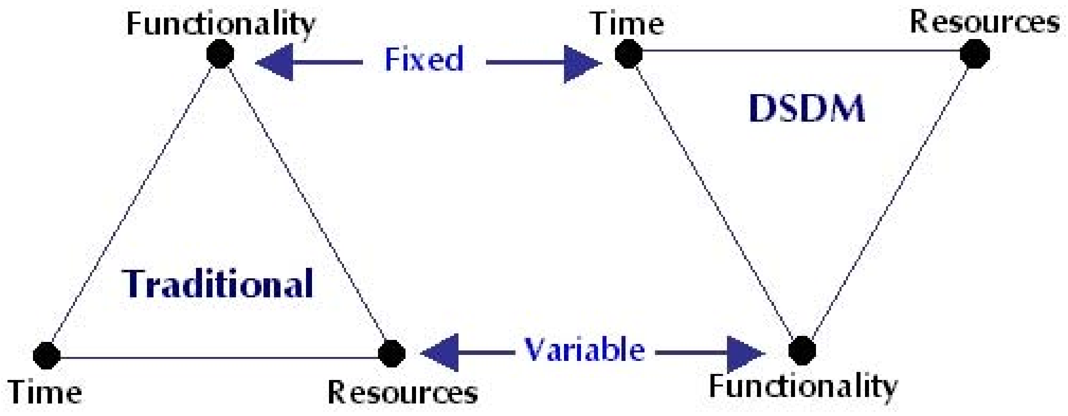

Figure 3: Traditional and DSDM (Source: http://www.codeproject.com)

DSDM respect the needs that larger organisations have to manage portfolios of projects, architectural diversity, resources and to make project decisions on the foundations of a fully considered Return on Investment. DSDM, then, had to, and still does, accommodate these corporate pressures more readily than most other agile approaches by considering a project in a wider context than software delivery alone (DSDM Consortium, 2008). It does this by having a more liberal lifecycle, by presenting and operating the agile development techniques in a way that makes as much sense to the wider organisation as it does to the project teams and by defining responsibilities within key roles to manage the corporate dependencies and preconditions (DSDM Consortium, 2008).

b. *DSDM Lifecycle*

The DSDM lifecycle consists of 4 main phases. The diagram below explains these phases. The phases are Feasibility Study, Business Study, Functional Model Iteration, Design and Build Iteration and Implementation. These phases operate in an iterative manner and have ability to jump to any other phase if required. This is the significant difference made in DSDM compared with the traditional water fall model.

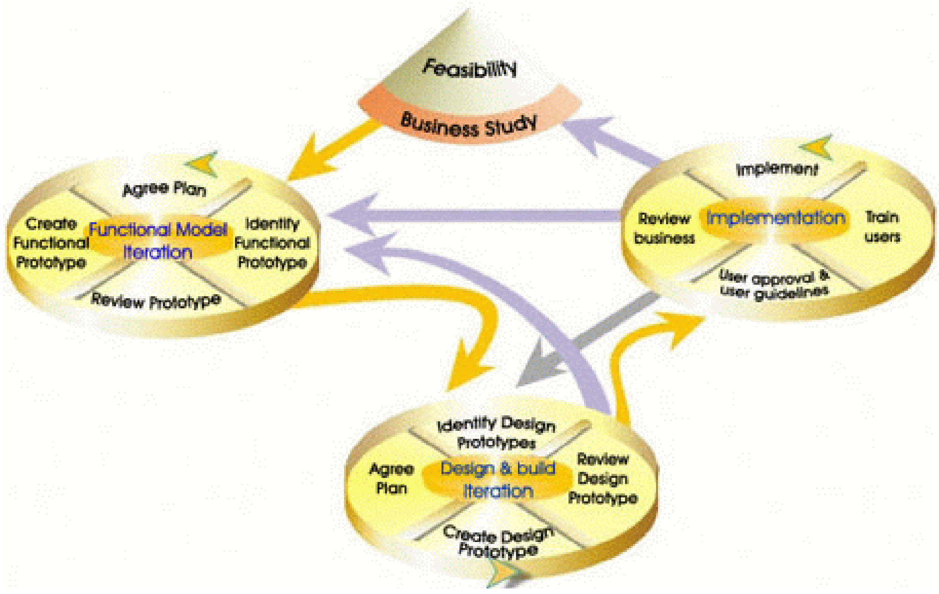

Figure 4: DSDM Lifecycle (source: www.topdownsoftware.com)

There are nine important guiding principles defined for DSDM. These principles describe what DSDM should be and how it should operation when using for a specific project. The following are from Moonzoo (2007)

- Active user involvement is imperative.
- DSDM teams must be empowered to make decisions.
- The focus is on frequent delivery of products.
- Fitness for business purpose is the essential criterion for acceptance of deliverables.
- Iterative and incremental development is necessary to converge on an accurate business solution.
- All changes during development are reversible.
- Requirements are base lined at a high level
- Testing is integrated throughout the life-cycle.
- Collaboration and cooperation between all stakeholders is essential.

c. *Responsibilities and roles of DSDM*

Following are several key roles that should be filled by members of the team as describe in an article by Clifton and Dunlap (2003a).

- Ambassador - The person who acts as intermediate between the users and the development team. He manages the development team, and usually has a good overall understanding of how the system will work.
- Visionary – This role is the driving force behind the project. This role keeps the project steered on course towards the business goals. Often is the person who started/thought of the project.
- Advisers - People who have practical knowledge in areas of the business that need to be automated, and/or in the technologies needed to automate these areas.

d. *Technical Practices*

There are nine principles at the core of the DSDM methodology. Some clearly overlap with XP and similar approaches. However, DSDM's principles are sufficiently robust to minimize damage to schedules and resources when a business process radically changes or a major component's design is faulty—problems that could cripple an XP project (Robinson, 2002).

- Active user involvement is a must.
- Design groups are empowered to make system development decisions.
- Frequent and regular delivery of components is a priority.
- The primary acceptance criterion for a system or component is its fitness for business purposes—the design driver is business benefit.
- The business solution is the goal, and iterative and incremental development is necessary to converge on that solution.
- All changes made during development are reversible.
- Initial requirements are defined very generally.
- Testing is not a specific project phase; it occurs constantly.
- It's essential to have collaboration and cooperation between all project participants.

f) *Traditional Software Development*

i. *Outline*

"By applying a methodology to the development of software insights are gained into the problems under consideration and thus, they can be addressed more systematically. Software should comply with the important quality requirements of timeliness, relevance, accuracy and cost effectiveness. Software engineering aims to bring to bear the more rigorous methods used in the engineering world in the software development world." (Georgiadou, 2001).

Traditional software development methodologies are the first methods of software development. They are also known as heavyweight methodologies. They are considered to be the classic way of developing software. These methodologies are mostly based on a series of sequential steps, such as requirements definition, solution building, testing and deployment.

Traditional software development methodologies require defining and documenting a stable set of requirements at the beginning of a project.

ii. *Waterfall Model*

a. *Outline*

The Waterfall model is known as the classic model of software development. The Waterfall model also known as the "top down" approach, was proposed by Royce (1970). Until the mid 80's it was the only model with a level of general acceptance. It was derived from models used in traditional engineering activities with the objective of establishing an order in the development of large software products. It is more rigid and less manageable compared with other software development models.

The Waterfall Model is one of the most important models ever published. It is a reference to others, and serves as the basis for many modern projects as well. Its original version was improved over time and is still frequently used today (Peters and Pedrycz, 2000). A great part of the success of the Waterfall Model is due to the baseline management, which identifies a fixed group of documents produced as a result of each phase of the life cycle (Peters and Pedrycz, 2000). The produced documentation includes more than text files, it has graphical representations of the software and even simulations.

b. *Waterfall Model Life Cycle*

Waterfall model phases are executed systematically in a sequential order. The model usually has the following phases: Analysis, Design, Implementation, Testing, Deployment and Maintenance.

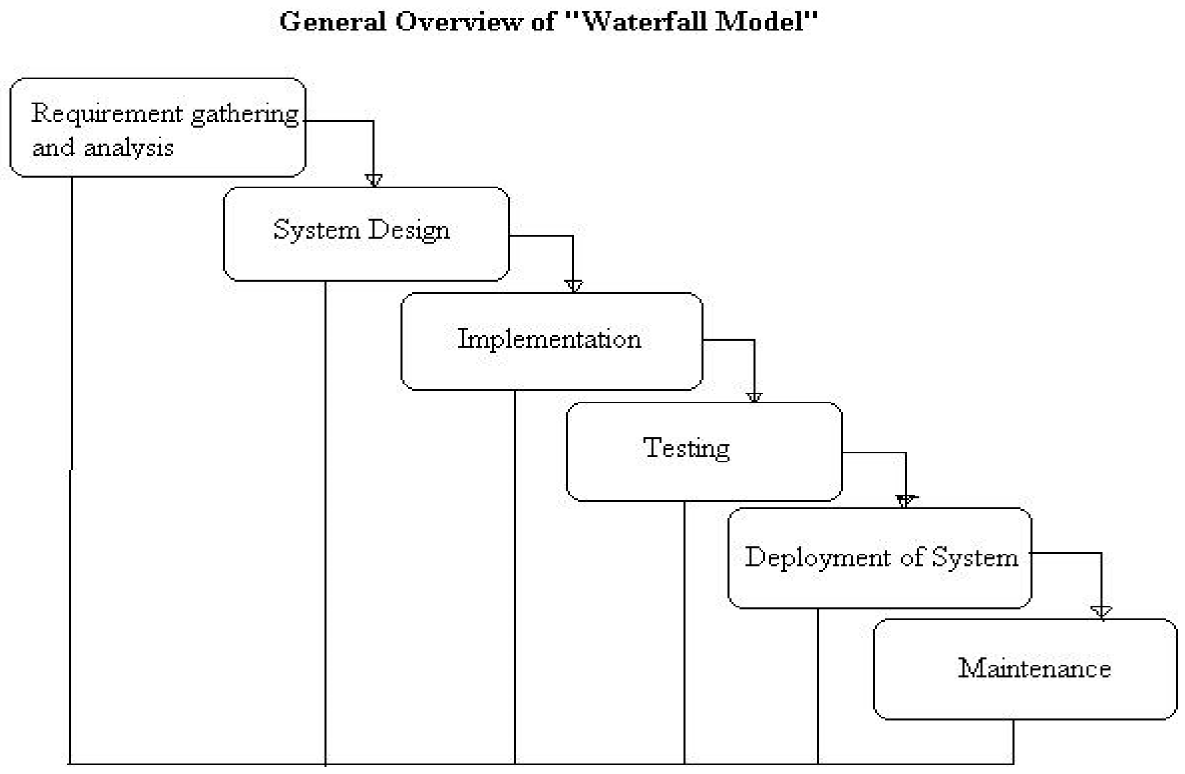
General Overview of "Waterfall Model"
Requirement gathering and analysis
System Design
Implementation
Testing
Deployment of System
Maintenance

Figure 5: Waterfall model (Source: www.Buzzel.com (2000-2009))

Requirement gathering and Analysis – This is the phase where all the requirements to be developed are captured. This is done by conducting consultations, interviews, observation and so on. A document called requirement specification is created including all the gathered requirements at the end of this phase (Parekh, 2005a).

System design – Looks at the overall system in a design and architectural level before starting actual coding. This is to get an idea how the system look like at the end of the project. All hardware, software and resource requirements are considered here and finally create the system design specification to start the next phase.

Implementation and unit testing – The actual coding begins in this phase. According to the system design spec system is built in small units. Each of these units are tested to ensure that it servers the purpose that unit is built (Parekh, 2005a).

Integration and system testing - In the previous phase the system is built in units. This phase focuses on getting these units together. The system is build by putting the units together. Units are tested with each other to ensure that they work and communicate with each other and give the final outputs which are expected from the whole system (Parekh, 2005a).

