RESEARCH THESIS

COMPARISON OF DIFFERENT AGILE METHODOLOGIES AND DETERMINING WHICH IS MORE EFFECTIVE FOR HEAVY AND COMPLEX PROJECTS OF SOFTWARE INDUSTRY

SUBMITTED BY

SOHAIB EJAZ (03-298142-033)
SESSION 2014 – 2016

BAHRIA UNIVERSITY-LAHORE CAMPUS
LAHORE

SUBMISSION FORM OF THESIS FOR HIGHER RESEARCH DEGREE
BAHRIA UNIVERSITTY, LAHORE

Candidate Name: SOHAIB EJAZ,

I submit four Copies of thesis for examination for the degree of Mater of Sciences in Project Management, Thesis Titled: Comparison of different Agile Methodologies and determining which is more effective for Heavy and Complex Projects of Software Industry.

Candidate Signature: _____ Date: 12th April, 2016

Certificate of Principal Supervisor

I Dr. Ghulam Mohey-ud-din being the principal Supervisor for the above student, certify that thesis is in a form suitable for examination and that the candidate has pursued his course in accordance with the Rules of the University.

Signature: _____ Date: 12.04.2016

Recommendation for Examination

I recommend that the thesis be examined.

Principal Supervisor: _____ Date: 12.04.2016

Co-Supervisor: _____ Date: 12.04.2016

Not Recommended for Examination

I recommend that the thesis be examined.

Principal Supervisor: _____ Date: _____

Co-Supervisor: _____ Date: _____

Statement by the Head Faculty/Department

I support the submission of the thesis of the above named student for examination under the University Rules for higher degrees.

Signature: _____ Date: 13.04.2016

BAHRIA UNIVERSITY, LAHORE CAMPUS

APPROVAL SHEET

SUBMISSION OF HIGHER RESEARCH DEGREE THESIS

Candidate's Name:	**SOHAIB EJAZ**
Discipline:	**Project Management**
Faculty/Department:	**Department of Management Sciences**

I hereby certify that the above candidate's work, including the thesis, has been completed to my satisfaction and that the thesis is in a format and of an editorial standard recognized by the faculty/department as appropriate for examination.

Signature(s):

Principal Supervisor: Dr. Ghulam Mohey-ud-din

Date: 12th April 2016

The undersigned certify that:

1. The candidate presented at a pre-completion seminar, an overview and synthesis of major findings of the thesis, and that the research is of a standard and extent appropriate for submission as a thesis.
2. I have checked the candidate's thesis and its scope, format; editorial standards are recognized by the faculty/department as appropriate.

Signature(s):

Head of Management Science Department
Bahria University
LAHORE

Dean/Head of Faculty/Department:

Date: 13-04-2016

ABSTRACT

Project Management has reached new heights, new a days. The reason behind the scene is that, it has created a better understanding of how the projects are to be monitored in a better fashion, how the projects are to be evaluated in a better way and what kind of remedies are available in case if the project is not on track as it was decided in the planning phase. An eye opening intervention was the use of Agile Models. Agile models, contain a bucket full of different models with different properties. The organization which is implementing the agile models has to make sure that the model's properties must align with the project's nature. Agile models have been in practice for small scale projects since long ago. But current study involves the application of agile model in large and complex projects of Software Industry. By taking the expert's opinion, using questionnaire technique, few recommendations are set for this thesis.

With the help of questionnaire, different agile models were compared and best voted by the people of industry including the General Managers, Senior Managers, Project Managers, Team Leaders, Developers, Quality Assurance, Designer and Business Analyst chose models from Scrum, Extreme Programming, Dynamic System Development and Adaptive Software Development.

It was concluded that most of the project that were termed as heavy and complex were using models Scrum, Adaptive Software Development and Extreme Programming respectively.

Table of Contents

List of Tables

List of Figures

Chapter # 1

Introduction

1. Introduction

The term 'Project' has been a new word in the dictionary of many industries, unlike, software industry the word project has been decades ago. At first the project term was only associated with software industry but with the passing time the project got its root in every field of life like construction, mechanical and others. Like for instance, construction of a house, setting up an orange pulp plant. (Ken, 2002)

People have been hearing the project term over the years, and have created awareness about it. They have built up different strategies to handle them. Like, handling of a construction project would be different than designing of a new car design or construction of some software. Different industries use different types of strategies to handle their projects.

Projects of software industry have such a diversified nature that no hard and fast strategy can be applied to them. Some projects are complex, some are large scale, and some have changing requirement issue. Every particular scenario needs a different strategy. Some of the famous models for handling software projects are waterfall model, incremental model, iterative model, spiral model and so on so forth.

The first of all, waterfall model, simplest, easy to understand and comprehend, has been in practice for decades. But soon the industry felt the need of a new model which can handle and manage projects in a better fashion, unlike, that of waterfall model. Soon, other models

came into play and tried to fulfill either the short comings of other models or could adopt the changing nature of new project. The most and currently, used is Agile Model.

Agile Models, has a family of different models. At first, let's see the dictionary meaning of Agile. The dictionary meaning of agile is active, responsive or swift. In context of Project Management, it means that it is very active to such environments where requirements are not clear, and change very often.

Not all agile models can be discussed, few of them which will be discussed which *Extreme Programming (XP)*, *Scrum*, *Adaptive Software Development* and *Dynamic System Development Method.*

1.1 Extreme Programming (XP):

Extreme programming has been among the most advance used agile methods being used now a days. It has been in practice by several people but formally they were not aware of its name. After a number of projects have made their way to success through Extreme Programming (XP) this method was theorized. The word 'extreme' comes from taking these common sense practice and principles of software development to extreme level. It has five phases:

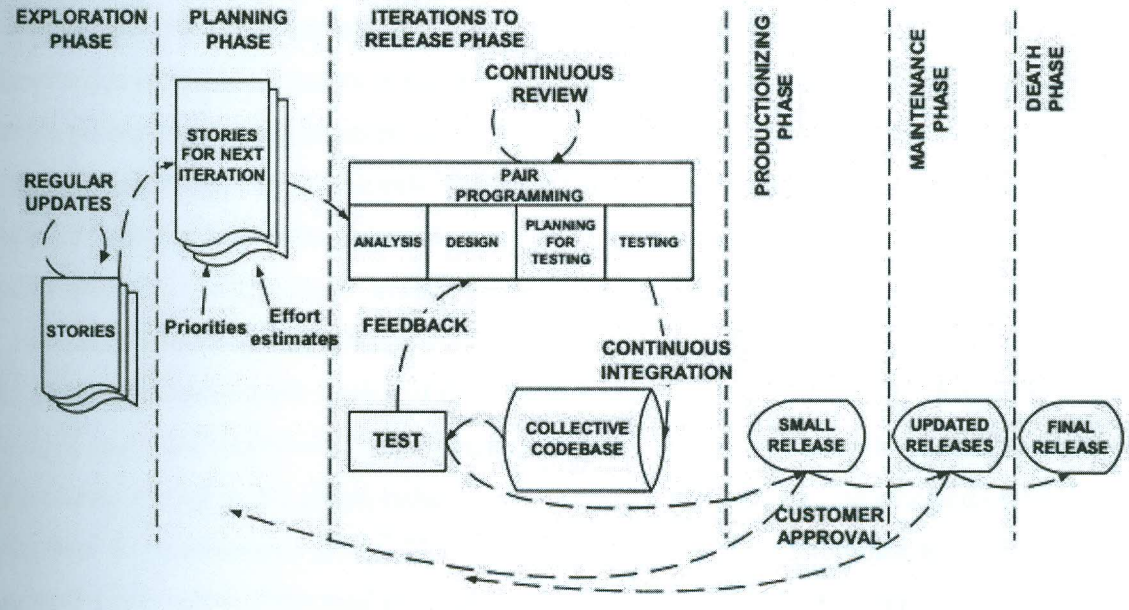


Figure 1.1

Source: Agile Software Development Methods, Review and Analysis- page 19

1.1.1 Phases:

1.1.1.1 Exploration phase:

In this phase user write user stories (use case) of what the component is supposed to do. Every user story or use case is basically a feature that is to be added into the system. In the same phase, depending upon the team skills and expertise to use the tool and technology can be enhanced. The team utilizes this time period to get grip and update itself from the user demanded technology.