Operations and maintenance – This phase is normally considered the longest of all. Issues and errors of the system which were not found during the development stages come alive once the system starts to operate in a live environment. This will normally happen time to time. So this phase is called maintenance (Parekh, 2005a).

iii. *Spiral Model*

a. *Outline*

The spiral model was introduced by Barry Boehm in 1980s, based on experience with various refinements of the waterfall model as applied to large software projects. This method combines elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts (Boehm, 1988). There are four main phases of the spiral model (Boehm, 1988):

- Objective setting – Specific objectives for the project phase are identified.
- Risk assessment and reduction – Key risks are identified, analyzed and to reduce these risks information is obtained.
- Development and Validation –For the next phase of development an appropriate model is chosen.
- Planning – For the next round of spiral the project is reviewed and plans are drawn up.

b. *Spiral Model Lifecycle*

As shown in figure 6 there are four main phases in spiral model. They are Planning, Evaluation, Risk Analysis and Engineering. These phases follow one after another in an iterative manner. The objective is to eliminate the problems occurred in the waterfall model. In an article by Parekh (2005b) mentions that even though the iterative approach became a solution to waterfall model issues, spiral model requires people with high skills in the area of planning, evaluation, risk and customer relations. The project becomes more costly than planned due to the demand for more than one iteration cycle. Following describes the main phase in spiral model.

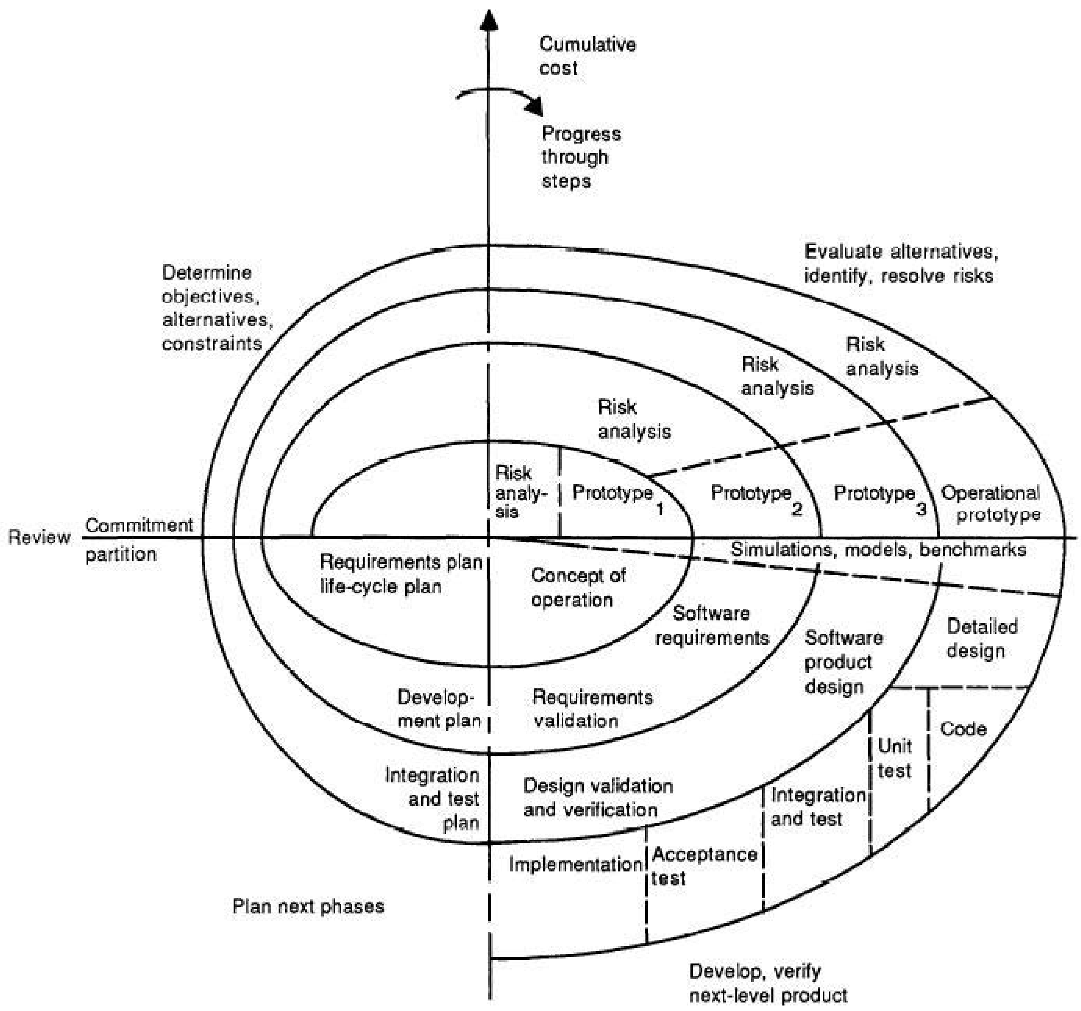

Figure 6: Spiral model (Source: Boehm (1988))

Plan phase – This phase gather and finalize the objectives and constraints of the project and documented. These are kept locked in order to decide on the approaches and strategies of the project.

Risk analysis – This is considered as the most important phase of the model. All the approaches and strategies are analyzed for risk factors. Prototyping is used to find solutions and to develop a low cost and quality system if there are any indications of risk.

Engineering – This is the development phase. Development outputs are carried through all the phases iteratively for improvements.

Customer evaluation – The built product is passed on to the customer in order to receive feedback. This phase is expected to come across possible errors and/ or changes. This is similar to system testing.

iv. *Unified Process*

a. *Outline*

Unified process is actually not a process rather it can be called as an extensible process which can be customized according to the nature of different projects or organisations. Every approach such as modeling is organized into workflows in the Unified Process (UP). UP is performed in an iterative and incremental manner and some of the key features of the UP are described below (Booch, 1994):

- UP consists with an architecture based on components which creates a system that is easily extensible, supports software reuse and intuitively understandable. The component commonly being used to coordinate object oriented programming projects.
- It uses modeling software such as UML to represent its code graphically as a diagrammatic notation to allow less technically capable individuals, but with a better understanding of the problem to come up with a greater input.
- The use of use-cases and scenarios to manage requirements seems to be very effective at both capturing functional requirements and help in keeping sight of the anticipated behaviors of the system.
- Since the design is done in an iterative and incremental manner it helps reduce project risk profile. Further it allows greater customer feedback and help developers stay focused.

- Verifying software quality is very important in a software project. UP assists in planning quality control and assessment built into the entire process involving all member of the team.

b. *UP Lifecycle*

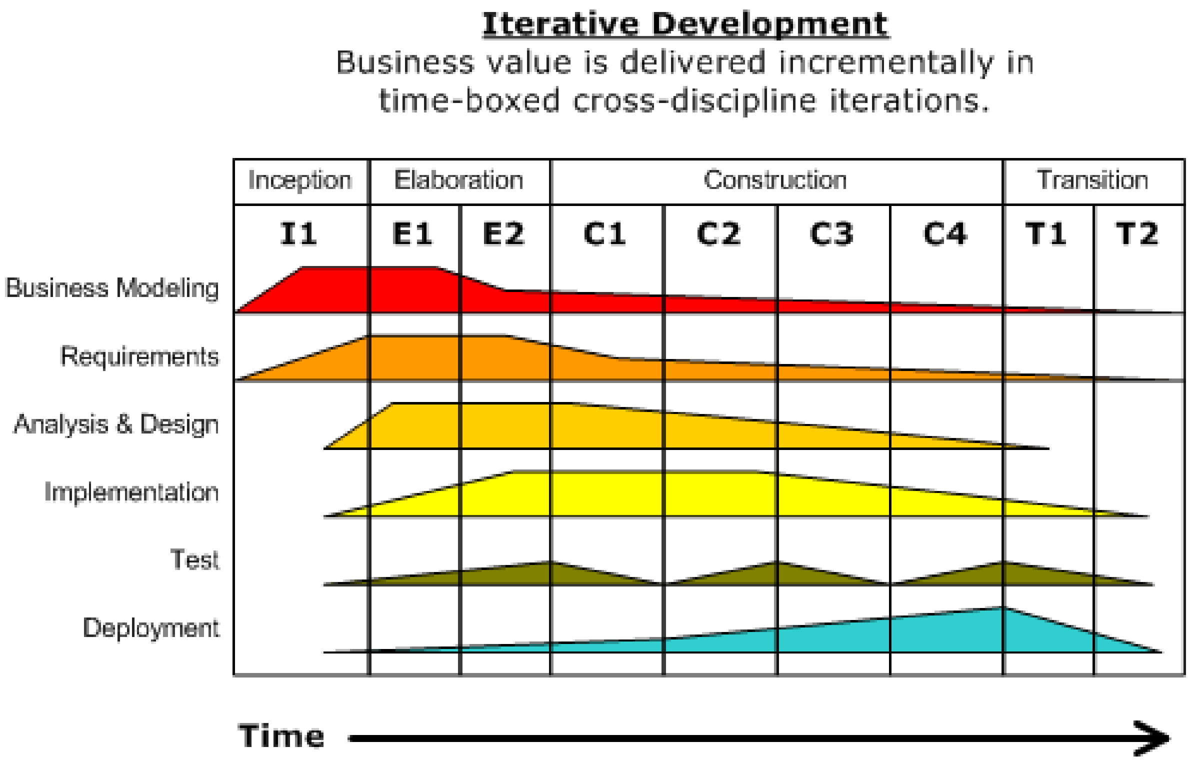

Figure 7: UP lifecycle (Source: Wikipedia at http://en.wikipedia.org/wiki/Unified_Process)

The above diagram indicates the four phases in UP lifecycle. These four phases are described below (devdaily).

- Inception – This phase creates a business case at the end of the process. The feasibility of the system is measured and the scope of the system is defined.
- Elaboration – The basic architecture of the system have been produced and a construction plan is agreed. Furthermore a risk analysis takes place and major risks are addressed.
- Construction – The system is produced and released for testing. This is not a full functioning system. A working system should be available and sufficient enough for testing under realistic conditions.
- Transition – The system is finally up to the standard to go in a live environment. So it is introduced to the stakeholders and intended users. Once the customers and the project team agreed that the intended target is met and the user is satisfied the system is completed.

There are approximately 50 work steps that has to be completed in UP during the process (Larman, 2004). All this documentation and this rigid approach add a lot of complexity to UP. UP has predefined roles to the project team making it less flexible in working.

g) *Comparison of Agile and Traditional Methods.*

In the previous section some discussions were there on both agile and traditional methods to identify the characteristics of these methods. It is important to do a comparison on these two methods in order to understand the differences that will affect different projects.

Table 2: Comparison of Agile and Traditional (Source: Khan and Balbo, 2004)

	Agile Methods	Heavy Methods
Approach	Adaptive	Predictive
Success Measurement	Business Value	Conformation to plan
Project Size	Small	Large
Management Style	Decentralized	Autocratic
Perspective to Change	Change Adaptability	Change Sustainability
Culture	Leadership-Collaboration	Command-Control
Documentation	Low	Heavy
Emphasis	People-Oriented	Process-Oriented
Cycles	Numerous	Limited
Domain	Unpredictable/Exploratory	Predictable
Team Size	Small/Creative	Large
Upfront Planning	Minimal	Comprehensive
Return on Investment	Early in the project	End of the project

It is mentioned in the early sections that traditional methodologies were ruling the software industry for a long time until practitioners begin to understand some of the drawbacks largely affecting the software projects. Extreme programming became popular in the industry when it was introduced in late 90's by Kent Beck. Then agile was introduced based on the concepts used in XP. Agile handle projects mostly in a volatile and uncertain environments. But with the passage of time practitioners came to realize that agile cannot handle all types of software projects as it has some limitations as well. Both of these methodologies have their strengths and weakness. Now the organizations tend to use the strengths of both together in their projects. There are three main factors which need to be considered when selecting a methodology. They are people, project size and risk.

i. *People*

This is one of the main important factors considered in software development. Especially agile methodologies strongly believe in human factor. Bohem and Turner (2003) believe that "In essence, software engineering is done 'of the people, by the people, and for the people.'" The agile manifesto stresses about the importance of the human interactions and customer collaboration in their basic values of agile methodologies (Fowler and Highsmith, 2001).

Developers and customers are the most important categories in people needed for software development. When using agile methodologies the people factors for developers were identified as skill, talent, communication and amicability (Bohem and Turner, 2003). Agile unlike traditional methodologies encourage working closely with the customers. This is important for a successful development environment.

The organization's culture has an impact on the people factor. If the developers are under the tight rules of the organization, it is hard to adopt agile since the developers will not get the maximum out of agile methodologies.

ii. *Project Size*

Project size of software is another major factor and considered as a challenging factor. In the early stages of project size estimation it was measured by predicting the number of lines of code the project may need (Dekkers, 2005). This is one of the limitations agile is facing today. For most of the large scale projects which involve more than 50 software developers agile seems to be working in a negative manner. This was shown in a study conducted by ambler (2008). Cockburn (2008) states that "A larger methodology is needed when more people are involved. Larger means containing more control elements." This statement is further supported by the following figure.