1.1.1.2 Planning phase:

In planning phase, the stakeholder or the client decides what to construct first, i.e. he/she gives her preference about the user stories or the use case which he/she has written in exploration phase. The selected user stories are reviewed by the team and the manager and finalized them before going to start the project. Once the user stories have been finalized, the team sits and estimate the time duration and manger along with senior manager sits to decide the cost estimate of the

1.1.1.3 Iteration phase:

As name mentions, it is something that is to be done recursively, iteratively, again and again. This phase represents the actual development of the system. The development process is an iterative process which takes into account the requirements i.e. user stories or use cases and after a specified duration the expected product or sub-product or some deliverable is pulled out. The schedule in the planning phase is broken down to duration of usually 4 weeks named as sprint. Customer decides the user stories that are to be added to sprint at the start of each sprint. At the end of each iteration, the end product is tested by client. Similarly, so many iterations are done to achieve the final end product.

1.1.1.4 Productionizing phase:

This phase requires in depth testing of the system which is being constructed. Changes in the requirement are welcomed, but only major changes which are based on the architecture are not implemented in this phase, rather they are accumulative and are implemented in the next phase i.e. maintenance phase. The iterations in this phase are shortened to 1 week, usually it is one fourth of the total development time spend on this application.

1.1.1.5 Maintenance phase:

In this phase, the end product which was created at the end of productizing phase is deployed to client for testing or reviewing and the clients provide the feedback about the product. On the other hand, the team continues to add new features to the product which had already been backlogged. The new changes that the client desires are also welcomed. This phase also takes into account the customer support thus a new team may be created or the structure of the existing team may be changed depending upon need.

1.1.1.6 Death phase:

Death phase, when there are no more user stories i.e. all the user stories or use cases have been implemented and user founds the system satisfying in both the functional requirements as well as the non-functional requirements. Functional requirement contains user stories and the non-functional requirement concerns the attribute of product like reliability, performance and etc.

At this time, necessary documentation is made and finalized which contain all the technical things like architecture, design and code or anything else depending upon the organization as well as the client which are key stakeholders of the product. Death phase also may occur, in case the end product could not deliver the promised value to the client.

1.1.2 Roles and Responsibilities:

There are different roles which are performing different responsibilities in XP practice. Following are the roles and their responsibility, according to Beck (1999b):

1.1.2.1 Programmer:

Person responsible to write code and executes it. He is the one who gives shape to orally defined things to some logically thing. Usually, depending upon nature and complexity of project, the number of programmers are either increased or decreased. The more number of programmers the more difficult is to maintain communication between different team members.

1.1.2.2 Customer:

Customer is responsible for defining functionality of the software and writing user stories, and to prioritize them so that the team could develop the high priority features from the very start.

1.1.2.3 Tester:

Tester, test the application and his main focus is to write test scenario and within each scenario write test cases. So, at the time of application the tester can execute the test cases and check the functionality of the system being developed.

1.1.2.4 Tracker:

Tracker, is the person who tracks the progress of the project and provides the feedback on how accurate they are in order to improve the future estimations. He monitors the progress of each iteration and evaluates if goals are achievable within provided time and resource constraints.

1.1.2.4 Coach:

This person is responsible for the correct implementation of all the process of XP. Person with this role should have a sound understanding of the XP and could correct if any process deviates.

1.1.2.5 Consultant:

This person is external to the project team having sound and good knowledge of both the domain and in-depth understanding of the technology. This role provides the consultancy in solving problems

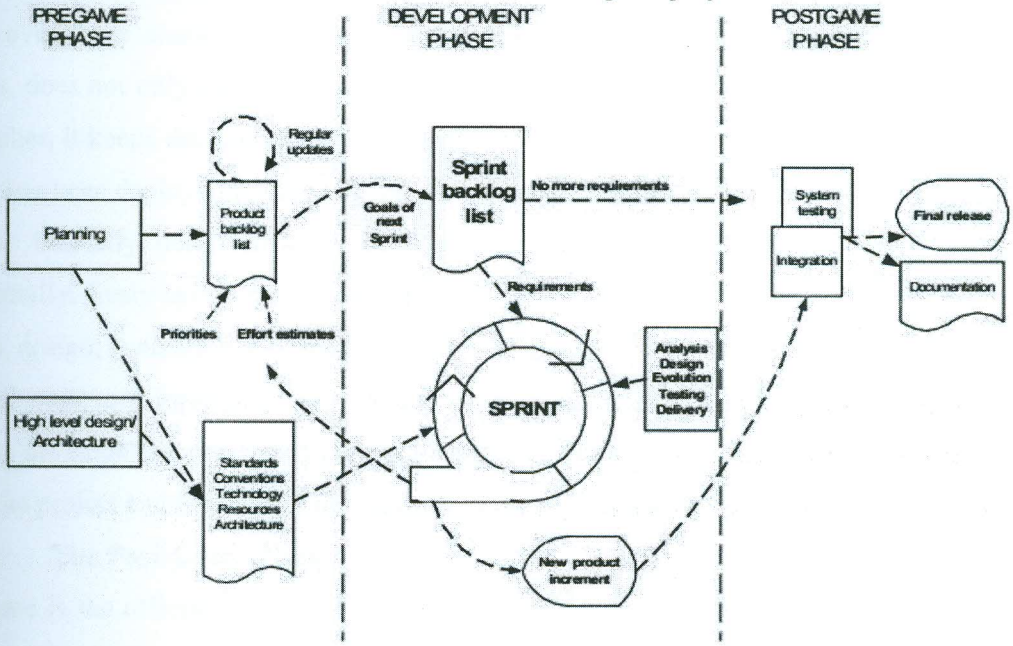
1.1.2.6 Manager (Big Boss):

Person having role of Manager has complete authority to make any decisions depending upon circumstances. In order to exercise full control of decision making, the project manager, calls meeting and discuss the situation with other team members and communicate the decision.

1.2 Scrum:

The term 'scrum' is derived from a strategy in the game of rugby where it denotes "getting an out-of play back into the game" with teamwork. This approach helps in managing the system which is being constructed in a better fashion. Scrum is basically applying the industry process to system being developed thus adding more flexibility, adaptability and productivity. Scrum is a way in which team members function as a whole to produce a system in changing environment.

The development process involves different constraints which include but are not limited to time constraint, cost constraint, resources and technology constraint, which makes it more complex and unpredictable. So, the use of traditional development models makes it really hard to predict all the events that would occur during the project execution.



Source: Agile Software Development Methods, Review and Analysis- page 28

Figure 1.2

1.2.1 Phases

The above mentioned process, first introduced by Schwaber (1995), has three main phases.

Pre-Game Phase, Development Phase and the Post-Game Phase.

1.2.1.1 Pre-Game Phase:

In this phase, a product backlog is created, the customer and other stakeholders write user stories or use cases, as functional requirements of the system, and add them to the product backlog. The customer then prioritize each user story so that once the iteration starts all those requirements are taken into account that are of high priority. The backlog is constantly updated with new user stories or modification to existing user stories or deleting the existing user stories. At the beginning of every new iteration, new user stories with high priority are added to team development.

1.2.1.2 Development Phase:

In the development phase the team is fully ready to expect the unexpected events which will try to take the project off from the track. Changing requirements, quality, scheduling, tools and technologies are few constraints which are prone to risk, scrum practices exercise control over these areas during sprint in the development phase. Scrum, unlike other methods, does not only take measures at the beginning of the software development life cycle rather, it keeps the team on toes throughout the software development cycle till the product has been deployed to client successfully.

Sprint is basically iterative cycles in which the software is software is developed incrementally. Every sprint has all the steps of any traditional method, like, requirements, analysis, design, evaluation and finally delivery. The time span of one sprint usually limits from one week to a duration of four weeks. There could be like five to ten sprints before complete development and deployment of the system. Number of sprints depends upon the size of the project and the change in requirements from the stakeholder.