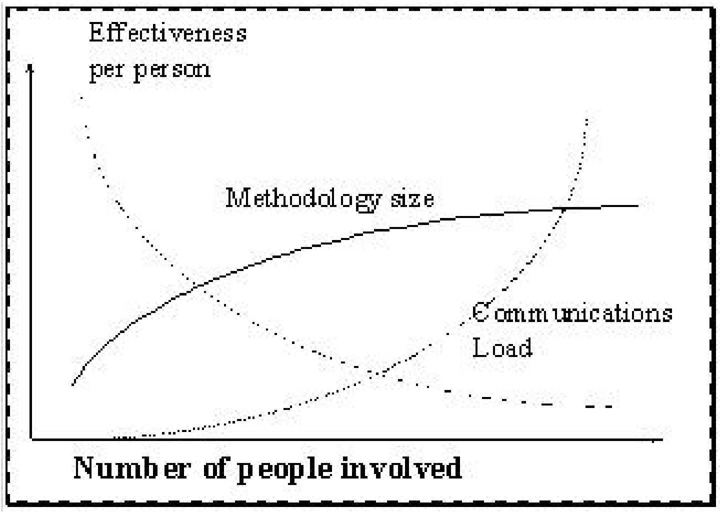

Figure 8: The effect on communication with people

The communication load rises when the number of people is increasing. Then the need of a bigger methodology occurs since that is a media of managing the people and therefore the communication. With this graph Cockburn further explains that "one should not expect a small-team methodology to work properly for a big team, and one need not use a big-team methodology for a small team." However Vaihansky et al (2006) argues that agile methods such as XP and SCRUM can be used successfully for large projects. "Best current Scrum practice is for local Scrum teams at all sites to synchronize once a day via a Scrum of Scrums meeting." The organization should decide on which type of methodologies they are going to use depending on the time and project size.

iii. *Software Risk*

Software project risk may result in lots of problems. Budget and plan overruns and unable to meet the expectations of the uses and many more (Renhui and Fengyong, 2007). There are few categories of risk according to Renhui and Fengyong (2007), and there are;

- Team risk
- Environmental risk of organization
- Demand risk
- Plan and control risk
- User risk
- Complexity risk

An organization should be careful when handling these risks. Traditional methodologies are used for large critical systems with security and reliability such as military systems. However, for the systems that can be made quickly and have lots of uncertainty, Agile is the most appropriate methodology. For example a system expected lots of change of requirement during the development phase through customer involvement agile is the best methodology as it can respond to changes faster.

III. Research Methods

a) *Introduction*

This chapter discuss about the methodology used by the researcher to present a research into the statement of aim. The main purpose of this section is to evidently define the specific guidelines which will make possible the researcher to substantiate the achieved hypothesis. In brief, this section discusses about the ideas, which are used in the course of primary and secondary.

b) *Research Philosophy*

Research philosophy depends on the way a researcher thinks about his/ her development of knowledge (Saunders et. al., 2003). The major research philosophy theories are Positive, Phenomenology and Realism (Maylor & Blackmon, 2005).

Positive or scientific method affirms that there is just one truth about the world. It is understood that such truth is objective and does not entail any value judgments. Finding this truth requires a process based on a deductive method for which data must be collected. In this sense positivist researchers stand that the data is not affected by the researcher opinion and that the more objective the data collection the better. (McNeill, 1985)

Usually the data is collected, interpreted and analyzed following the quantitative method and according to a statistical approach. Data collection might be achieved through surveys. The survey aim is to test the original hypothesis and therefore, to establish the truth of a specific phenomenon. The relevance of this kind of approach resides in its objectivity, since the results obtained are independent of the subjectivity of those involved in that process.(McNeill, 1985)

Phenomenology (ethno methodology), has as main principle that there is not a unique truth. According, the explanation of a phenomenon emerges from different points of view people affected and involved have in relation to the phenomena analyzed. This is an action- reaction process. Every single person has his own interpretation of the world and phenomenon. In this sense, there are different truths and realities, and sharing meanings and interpretations vary depending on the context. (McNeill, 1985)

Realism shares some philosophical views with positivism, since it is based on the impression that there is an intention reality, which is self-governing of human beings' thoughts and beliefs. However, realism also recognizes that humans are not substance to be considered in the style of natural science. On the other hand it takes social influences into account. Realism recognizes the importance of the fact that those social influences, although are independent of individuals, affect the way people make sense of their world, whether they are conscious of these forces or not. (Saunders et. al., 2003)

The comparison between the characteristics of each research philosophy is summarized in the following Table 3.

Table 3: Characteristics of Positivism, Phenomenology and Realism research philosophy (Source: Adapted from Saunders et. al., 2003)

Positivism	Phenomenology	Realism
Objective truth analysis Value-free data collection Law-like generalization Quantitative approach	Subjective truth analysis People's account, motives and intentions Complex and dynamic Qualitative approach	Socially constructed environment analysis Independent reality Social influences recognized Qualitative approach

In this research, researcher uses realism philosophy because it helps to find out the research questions more efficiently.

c) Research Design

According to Kerlinger (1994) "A research design is the plan, strategy and structure of exploration conceived so as to achieve answers to research questions and to control variance."

Sekaran (1992) states, research has been defined as:

"An organized, data based critical, systematic, scientific enquiry and exploration into a particular difficulty, undertaken with the intention of finding answers or solutions to it."

According to Saunders et. al.(2003), there are three different types of research design, which are; 1) Exploratory 2) Descriptive 3) Explanatory.

The concept of each is discussed below.

i. *Exploratory*

Exploratory research is a kind of investigate conducted because a problem has not been evidently defined. Exploratory research helps decide the best research design, data collection process and variety of subjects. Investigative research relies on Secondary research. Though, research that is conducted with a desire to discover are called an exploratory research.

ii. *Descriptive*

Descriptive analysis describes data and characteristics about the society or phenomenon being studied. If the function of the study is to describe, the study is measured to be descriptive in character. It mainly gives the researcher a choice of aspects, perspective, levels, terms and concepts, as well as to observe, register, systemize, classify and interpret.

iii. *Explanatory*

Explanatory research is useful when the issue is previously known and has a explanation of it. The ambition to know "why" to provide details is the point of explanatory research. It builds on descriptive and exploratory research and goes on to identify the cause for something that occurs. Explanatory research looks for reasons and causes.

The different between exploratory, descriptive and explanatory research design

Table 4: Characteristics of exploratory, descriptive and explanatory research design (Source: Adapted from Saunders et. al., 2003)

Exploratory	Descriptive	Explanatory
A study to find new insights Useful for clarifying the understanding of the problem Qualitative approach	A study to describe an accurate profile of persons, events or situations Useful for giving details of incidence or phenomena and for predictive findings Quantitative approach	A study to find casual relationship between variables Useful for explaining the relationship of two or more incidents in terms of cause and effect Quantitative approach

In this research, the researcher has explored "Marketing strategy in fast food restaurant" in particular through his own literature view. The researcher has tried to explore the relationship between the impacts of marketing strategy in fast food restaurants and consumer intentions of coming back to the restaurant. On the beginning of this correlation the researcher has been capable to explore the various features of the marketing strategy. Consequently, the researcher has coined his research as an exploratory research.

d) Data collection Method

Data collection method is an important stage of a research and must be well planned to ensure that researchers will not face the problem of being overwhelmed by the data, which become a barrier rather than an aid to the research project. In order to be able to plan and organize data collection systematically, an understanding of the various types of data depending on different approaches to, methods of, and techniques of data collection is significantly required.

According to Saunders et. al. (2003), data comes in various shapes and forms, but can be distinguished between two main categories: 1. Secondary data, and 2. Primary data.

i. *Secondary Data*

Secondary data is data which has been composed by agencies or individuals for purposes other than those of our meticulous research study. For example, if a management has carried out a review of, say, expenditures of family food, and then a food producer might apply this information in the organization's assessments of the whole probable market for a fresh product. Similarly, statistics arranged by a ministry on agricultural production will demonstrate useful to whole lots of people and organizations, including those marketing agricultural supplies.

The most frequent exercise of secondary data in marketing research is to achieve familiarity and to create a background in which primary data are composed, reported and analyzed, the problem is defined, and the research is planned. This approach is a literature search – an assessment of exiting material, penetrating for information pertinent to the present marketing research project. Materials are typically scholarly magazines, journals, books, newspapers, and company records (accessed through computer data bases). (Patzer L. Gordon, 1995)

Secondary data can give information about performance and procedures for conducting marketing research. For example, these data can help learn language for communication with the research sample members, questions and topics to avoid, problems likely to be encountered, and statistical techniques to engage. (Patzer L. Gordon, 1995)

Secondary data are potentially misleading term for people not experienced with marketing research. For example, it is misleading to think of secondary data as being of second importance, minor importance, inferior value, or in any way not necessary. Their worth, like that of all data, depends instead on the marketing research project. However, it is reasonable to conclude that secondary data play a significant role in almost all marketing research projects. Another misconception is to think of secondary data as coming second in a sequence. The sequential order is just the opposite: secondary data typically are collected and analyzed first, before primary data. (Patzer L. Gordon, 1995)

ii. *Secondary data sources*

Book reviews: The external research will be carried out through the reading and understanding of published material. This includes books and articles written on online shopping, catalogue shopping and consumer perception and satisfaction. Book and journal reviews are a very good source of collecting data as can get a wide variety of theories and authors references.

Internet Research: Internet research is another source of secondary data. This will be used to gather historical and present information about online shopping, catalogue shopping and consumer perception and satisfaction. This will also help to get contact details about the bottom level consumer as a whole. Helps to gather and analyses articles and journals about catalogue shopping and consumer perception and satisfaction. Collecting data from internet search is widely used now a days and is very quick and also you can get a wide variety of data through internet search.

Documents: Documents can be treated as a source of data in their own rights. In effect it can be an alternative to questionnaires, interviews or observation. This includes published materials of company details, like annual and financial reports of the proposed banks as well as other banks.

iii. *Primary Data*

Primary data means the data that are to be collected by the researchers themselves through a variety of data collection methods and techniques, for example, interviews, questionnaires, experiments, observations etc. Although the process of collecting primary data may have more requirements than secondary data in terms of time , effort and resources, the result is likely more relevant for answering the research question.

Regarding collecting data primarily, we can distinguish the type of data collected into two sub-categories; 1.quantitative data and 2. Qualitative data

Quantitative Data

Quantitative data means data which is number based or can expressed numerically as well as classified by some numerical value. In contrast, qualitative data means data which is in the form of

Global Journal of Computer Science and Technology (C) Volume XX Issue II Version I Year 2020

descriptive accounts of observation or classified by type. (Ghos B.N & Chopra P.K., 2003)

Quantitative data is more objective and scientific than qualitative data. It involves the implication that what is being researched can be quantified, and, therefore, is only applicable to incidence that can be quantified and measured.

Qualitative Data

Qualitative data explained items in terms of some feature or category that possibly informal or may use comparatively imprecise characteristics such as benevolence and flavor. However, qualitative data can contain well-defined aspects such as gender, nationality or object type.

Qualitative research apply individual in detail interviews, focus groups or questionnaires to gather examine and interpret information by observing what people do and say. It reports on the concepts, meanings, definitions, characteristics, symbols metaphors, and descriptions of things. It is more individual than quantitative research and is often investigative and open-ended. A little numbers of people are interviewed in detail or a relatively small numeral of focus groups is performed. Qualitative research engages the deliberate exercise and selection of a variety of practical materials, such as personal experiences, case study, introspection, interview, life story, artifacts, observational, historical, interactional, cultural text and productions, and visual texts that describe typical and controversial moments and meanings in individuals' lives. Saunders et. Al. (2003) suggest the distinctions between quantitative and qualitative data as shown in the Table 5 below

Table 5: Distinctions between quantitative and qualitative data (Source: Saunders et.al., 2003)

Quantitative Data	Qualitative Data
Based on meaning derived from numbers	Based on meanings expressed through words
Collection results in numerical and standardized data	Collection results in non-standardized data requiring classification into categories.
Analysis carry out throughout the use of statistics and diagram	Analysis conducted through the use of conceptualization

iv. *Primary Data Sources*

Interviews: Direct Interview is one of the major sources of primary data today. This method is would be used for the internal research. The internal research will focus on a few semis structured interviews with a few senior and top managers. The intention is to ascertain a true picture of the perceptions and satisfactions that a consumer feels when they eat in a fast food restaurant. These interviews will help to find out the secrets of their success or reasons for failure.