1.2.1.3 The Post-Game Phase:

This phase is the official closure of the project. When a project reaches this phase, this means that no new requirements are to be added in the product and the product has reached the end state. All the issues have been fixed and the desired quality has been achieved. Some documentation is made and the product is finally released to client. All the components are integrated and a single system is deployed.

1.2.2 Roles and Responsibilities:

According to Schwaber, there are a total of six roles identified in scrum. Following are the roles:

1.2.2.1 Scrum Master

Scrum master is the person, who is responsible for carrying and ensuring that the team follows the actual practices of scrum. He/she is also responsible to make sure that the project is right on the track as planned. Scrum master behaves like a bridge between the team, the customer and the management.

1.2.2.2 Product Owner

Product owner is responsible for the project, its evaluation, monitoring and controlling and also the backlog items. Scrum master, management and customer select the person with

job role of product owner. His/her decision, related to product, is considered as final. He is also responsible for making sure that all the items in the backlog are developed and takes an active role in estimating the effort for each task.

1.2.2.3 Scrum Team

The team which has been assigned to the project, whose sole responsibility is to achieve the goals through each iteration or sprint and add value to the product as planned. Team's major role is to develop and give shape to user stories but they also does estimating effort, reviewing backlog items and suggest any change depending upon its feasibility.

1.2.2.4 Customer

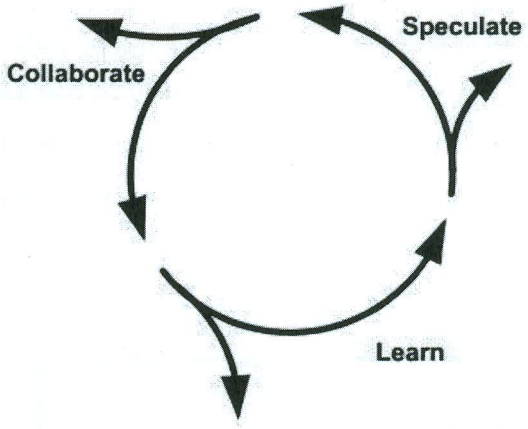
Customer writes user stories, prioritize them. Review the deliverables and provide the feedback. Feedback may contains request for change depending upon the need of client.

1.2.2.5 Management

Management plays an important role in decision making along with product owner, scrum master and the customer. Management's role also comes into play during the requirement gathering phase.

1.3 Adaptive Software Development:

ASD (Adaptive Software Development), was developed by James A. Highsmith III and was published in 2000. Adaptive Software Development had its major roots from an approach called RAD (RADical Software Development), which was again developed by Highsmith and S. Bayer in 1994. The objective of ASD is to provide development team with enough of the guidelines so the project could not fall to failure, but they do not provide too much guidelines, like other agile models, which could suppress the creativity.



Source: Agile Software Development Methods, Review and Analysis- page 69

Figure: 1.3 Adaptive Software Development Phases.

1.3.1 Phases:

ASD, has been divided in three main phases, as mentioned in Highsmith 2000. Following are the three phases of Adaptive Software Development:

1.3.1.1 Speculate:

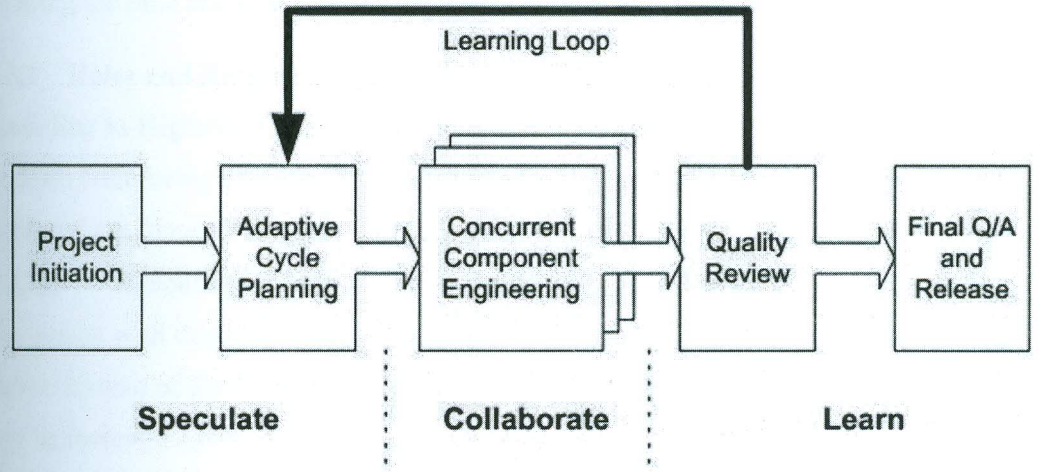
Like all other agile models as well as other traditional models, planning is the first phase. Similarly, in adaptive software development, speculate is the phase. This phase is equivalent to planning phase, but the name had its background that conventionally if the project does not follow the set plan, it is considered to be failure. As, adaptive software development does provide only sufficient guidelines to avoid the failure and like other agile models does not stick to some plan in advance and supports the creativity. Also, the plan represents uncertainty but in speculate there is nothing like uncertainty.

1.3.1.2 Collaborate:

The word collaborate the phase meaning in very much detail. Dictionary meaning of collaborate is work together, this means that this phase represent the team effort and binds them together to a single node so that they can work together to make a successful system. Collaboration plays an important role in construction of any software development. As this is agile model so it focuses to adapt to changes, and for this a combine effort of the team is of mere importance.

1.3.1.3 Learn:

The learn phase is very important phase of any model used to manage the project. Other models use this but they don't put so much stress on this phase but adaptive software development does. That's why it has been explicitly mentioned in this phase while others take it to be implicitly. Learn phase means to learn from mistakes and to apply the learning of other projects to some other projects. This is basically Organizational Process Assets (OPA).



Source: Agile Software Development Methods, Review and Analysis- page 70

Figure: 1.3(b) Adaptive Software Development Life Cycle

Speculate phase includes both project initiation and starting the iteration of ASD cycle. In initiation the project is to conceive logically and is agreed by both parties i.e. the company and client agree upon some conditions and come in some contract. Then the next step of starting the iteration, team decides what task to do first or sometimes the client puts the preference and guides the team from where to start.

Collaborate phase includes multiple components which are under construction concurrently, team collaborates with each other and the project manager to keep things in control and be able to manage them efficiently and effectively. Different components once have been completed are merged to one system so that the quality person can easily verify the system.

Once the collaboration phase ends, the last phase of learn begins. There are two parts of this phase. The first part relates to the quality review by the in house quality people. But sometimes the quality review includes people like client or client representative. And both the client and development team validates the system which has been developed. This is known as JAD (Joint Application Development). The client is shown UI of the system which is to be developed. After the approval of the client the UI is further taken to next level of development. This helps in early identification of bugs, defects and issues. Thus avoiding the time and cost which was to be added for adjustment of system.

1.3.2 Roles and Responsibilities:

According to Highsmith, there are no formal roles defined in ASD, as it does not restrict the team from being creative. The major roles which are identified in all agile models are listed and are found in adaptive software development.

1.3.2.1 Executive Sponsor:

The person with this role is overall responsible for the product which is being developed. He/she is most of the time in JAD (Joint Application Development), so that requirements may be review.

1.3.2.2 Scribe:

The person with this role has main focus to note the MoM (Minutes of Meeting), so that the client and development team can review it afterward.

1.3.2.3 Facilitator:

This person is responsible for conducting a session between client representative and the developer representative. Facilitator invokes the people for JAD (Joint Application Development), or any other meeting.

1.3.2.4 Project Manager:

Person with the role of Project Manager is to manage the project. Keep a track of what's happening and how much effort is being inputted by the team members to complete the mission of the project.

1.3.2.5 Customer:

The person for whom the system is being developed, could be end user too.