Interviews are a good source of collecting data. Also it is relatively cheap and quick to collect data through conducting interviews. But also there are some disadvantages in conducting an interview: -

1. As the nature of topic suggests it will be highly impossible to contact top level officials of the company and to ask them to give information about their company.
2. The second disadvantage is that the nature of the topic is so complex that there is a chance of getting biased opinion and it will be highly risky to rely on these answers.

Questionnaires: Another methodology that is the questionnaires. In this research, researcher uses seventeen relevant questions to find out the findings of this research which are given in APPENDIX 1. Questionnaires are more economical, easier to arrange, the answers will be standardized. In situations of difficulty to get appointments with the top-level managers this method would be used to. Postal questionnaires will be sendingto top managers of the banks and the responses can be analyses.

Collecting data from questionnaires is often for getting information and also it is relatively cheap. But it also has got some disadvantages like: -

1. Collecting data from questionnaires is a long procedure and takes long time to collect and analyze such data.
2. The second disadvantage is that people generally don't like to spend time in giving answers in writing.

e) Data Analysis

After the data have been composed, the researcher turns to the responsibility of analyzing them. Analysis of this data needs a number of closely connected operations such as creation of category, the importance of these categories to unprocessed data through tabulation, coding and then sketch arithmetical inferences. Scrutiny work after tabulation is mainly based on the calculation of various coefficients, percentages, etc. in brief the researcher can analyse the collected data with the assist of various numerical equipment.

f) Reliability

According to Joppe (2000)

"The extent to which results are consistent over time and an accurate representation of the total population under study is referred to as reliability and if the results of a study can be reproduced under a similar methodology, then the research instrument is considered to be reliable."

In this study, the researcher used the method of qualitative research in order to explore and understand the implementation of marketing strategies in the fast food restaurants, so that the researcher was then able to compare and contrast the findings with the literature, and eventually, was able to give suggestions about the issue. The case to be explored is dynamic and complex, and, therefore, it cannot be ensured that the research can be replicated and will give the consistent result when the time and circumstances have changed.

g) Validity

According to Winter (2000) "The traditional criteria for validity find their roots in a positivist tradition, and to an extent, positivism has been defined by a systematic theory of validity. Within the positivist terminology, validity resided amongst, and was the result and culmination of other empirical conceptions: universal laws, evidence, objectivity, truth, actuality, deduction, reason, fact and mathematical data to name just a few."

Researcher built the validity by establishing correct operational measures for the concepts of study. Researcher used the structured questionnaire as the mean to obtain the data.

The questions were designed and pre-tested in order to minimize as much as possible the misunderstanding and problems for the respondents; meanwhile it also increased the internal validity and reliability of the data.

h) Limitations of the research project

Researcher found some limitation at the time o research work. These are: a) Extent of research will provide a general overview of the entire outsourcing operations rather than complete audit. b) Limited amount of time available for completing the study. c) May not be possible to conduct interview with all of the firm's clients. d) Some of the data gathered may not be totally relevant to the research topic. e) Research needs to be conducted on a very low budget. f) There could also be a problem with translating the questionnaires and interviews as the company is located in a region where English not the main language in use. So there are chances that some data corruption might occur.

i) Methods for this Research

The chapter describes the methodologies used in the research. The project used both qualitative and quantitative methods. Using the following methods, a detailed study of the software development methodologies were carried out. The research is in two sections. Primary research carried out with a questionnaire. It consists of a survey. The Secondary research comprises of Literature survey from various sources.

i. Completing the Questionnaire

The questionnaire has been created in a way so that responders can answer quickly and easily. It is divided in to three main sections and contains all close-ended questions. The time taken to complete the questionnaire was approximately 15 minutes to 17 minutes.

Individual and organizational questions – This section contains questions on respondent's position in the industry as well as the position of the organization. It also contains question on the size of the organization including the number of employees, the projects they adopt and the likeness of adopting new technologies.

Methodology knowledge questions – This section focused on the knowledge of the respondent on the methodologies

Software development questions – This section contains questions on the different agile and traditional methodologies used on different projects. This is scaled on the project sizes measured in person months. The scales are selected as small scale, medium scale and large scale projects. There are questions to capture the opinion of the respondents on how effective the used methodologies were with regard to cost and quality of the software. Finally questions were included to capture their opinion on the preferred characteristics of both development methodologies from their point of view. The questionnaire is included in Appendix 1.

ii. Target Audience

The questions were distributed among software companies of various sizes and types. The respondents involved were mainly software architects, software engineers, and project managers. However, there were some other roles involved in software development as well.

j) Research Audit

Different resources were used for the research. The resources include various books on software engineering and development methodologies from different authors including Cockburn, journals related with software industry, white papers on agile and traditional methods, and websites from the internet which are related with the subject area.

IV. Research

The following are based on the data that were collected from various companies in the software industry. A questionnaire was prepared and provided in order to collect these data.

a) Data Collection

Most of the respondent was from Bangladeshi 21 different organizations. Among the organization 15 organizations were Information technology related organizations, 3 organizations were from

Telecommunication, 2 organizations were from the Engineering and 1 was other organization.

i. *Organizational Characteristics*

When analyze the results from the sample question it was discovered that about 70% of the respondents were from organizations with an Information Technology background. There were other respondents from telecommunication, engineering and medical organizations as well. There were some cases that projects were outsourced to information technology organizations. It is shown in the table below in a ratio of 100

Table 6: Survey Response from the organizations

Information technology	Telecommunication	Engineering	Others
70%	15%	10%	5%

Figure 9 represent the results

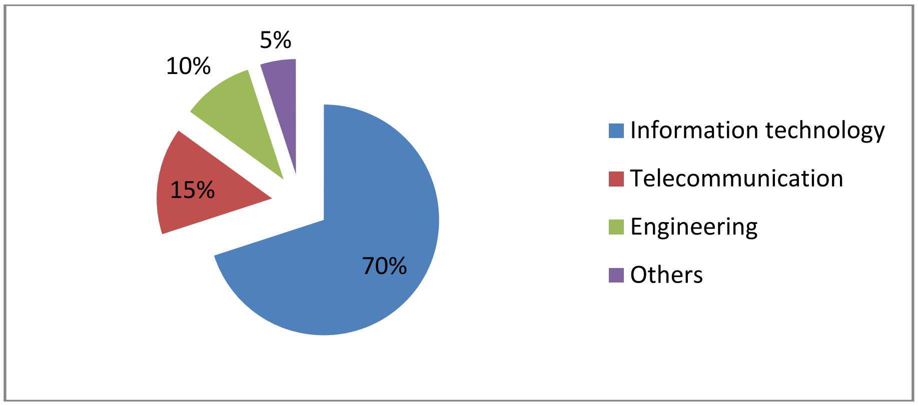

Figure 9: The type of organisations involved in the survey

ii. *Individual Knowledge Gathering from Different Organization*

Among the organization the number of the respondent was 21 and majority of the respondents to the questionnaire were software developers. The other respondent were System Analysts, software engineer and software architects. It is shown in table below in a ratio of 100.

Table 7: Survey response by job position

Developers	Analyst	Software engineer	Project manager	Executive
53%	13%	20%	7%	7%

Figure 10 represent these results.

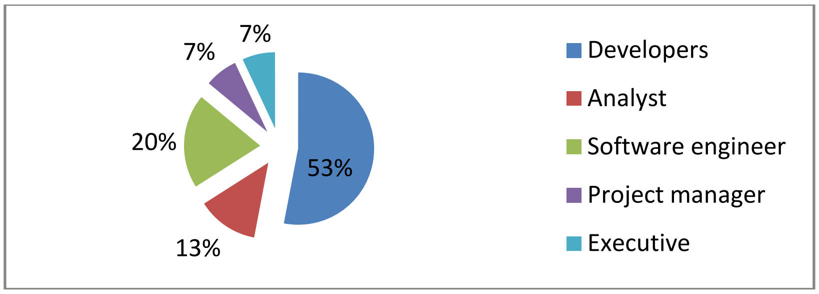

Figure 10: Respondents' job positions

iii. *Organization size based on employee*

The organizations were in different sizes of course. Around 73% of the organizations employed staff between 10 and 100.Most of them is employees who are working in Information Technology .The remaining 27% of the organizations fall under more than 200 employees or less than 10 employees working for the organization. It is shown in the table below in a ratio of 100.

Global Journal of Computer Science and Technology (C) Volume XX Issue II Version I Year 2020

Table 8: Employees' in software development in organizations

Less than 10	Between 10 and 20	Between 21 and 50	Between 51 and 100	More than 100
7%	20%	26%	27%	20%

Figure 11 represent these results

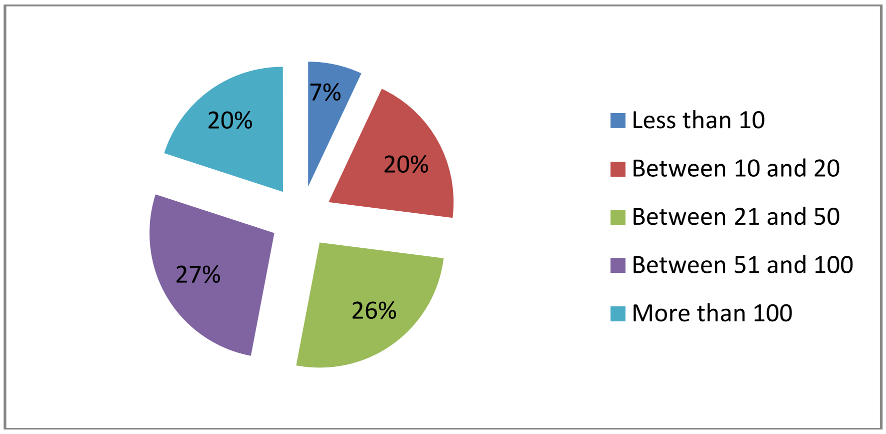

Figure 11: Employees' in software development in organizations

iv. *Agile and Traditional Software methodology Knowledge of the respondents*

When it comes to the knowledge rating for different methodologies more than 90% of the respondents have an understanding about agile and traditional methodologies in an average or higher level. 12 out of the 15 respondents have rated their knowledge of agile methodologies as average or broader, out of that 6 of the respondents rated their knowledge as broad or very broad. For traditional methodologies the rating was broad or very broad for 12 respondents. When compare the experience they have in the software industry it was revealed that with less experience in the field or in other words people who have experience less than four years have less practical knowledge in traditional methodologies. Figure 12 presents the results below.

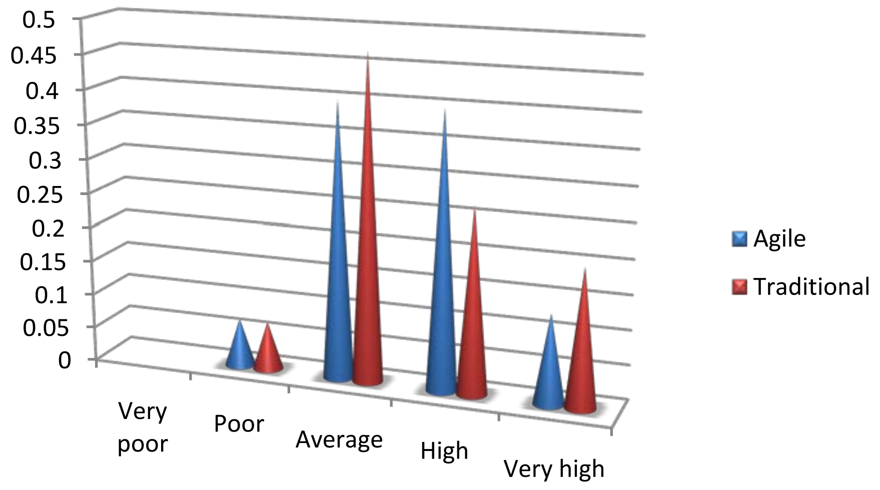

Figure 12: The methodology knowledge of the respondents'

v. *Knowledge of adopting new methodology by the respondents*

According to the respondents, the result found on adopting technologies in different organizations was interesting. More than 75% of the organizations were either Leaders in adopting a methodology or followers. But there are other organizations who describe themselves as conservatives. This means that there are organizations which will hang onto their accepted methods and not willing to experiment something new. It is illustrates in figure 13.