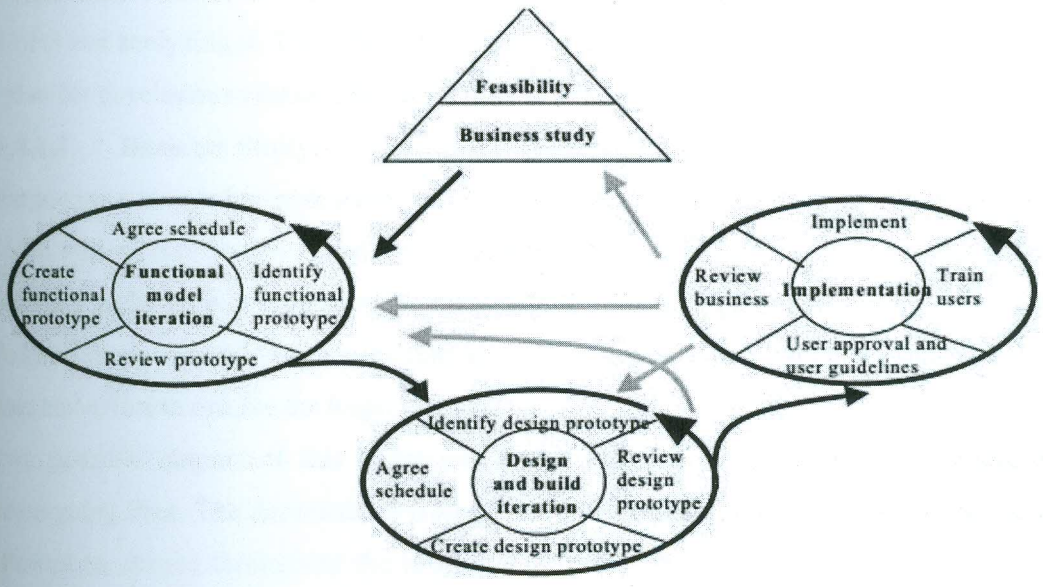
1.3.2.6 Developer Representative:

The person with this role is responsible for attending the JAD sessions, also responsible for making the said changes during the session and plays a role of communicator between the development team and the client.

1.4 Dynamic System Development Method:

Dynamic System Development Method was developed in 1994, since its development it has been a major breakthrough as its root resides in RAD (Rapid Application Development). DSDM (Dynamic System Development Method) has been created by DSDM Consortium.

DSDM is different from other agile methods in the sense that other agile models fix the amount of functionality of the product and then adjust time and resource, but in Dynamic System Development Method first the time and resources are fixed then the amount of work is adjusted. This means that there is very little chance that the project mat run short of schedule.



Source: Agile Software Development Methods, Review and Analysis- page 62

Figure: 1.4 Dynamic System Development Method Life Cycle

1.4.1 Phases:

Dynamic Software Development Method consist of total of 5 phases. The first two phases happened to be once in product's life cycle while the remaining three phases are in iteration. The actual development of the product starts with the last three phases. DSDM consider iterations as timeboxes. Timebox has some specified duration and every iteration is mandatory to end within the defined timebox. The duration of timebox varies from few days to few weeks.

Following are the phases of DSDM (Dynamic Software Development Method)

1.4.1.1 Feasibility Study:

In feasibility phase, two things are checked for their feasibility. One is the implementation of the model and the other is the technical aspect. In the first part the suitability of applying the DSDM (Dynamic Software Development Method) to the project is reviewed. Few factor like project nature, organizational culture and conflict resolution management that has been implemented in the organization all should be align with the DSDM. Other part of feasibility is technical possibilities and reviewing the requirements and domain of the project and analyzing it. The output of this phase may involve with a feasibility study and a plan for development team. This phase should not take more than few weeks.

1.4.1.2 Business Study:

The second phase is business study and it consists of business case and the technology in which system is to be developed. In this phase, the end user is identified and all possible users are gathered in a workshop, so that all information and domain can be understood clearly. Client and end user are invoked in early stages so as to avoid bugs and fixes and time and effort to resolve the bugs.