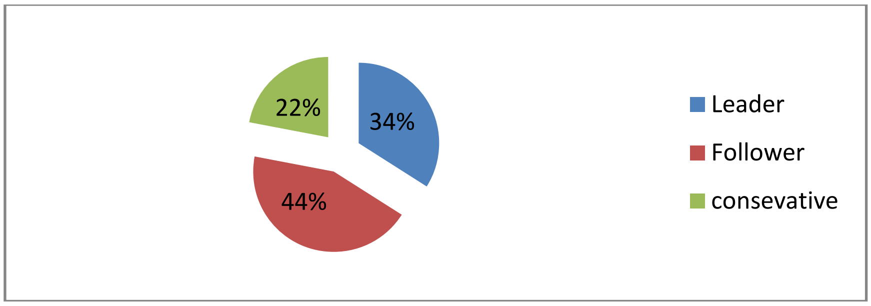

Figure 13: Adopting new methodologies in the industry

According to a survey results published by Ambler in early 2006 he has found that even though more than 60% were fully or partially using agile, there is a considerably a large number of organizations who are still have no idea of adopting agile. In that survey result shown below

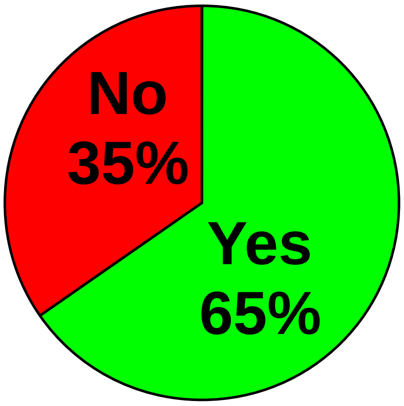

Figure 14: Ambler survey result of Adopting agile methodology (www. ambysoft.com)

In this survey about 75 % of the respondents are either leader or follower to adopt agile methodologies, on the other hand in Ambler survey 65% of the respondent said yes for adopting agile methodology.

So with the result of this survey, it proves that other than the organizations who adopted agile at the beginning, the potential of organizations adopting agile at a later stage without any assurance is minimal. A possible reason could be that these organizations are just waiting to see how agile projects will result in the future.

b) Methodologies used in Organizations

In this section focus on which type of agile and traditional methodology used on different organizations. If the organization is small which types of agile and traditional development methodologies they are using. In similar way it is focused for the medium and large scale organizations.

i. *Use of Agile Methodologies*

According to the respondents of the different types of organization indicate that Extreme programming (XP) is the most popular method used in the industry. But SCRUM also maintains a good position within the industry even though it is not up to XP level. There was a remark about XP stating that it sometimes gives bit of a fear because of the steps it includes and also the "Name itself". There is an interesting point that was found during the analysis. The next most popular was in-house build methods by organizations for their own use. In an article published by Sliwa (2002) mentions that agile methods can be mixed for different organizations purposes. The article further stated that;

"Schwaber, a Scrum co-creator, said it makes sense to combine Scrum and XP because Scrum focuses on management practices and XP centers on engineering practices for building object-oriented software."

The result proves this point as organizations are already using combined methods according to their needs for a better result. Another point was that some organizations tend to mix other new techniques built for specific tasks in software development with their development methodology. For example they use scheduling techniques such as planning porker for estimating time for development tasks. Planning poker is a technique which is used in Scrum in most cases to estimate time for development tasks. It has a deck of cards with different estimates which the developers can use (Cohn, 2005) cited by planningpoker). Figure 14 represent the results of respondent.

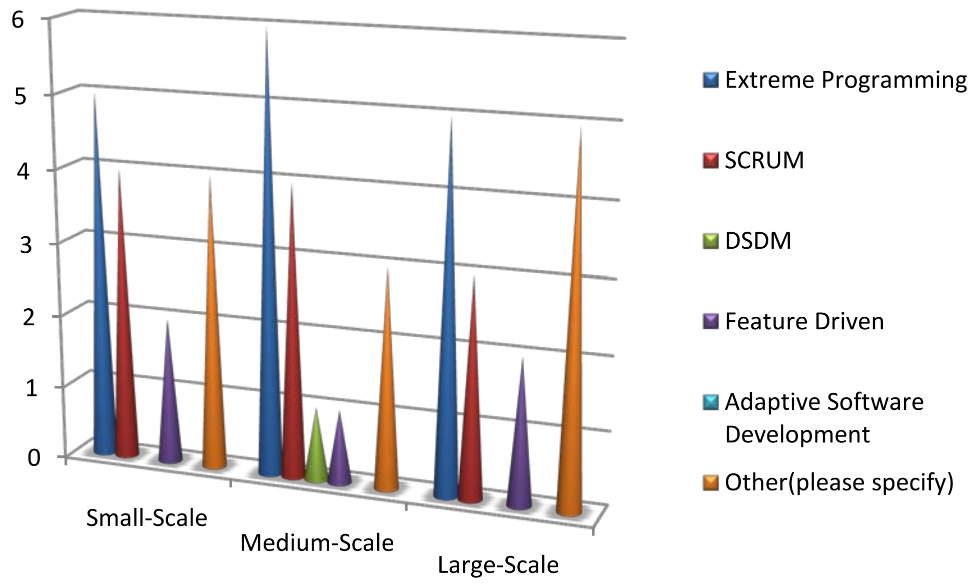

Figure 15: The use of agile methodologies

In figure 15 it is indicate that for small, medium and large scale organizations Extreme programming (XP) is most popular among the agile methodologies and SCRUM is in second position. Other methodologies are using in a very small scale in different organizations.

ii. *Use of Traditional Software development Methodologies*

According to the respondents of the questionnaire, for traditional methodologies more than 50% respondents use the waterfall model, 23% of respondents were interested in unified process and the rest was on in-house build methods for different type and sized projects. Figure 16 represent the results.

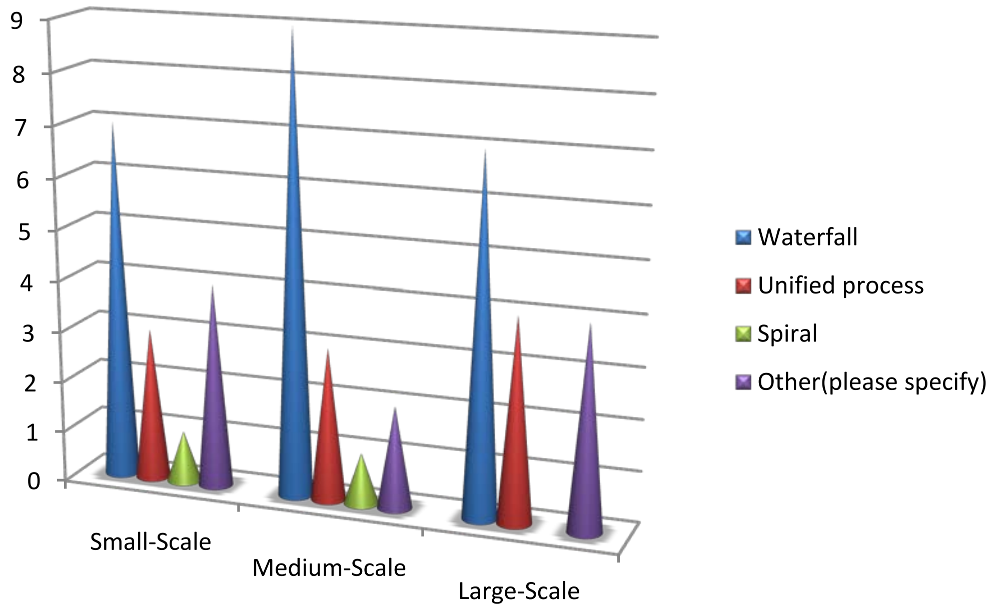

Figure 16: The use of traditional methodologies

V. Analysis

In this chapter the discussion will be focused on analyzing these collected data and find out the responses from the software industry professionals.

a) *Most Appealing Agile Values over Traditional Characteristics*

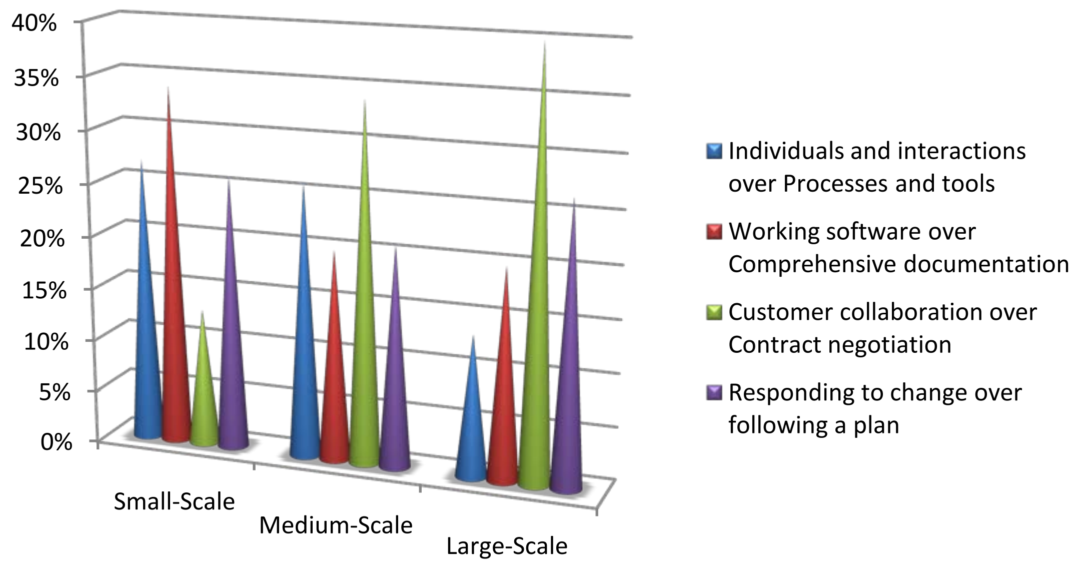

Figure 17: Most appealing agile values

When consider the small scale software projects more than 30% of the respondents think that working software is more important. People interactions and responding to change come after respectively. All the respondents' believe that human interaction is an important fact for better software development regardless of the project size. Cockburn (2001) point out that;

"Core to agile software development is the use of light-but sufficient rules of project behaviour and the use of human and communication-oriented rules" proving the point made out from the survey results.

Respondents' have a different view about Medium and large projects. For both of these project types, customer collaboration have obtain the highest votes. This means that when the system is getting bigger more customer collaboration helps to keep the development on the track. Medium projects have a higher percentage of votes for people interaction than large projects. Even though this is outside the expected result for large projects, it may be due to the reasons that respondents think it is hard to communicate within large projects. Figure 17represents the results obtained.

b) *Factors that Influence to use Agile Methods over Traditional Methods*

Cost and quality of software products are the main concerns in the industry when it comes to software engineering. It is important for both software organization as well as the customers (Krasner, 1998). Due to this in the questionnaire, it was necessary to include questions regarding the cost and software quality. The reason was to find out how agile methodologies have affected on these two features of a software project compared with traditional methodologies.

The questions were targeted to capture the opinions of the respondents, whether they believe by adopting agile methodologies will affect the software cost and quality of a project than the traditional methodologies. Since agile is making a huge entrance to software industry I was expecting a very higher positive feedback. Even though the result was rather different from what I was expecting.

When it comes to cost of the software project 50% of the respondents agreed that there were no change in cost at all by using agile methodologies but according to Ambler (2007) it was 47 % (in Figure 19). Surprisingly 22% of the respondents have voted as the affect of the cost has slightly decreased than the traditional methodologies but according to Ambler (2007). Only about 18% of the respondents believe that agile methodologies have made a slight increase affect on cost. The rest of the respondents falls both sides to the far end of the ratings.

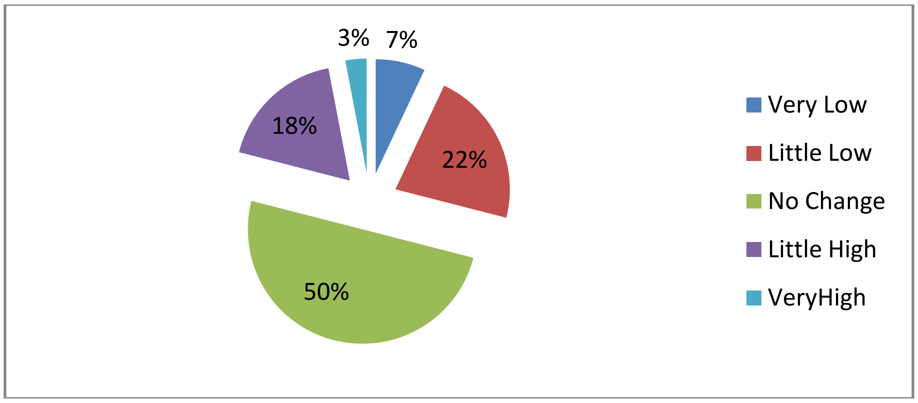

Figure 18: Affect on software cost from agile methodologies

But with regard to software quality the result I obtain was different than the cost. Overly respondents have a positive feedback on the quality. More than 30% of the respondents believe that adopting agile methodologies have slightly increased the affect on quality compared with traditional methodologies. 13% of the votes were even higher. They believed that the affect was in a very higher state. But again there were huge number respondents who really did not believe in agile methodologies as 39% was on the no change state. The rest was in the low side of the rating.

Figure 18 and figure 19 show the results for the software cost and quality I discovered.

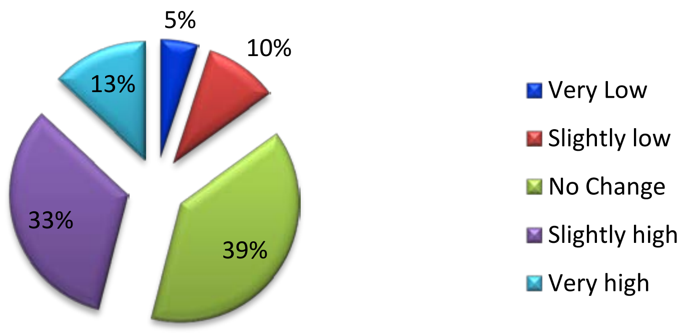

Figure 19: Affect on software Quality from agile methodologies

So comparatively organizations believe that there is a higher effect to quality from agile than the effect to cost. In the survey done by Ambler in early 2008 the results on quality was noticeably different. In his survey 67% of the votes said that they experienced better or significantly better affect on the quality of software projects with the adaptation of agile methodologies.

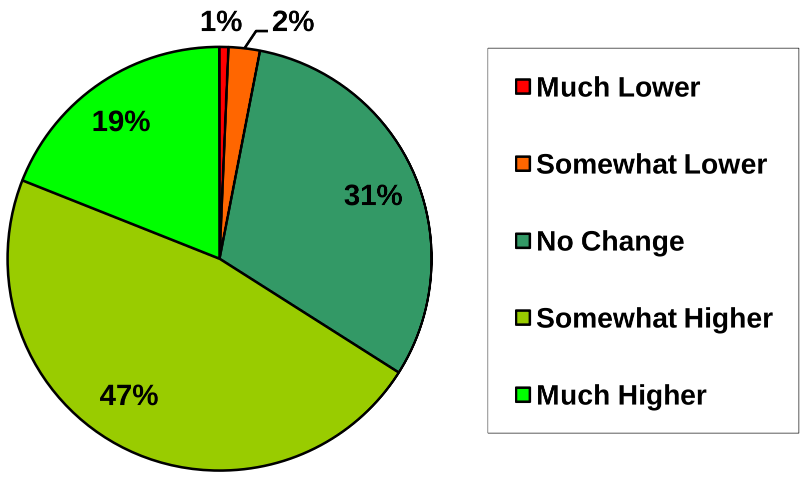

Figure 20: According to Ambler affect on software quality from agile methodologies

The difference to my results is that considerably a large number of respondents voted for no change. Since the results I got mentioned about medium size projects than other two there can be issues occur when practicing agile values such as team communication and customer feedback. Due to these reasons there may be problems when try to capture the quality of the project. The cost affect was slightly tally with the results from the Ambler's survey. Figure 18 and figure 19 show the results for the software cost and quality I discovered.

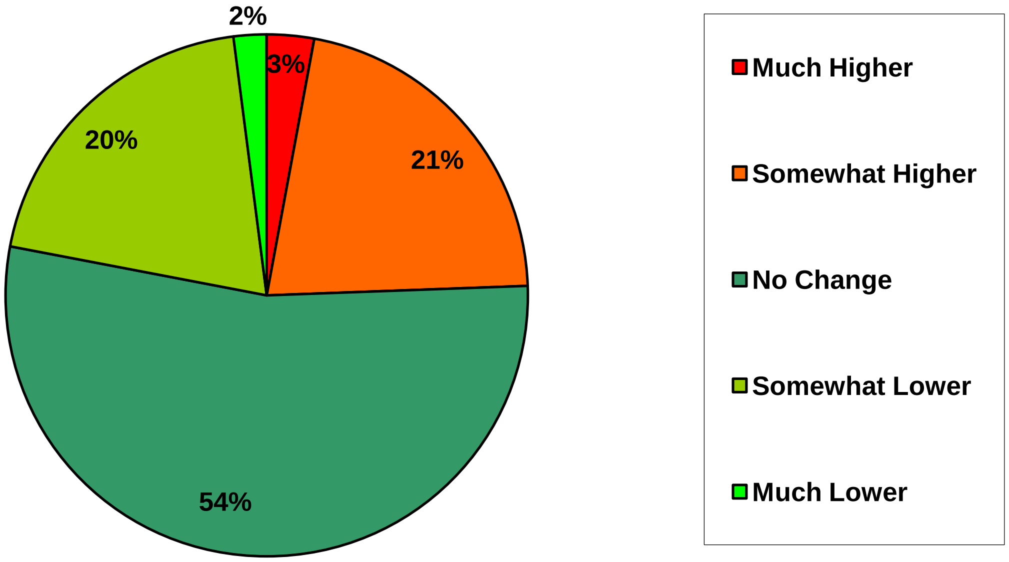

Figure 21: According to Ambler affect on software cost from agile methodologies

i. *Comparison of software cost from agile methodologies between Ambler and this survey*

Table 9: Comparison of software cost from agile methodologies between Ambler and this survey

Cost	This surver	Ambler (2007) survey
Very low	7%	2%
Slight low	22%	20%
No change	50%	54%
Slight High	18%	21%
Very high	3%	3%

This two survey result on software cost are closely similar and their correlation coefficient is 0.991039.

ii. *Comparison of software quality from agile methodologies between Ambler and this survey*

Table 10: Comparison of software quality from agile methodologies between Ambler and this survey

Quality	This Survey	Ambler (2007) survey
Very low	5%	1%
Slight low	10%	2%
No change	39%	31%
Slight High	33%	47%
Very High	13%	19%

This two survey result on software quality are closely similar and their correlation coefficient is 0.87579.

c) *Preferences for Agile and Traditional Methodologies*

When an organization uses a methodology, there are processes and techniques they have to follow regardless of the type of the methodology. From the past experiences in the industry I had the understanding that there were some processes which development teams think is useless for the success of the project objectives. To have a broader view in these aspects questions were included in the questionnaire to find out respondents opinion on certain characteristics in both methodologies.

According to the results shown in figure 22 more than 50% of the respondents' believe that low management control is a drawback for small scale and medium scale projects. In fact they believed that low management affects all sizes of projects in a considerable amount. The other major aspect was the project structure. Again all the respondents' believed

Global Journal of Computer Science and Technology (C) Volume XX Issue II Version I Year 2020

that lack of project structure affects all sizes of projects. By looking at the figure 21 it is possible to come to an understanding that large projects do not adopt agile methodologies because of this factor.

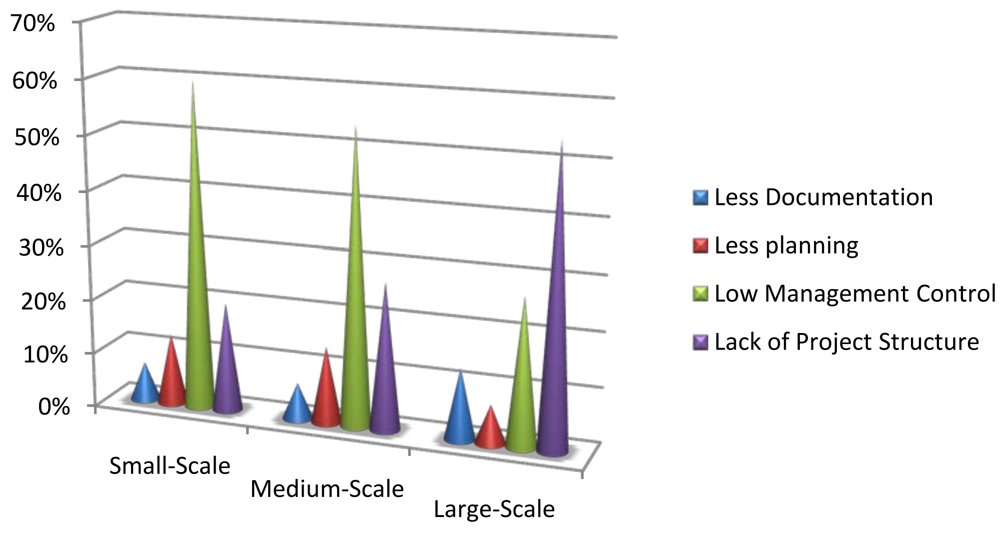

Figure 22: Low preferences of agile characteristics

Traditional methodologies always had the drawback on documentation. The results shown in figure 23 clearly indicate the respondents' opinion on the heavy documentation for all types of projects. Especially when it comes to small scale projects nearly 60%.

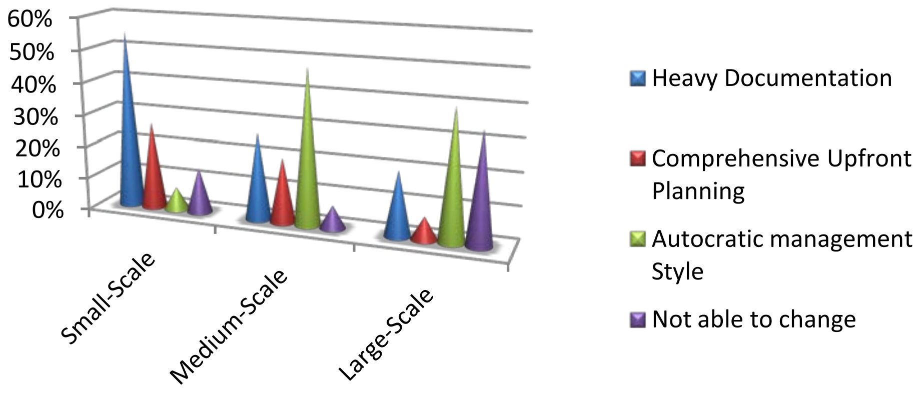

Figure 23: Low preferences of traditional characteristics

Agree that heavy documentation is a waste. Lindvall et al. (2002) clarify this in a survey paper by stating;

"Documentation should be assigned a cost and its extent be determined by the customer.

Many organizations demand more than is needed. The goal should be to communicate effectively and documentation should be the last option."

d) Methodology preferred

When analyzing data to find out which development methodology is preferred by the respondents I have realized that agile has come a long way during the past few years after it was properly published. But on the other hand it still has to go further to take over the whole software market.

i. *Methodology selection for different project sizes*

The results discover that almost all the respondents have agreed that agile methodologies are the best for small scale projects. This means that software organizations getting to know how to get their hands on agile methodologies to manage the tasks in small scale project environments. For medium scale projects both methodologies were voted. The gap between the results for the two methodologies was very less. This shows that agile is adopted by organizations than before for medium scale projects. But respondents had a different idea about large scale projects. Nearly

90% of the responds were bias to traditional methodologies. Only the remaining was for the agile and other methodologies.

The interesting fact was that organizations are using a mix of both methodologies when it comes to medium and large scale projects. Medium size projects are in this process more than the large size projects but it seems within the next few years large scale projects may also start to use a mix of both methodologies. Figure 24 below represents the methodology selection.

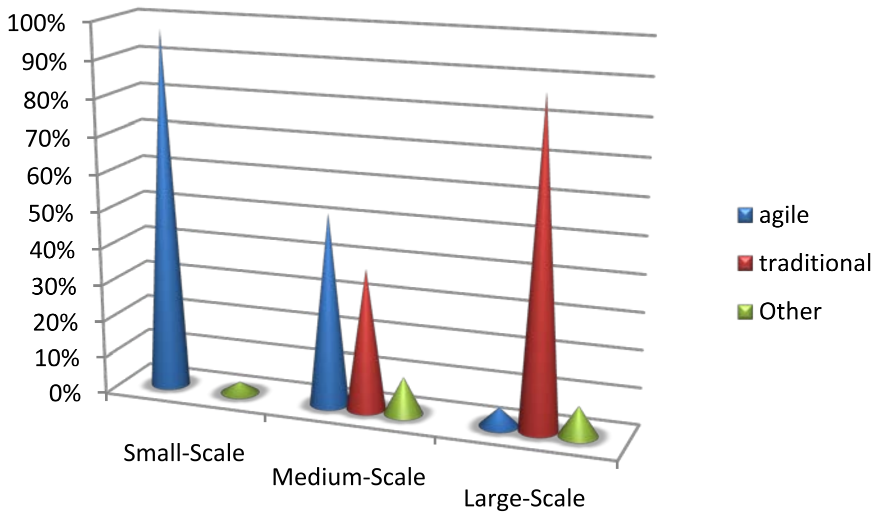

Figure 24: Methodology selection for different project sizes

ii. *Ratings for other mix techniques into development process*

The other fact was that some organizations mix other techniques also into their methodologies. Some respondents have rated for Scrum or Scrums and also planning poker which are new techniques to make the development processes more efficient. Figure 25 represents the total results.

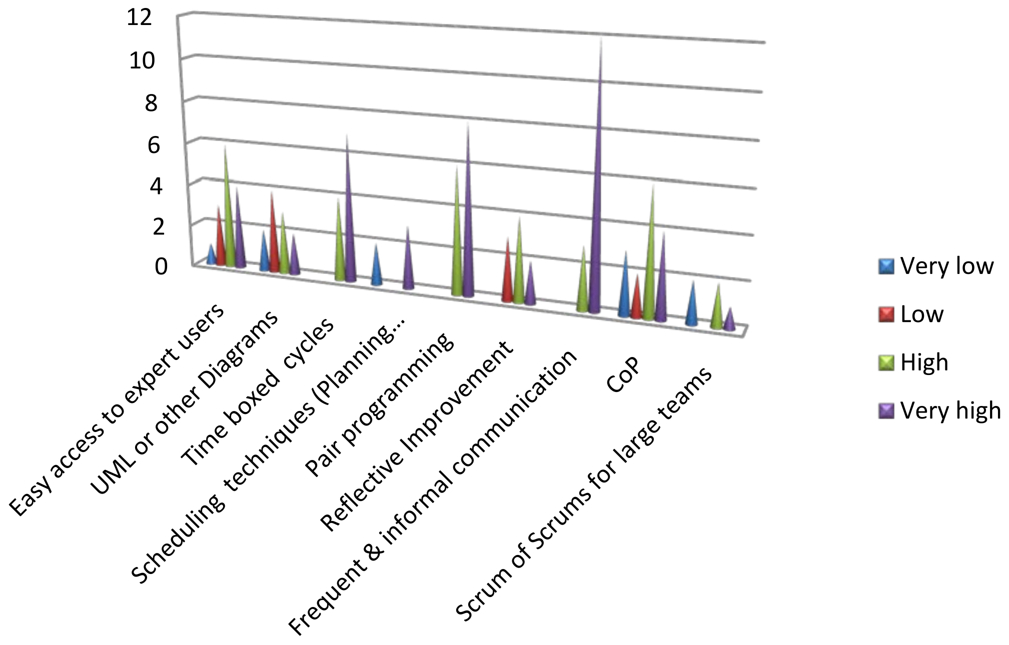

Figure 25: Ratings for other techniques

iii. *Use of agile methodologies*

Around 40% to 50% of the respondents' agree that for medium and large scale projects autocratic management is not necessary. This type of a management would keep the teams stick to the standard work and have no agility leaving the teams work without any innovation or creativity.

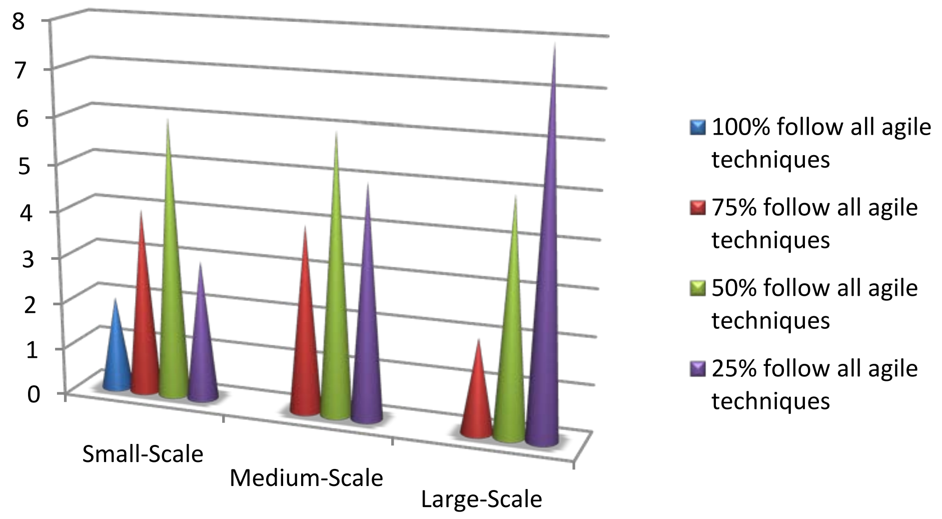

Figure 26: Use of agile methodologies

Figure 26 represents the extent which agile methodologies are used for different types of projects sizes. Only for small scale projects some organizations use 100% of the agile methodologies. Other than that for both small and medium scale projects majority of the respondents' agree only up to 50% of agile methodologies are used. There was a comment from a respondent saying that "It is hard to stick to agile methods especially when it comes to large projects. There are other techniques been used mixing with the practices in both agile and traditional methods?" This means that organizations are tend to use their own in house methodologies created to suits the projects they handle.

The findings of this project research study also confirm the appropriateness of the use of agile methodologies for small scale projects, traditional and agile methodologies for medium scale projects and traditional methodologies for large scale projects of an organization.

VI. Conclusion

The purpose of this research is to present a set of guide lines for a software organization to help choose the most appropriate development methodology according to most of the software projects they have in hand. The thesis starts with an overview of the software industry and explanation of the problem domain which is focused in the research. Through traditional and agile development methodologies, this discusses the different software development approaches used in the software industry. Further a discussion about the life cycles of selected approaches from both traditional and agile methodologies were carried out with identifying the roles, responsibilities and practices of each development approach. This would give the reader clear idea about the two methodologies and also the differences they have. Chapter 2 briefly presents a comparison on the methodologies and focuses on the problems in both methodologies. Finally, in order to get the professional opinions, the document presents the analyzed results from the survey conducted.

Throughout the research it was understood that the traditional methodologies were apparently handling a considerable portion in software industry. The basis was the complete planning, heavy documentation and extensive designs. Traditional approaches will still be useful in large, long lived projects that require special safety, reliability or security requirements. The military and defense industry gives a perfect example to prove this point. Lijek (2007) in a presentation discusses the reasons why agile methodologies are not adopted in the military and defense industry.

- Defense Contractor Mentality regarding change
- Safety Critical Systems
- Long development cycles
- Large teams
- Customer Relations

But in the near future with the improvements agile will be able to be adopted in these industries.

Agile methodologies cannot be defined by a small set of rules and practices. From the literature review and the survey results it became obvious that agile methods have the capability to respond to change faster, the ability to extract the hidden creativity and innovations out of the teams, the capability in balancing the structure and flexibility and to drive the organization through rough situations and uncertainty. Agile is more likely to dominate volatile environments with uncertainty and unpredictability where the exact customer needs are not clear. Organizations tend to respond to the market changes quickly with the customer needs. They make plans for the system but do not tie their view to it. Rather than making models they want to focus more on the working software. They focus on constant interaction within the team members, customers and management and individual skills as well. With all the readings and findings it is clear that there is no "one-size-fits-all" solution.

a) Limitations

There were some obstacles on the way to the success of this thesis. At first, Gathering the information from the professionals and practitioners in the industry was a problem as it takes long time for most of them to respond to the questionnaire. There were some returned questionnaires half-completed which had to be discarded. Another barrier was the time factor. Even though there are lots of areas that can be focused under this topic it was not possible since the allocated time was limited. But within the time period a good and original piece of work was produced with great attention.

b) The Guidelines

The guide lines presented are to support an organization to select the most appropriate software development methodology for software projects they undertake. For an organization, it is hard to have more than single software methodologies operating. Generally the top management and human resources would prefer all projects to use the same method for ease of handling.

However, software developing is a complex and uncertain process. To cater for specific needs, Project requirements and different teams may have to produce different results. Therefore, it is important to consider adopting different methodologies or a mix and match of several techniques from different methodologies at least between two departments or two different project sections which operate independently in the same organization.

The following guidelines are created with the knowledge obtained from the research on the literature and the analysis and understanding gained from the survey results which involved the real software development organizations.

- Flexibility – Everybody involved with software development needs to be flexible, starting from the top management. They should understand different projects have different needs and there are different ways to make them successful.
- Priority on the needs – Different projects need Different techniques and artifacts. Therefore, it is important to identify them and prioritize them. For an example the use of other techniques and artifacts outside the working methodology (e.g. planning poker) for certain types of projects may lead the project to greater success. But the management has to remember that, this may need some training to the team members as a person may have to deal with a range of methods and/or artifacts.
- Cater according to the team – For different projects, Development teams may be different in size. So it is important to use suitable methodology or mix of methodology to cater for that requirement. As an example, XP and scrum are suitable for projects with small-scale to medium-scale development teams with 4 to 20 members. However, for large and medium scale teams Unified Process can be used.
- Define targets – There are specified artifacts for each approach in traditional development. So organizations rely on these artifacts and always try to stick to them. Rather defining the targets with the help of the customers on what to build may be more productive. The artifacts will be decided along with the targets which is more useful for all the parties involved in that specific project.
- The use of methods – Organizations with large or medium scale projects can combine subsets of different methods. SCRUM is a methodology which can be mixed with different other methodologies including XP and waterfall. However, for organizations, who handle small scale projects can settle with a single methodology.

Come up with a specific set of rules is not that easy in a rapidly changing field with uncertainty like software engineering. For different organizations, these guidelines can be used in different ways. With time and experience these can be improved more. The best way is to experiment these in a real time environment and observe the validity and the success, which will give an understanding on how to improve them for better results.

Acknowledgement

I would like to acknowledge and state my total pleasure to those whose efforts and support in one way or the other contributed to the successful completion of this research study. I particularly wish to convey profound gratitude to my supervisor, Dr. Nigel Kermode, for his invaluable support, encouragement, useful suggestions and supervision throughout this research work. I also wish to thank my friend Mr. Khondker Razeeb-us-Saleheen, my class mates and my brother, Mr. Channa punchihewa for his support during my study. I am as ever, especially indebted to my parents, Mr. and Mrs. Rahman for their love and support throughout my life. Moreover, to my Almighty who made all the things possible.

References Références Referencias

1. Agile Alliance.2002. Agile Mainfesto http://www.agilealliance.org/
2. Abrahamson et al,P. 2002. Agile software development methods – review and analysis. VTTPublications 478. http://www.inf.vtt.fi/pdf/publications/2002/P478.pdf.

3. Fowler, M and Highsmith, J. 2001. The Agile Manifesto. Software Development Magazine. August.http://www.sdmagazine.com/documents/s=844/sdm0108a/0108a.htm.
4. Ambler, S. 2008. Agile Adoption Survey. Web site accessed at http://www.ambysoft.com/surveys/agileFebruary2008.html
5. Heckman, S. and Williams, L. A Model Building Process for Identifying Actionable Static Analysis Alerts , 2nd IEEE International Conference on Software Testing, Verification, and Validation, Denver, CO, USA, to appear.
6. Beck, K. 2000. Extreme Programming Explained: Embrace Change. Reading, Addison-Wesley.
7. Beck, K. 1999. Embracing change with Extreme Programming. IEEE Computer, 13, 10. Pp 70-77.
8. Beck, K. 2005. Extreme Programming Explained: Embrace Change, Second ed. Reading, Addison-Wesley.
9. Booch, G. 1994. Object Oriented Analysis and Design with Applications, 2nd edition. Addison Wesley Longman.
10. Boehm, B. W. 1988. A Spiral Model of Software Development and Enhancement. Computer, IEEE software.
11. Boehm, B. and Turner, R. 2003. People Factors in Software Management: Lessons From Comparing Agile and Plan-Driven Methods. CROSSTALK; the Journal of Defense Software Engineering.
12. Highsmith, J. and Cockburn, A. (2001). Agile Software Development: the Business of Innovation. IEEEComputer, 34 (9), 120-122.
13. Schwaber, K and A. Beedle (2002). Agile Software Development with SCRUM. Prentice-Hall, UpperSaddle River, NJ.
14. DSDM Consoritum. (2008). Dynamic Systems Development Method, version 3, Ashford, Eng.Ehn, P. (1989). Work-oriented design of computer artifacts, Lawrence Erlbaum, Hillsdale, NJ.
15. McCauley, R. (2001). Agile Development Methods, Poised to Upset Status Quo, SIGCSE Bullentin 33(4):14-15.
16. Miller,G.G.(2001),The Characteristics of Agile Software process, The 39th International Conference of Object Oriented Languages and Systems.
17. Larman, C. (2004), Agile and iterative development: a manager's guide, Addison-Wesley.
18. Dekkers, R. (2005), International Journal of Networking and Virtual Organizations, Volume 3: 1-24.
19. Langr, J. (2006).Agile Java™: Crafting Code with Test-Driven Development.
20. Rakitin, S. (2001). Manifesto Elicits Cynicism. IEEE Computer, 34(12): 4.
21. Rakitin, S. (2005) Agile Methods - Beyond the Hype. Food for Thought newsletter from Software Quality Consulting. July/August 2005, 2 (7). http://www.swqual.com/newsletter/vol2/no7/vol2no7.html#article, last visited 04/05/2007.
23. Cockburn, A. 2008. A Methodology Per Project. Website accessed at http://alistair.cockburn.us/Methodology+per+project
24. Cockburn, A. 2001. Agile software development. Addison-Wesley, Boston, MA.
25. Coffin, R. and Lane, D. 2007. A Practical Guide to Seven Agile Methodologies. Website accessed at http://www.devx.com/architect/Article/32836/1954
26. Cohn M. 2005. Cited by Planning Poker in Detail. Website accessed at http://www.planningpoker.com/detail.html
27. Moonzoo Kim. 2007. Agile development
28. Nandhakumar, J. and Avison, D. E.1999. The Fiction of Methodological Development: A Field Study of Information Systems Development, Information Technology & People, 12(2), Pages 176–191
29. Highsmith, Cockburn, A. 2001. Agile software development. Addison-Wesley, Boston, MA.
30. Renhui, L. and Fengyong, Z. 2007. Multi-Attribute Evaluation for Software Project Risk. IEEE Xplore. Pages 4942-4945.
31. Robinson, S. 2002. Consider DSDM as an XP alternative. Web site accessed at http://articles.techrepublic.com.com/5100-10878_11-1049982.html
32. Schwaber, K. 2007. What is Scrum. Website accessed at http://www.scrumalliance.org/resources/227.
33. Sliwa, C. 2002. XP, Scrum join forces. Website accessed at http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=69183
34. Takeuchi, H. and Nonaka, I. 1986. The New New Product Development Game. Comptuer. Harvard Business Review.
35. Vaihansky, P., Sutherland, J. and Victorov, A. 2006. Hyperproductivity In Large Projects Though Distributed Scrum. Agile journal. Website accessed at http://www.offshoreagile.com/news/?doc=205
36. Devdaily. The rational unified process. Access at http://www.devdaily.com/java/java_oo/node19.shtml
37. Paulk, M. C. 2001. Extreme programming from a CMM perspective. Software, 18, 6. Pages 19-26.
38. Davis, A. and SITARAM, P. 1994. A concurrent process model of software development. ACM sigsoft. Software Engineering Notes vol 19 no 2
39. Kalermo, J. and Rissanen, J. 2002. Agile software development in theory and practice.
40. Georgiadou E. 2003. Software Process and Product Improvement, A Historical Perspective. International Journal of Cybernetics. Volume1, No1. Pages 172-197
41. Ghosh, B.N. and Chopra, P.K.(2003)A Dictionary of Research Method, Leeds, UK.

42. Khan, A. and Balbo, S. 2004. A Tale of two Methodologies: Heavyweight versus Agile. Website accessed at http://ausweb.scu.edu.au/aw04/papers/edited/balbo/
43. Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L.,and Zelkowitz1, M. 2002. Empirical Findings in Agile Methods, D. Wells and L. Williams (Eds.): XP/Agile Universe. Pages 197–207.
44. Krasner, H., 1998. Using the Cost of Quality Approach for Software. Computer. CROSSTALK; the Journal of Defence Software Engineering.
45. Lijek, S., 2007. Software life cycle models in the defense industry. Website accessed at greenbay.usc.edu/csci577/spring2007/site/students/presentations/Lijek.ppt
46. Parekh, N. 2005a. The waterfall model explained. Website accessed at http://www.buzzle.com/editorials/1-5-2005-63768.asp
47. Parekh, N. 2005b. Spiral Model - A New Approach towards Software Development. Website accessed at http://www.buzzle.com/editorials/1-13-2005-64082.asp
48. Peters, J. F. and Pedrycz, W. 2000. Software Engineering: An Engineering Approach. John Wiley & Sons.
49. Royce, W. W. Managing the Development of Large Software Systems. IEEE Western Conference (Wescon). Pages 1-9.
50. Williams, L. 2003. "The XP Programmer: The Few Minutes Programmer," IEEE Software, vol. 20, no. 3.
51. Avison D.E. and Fitzgerald, G. 1995. Information Systems Development: Methodologies, Techniques and Tools. McGraw-Hill.
52. Charette, R. 2001. The decision is in: Agile versus heavy methodologies. Cutter consortium e- Project management Advisory service, 2, 19.
53. Clifton, M. and Dunlap, J. (2003a). What is DSDM. Website accessed at http://www.codeproject.com
54. Clifton, M. and Dunlap, J. (2003b). What is SCRUM. Website accessed at http://www.codeproject.com
55. Saunders, M., Lewis, P. and Thornhill, A. (2003). 3rd. Research Methods for Business Students, Prentice Hall.
56. Maylor, H., & Blackmon, K. (2005). Researching Business and Management. Hampshire and
57. New York: Palgrave MacMillan.
58. McNeil, P. and Chapman, S.(1985).Research Methods.
59. Kerlinger,F. and Lee,H. Foundations of Behavioral Research (New York: International Thomson Publishing, 2000).
60. Patzer, Gordon L. (1996): Experiment-Research Methodology in Marketing. Types and Applications, Quorum Books: Westport.

Appendix 1: Questionnaire

The objective of this survey is to find out various methods been used in the software industry for software development. The data collected will be strictly confidential and will only be used for this academic research. Please share your views about your experiences and your personal opinions. If you require a summary of the findings please complete the optional section at the end of this questionnaire.

The questionnaire is divided in to three sections. The questions contained are all close end questions. But if you have any comments for any of the questions please include them with the questions.

For any questions or clarifications please contact me,

A.K.M Zahidul Islam
akmzahidulislam102@gmail.com

1. What is your job Title?

Programmer / Developer	
Analyst	
Software Architect	
Software Engineering	
Project Manager	
Executive	

Other (Please Specify):________________________________

2. How long have you been working in the software industry?

< 1	
1-4 Years	
5-7 Years	
8-11 Years	
> 11 Years	

3. How would you best describe your organization type?

Information Technology	
Telecommunications	
Engineering	
Medical	
Education	
Government	
Other	
Information Technology	

Other (Please Specify):________________________________

4. How many employees are there in your organization engaged in software development/ maintenance?

<10	
10 – 20	
21 – 50	
51- 100	
101-200	
>200	

5. How would your organization react in adopting new technologies/ methodologies?

Leader (Look forward to adopting new technology as it release)	
Follower (Adopt the technology after the leader)	
Conservative (Wait till it is proven to follow)	
Static (Do not adapt new technologies)	

Section 2: Methodology knowledge

6. Which of the following more appropriate to rate your knowledge in Agile Methodologies?
(Agile methods: Extreme programming, SCRUM, DSDM, etc…)

Very high	High	Average	Poor	Very Poor

7. Which of the following more appropriate to rate your knowledge in Traditional Methodologies?
(Traditional methods: Waterfall, Spiral, Unified Process, etc…)

Very high	High	Average	Poor	Very Poor

Section 3: Software development

8. Which of the following best describe the last project you were involved?

Small scale project (3 to 7 people)	
Medium scale project (5 to 20 people)	
Large scale project (20+ people)	

9. Consider the last 5 projects undertaken at your organization; provide Yes (Y) or No (N) to the following.

	Proj1	Proj2	Proj3	Proj4	Proj5
Was it delivered on time?					
Was it delivered within budget?					
Did it satisfy the user's requirements?					
Did it require rework?					
Was it delivered on time?					

Comments (if any):

____

____

The following questions are based on the project sizes mentioned in question 8. Select the appropriate selections with a tick or cross (X).

10. Which Agile methodologies you prefer for each type of software development project?

	Small scale	Medium scale	Large scale
Extreme Programming			
SCRUM			
DSDM			
Feature Driven			
Adaptive Software Development			
Other(please specify)			

11. Which Traditional methodologies you prefer for each type of software development project?

	Small scale	Medium scale	Large scale
Waterfall			
Unified process			
Spiral			
Other(please specify)			

12. What is the average size of teams you use for each size of development projects?

	Small scale	Medium scale	Large scale
2 - 15 members			
16 – 50 members			
51 – 200 members			
More than 200			

13. If you prefer to use any of the following techniques outside specific software development methodology use how would you rate them? (Rate only the preferred else leave blank).

	1	2	3	4
Easy access to expert users				
UML or other Diagrams (use cases, ERDs)				
Time boxed development cycles				
Scheduling with techniques like Planning porker				
Pair programming				
Reflective Improvement				
Frequent & informal communication				
Communities of practice				
Use scrums of scrums for large teams				

14. Compared with traditional methodologies which of the following agile values most appealing to you for the different software development project sizes?

	Small scale	Medium scale	Large scale
Individuals and interactions *over* Processes and tools?			
Working software *over* Comprehensive documentation?			
Customer collaboration *over* Contract negotiation?			
Responding to change *over* following a plan?			

15. Which of the following agile characteristics would you think is not suitable for the three sizes of software projects?

	Small scale	Medium scale	Large scale
Less Documentation			

Less planning			
Low Management Control			
Lack of Project Structure			

16. Which of the following Traditional characteristics would you think is not suitable for the three sizes of software projects?

	Small scale	Medium scale	Large scale
Heavy Documentation			
Comprehensive Upfront Planning			
Autocratic management Style			
Not able to change			

17. How would you think the agile approaches affect cost of the three sizes of software projects than traditional methodologies?

	Small scale	Medium scale	Large scale
Very high			
Slightly high			
No change			
Slightly low			
Very low			

18. How would you think the agile approaches affect quality of the three sizes of software projects than traditional methodologies?

	Small scale	Medium scale	Large scale
Very high			
Slightly high			
No change			
Slightly low			
Very low			

19. To what extent do you follow agile techniques for the three sizes of projects?

	Small scale	Medium scale	Large scale
100% follow all agile techniques			
75% follow all agile techniques			
50% follow all agile techniques			
25% follow all agile techniques			

20. Which methodology do you prefer for different software projects?

	Small scale	Medium scale	Large scale
Agile methodologies			
Traditional methodologies			
Other (Please specify):			

Any suggestions or comments or your views regarding software projects and methodologies.

Optional

If you would like to have a summary of the survey results, please provide contact details

Name: ____________________

Email: ____________________

Organisation: ____________________

Thank you for all your valuable time in completing this questionnaire.