Two possible outputs of this phase are System Architecture Definition and other is Prototyping Plan. The Architecture is identified in its initial phase of development and is allowed to change throughout the project. The Prototyping plan is made for upcoming phases and also a configuration Management Plan is added.

1.4.1.3 Functional Model Iteration:

The Functional Model Iteration is the first phase which is both iterative as well as incremental. The duration of the iteration is finalized by the team as well as by the client and both parties agree upon the duration. The next step in this phase is the identification of

the functional prototype. Prototype is reviewed by the client and other stakeholders and then finalized so that further work could be done. The prototype developed for client review is evolutionary prototype. The initial prototype developed is made on some tool which is further then taken to final stage of development.

There are also few output from this phase amongst which prioritized functions i.e. it is a list of functions with high priority which are to be delivery in that particular iteration. The prototype for review collects the users' feedback about the system and the feedback is then added in next iteration. At the same time non-functional requirements are added on client request, if any. For future, risk analysis is done to avoid any major issue or misunderstanding of any requirement.

1.4.1.4 Design and Build Iteration:

It is the phase where actual development is done and software is develop. The output of this phase is a working piece of software. This software is tested and until the system fulfils the minimum required and then it is deployed. Further development depends upon the client's feedback.

1.4.1.5 Implementation:

In this phase the system is deployed as a whole to client from development environment to actual environment. End users are trained and given necessary help to use the system. The output may include a User Manual and Project Report.

1.4.2 Roles and Responsibilities

In DSDM (Dynamic System Development Method), almost 15 different roles have been defined, but according to Stapleton, most important ones are following:

1.4.2.1 Developers and Senior Developers

According to DSDM, the main body of development team is developers and senior developers. Seniority is set on the basis of experience which also set the hierarchy of the team. This contains all the team members including the designer, analyst and quality person.

1.4.2.2 Technical Coordinator

The person with this role is responsible for the technical aspect of the project, like for instance, the application architecture and quality. This person is also responsible for the software configuration.

1.4.2.3 Ambassador User

The most important role amongst the user is the Ambassador User role. The responsibility of this person is to bring the user reviews, knowledge and feedback to the development team and to take the suggestions and progress of the project back to user. These roles ensure a timely feedback is received to development team. Ambassador User could be an end user who has complete knowledge of the domain.

1.4.2.4 Adviser User

Person with this role has similar task and responsibility like the Ambassador User does. As, ambassador user cannot present the whole user's review and feedback so an additional role of Adviser User is set. An example of adviser user could be IT people or financial auditors.

1.4.2.5 Visionary

The person with this role, is the person who initiated the idea of the system which is to be developed. He/she ensures that the functional requirements are gathered early in the life of product, so that development team can work on them. The person with this role also ensures that the work of the development team is going on right track.

1.4.2.6 Executive Sponsor

The person with this role belongs to user area, and deals with the finance involved for the development of the system. He/she has ultimate authority to make decisions.

Chapter # 2

Review of Literature

Review of Literature

Lassenius (2012) focuses of using agile models including scrum and extreme programming on large scale projects. This article states that the use of agile models have a positive impact on software projects of small scale, but there is very little study done of similar nature and impact of agile model on large projects have yet unknown results. The author provided evidence using different case studies to the fact that the agile model have been a success for smaller projects and then he pulled in the same agile practices which small projects relate or implement to large projects to see the impact of the agile practices on it. Larger projects with changing requirements and with greater amount of risk, has to be managed and organized. A case study approach is used. Data collected contained interview and semi-open ended questionnaire.

I have chosen this article as it relates to my studies, this articles relates the benefits of using agile models on large software projects as we are already getting benefits of using agile models such as scrum, extreme programming, distributed scrum, adaptive programming and others. My studies also involve the implementation of agile model which gives maximum of benefit to the project team and produces the best quality product for client.

Abrahamsson et al. (2002), focuses on using agile software development methods, and providing review and analysis, and focuses on detail understanding of different agile models which include following, Scrum, Extreme Programming, Crystal Family of Methodologies, Feature Driven Development, Rational Unified Process, Dynamic Systems Development Method, Adaptive Software Development, Open Source Software Development and other agile model including Agile Modeling and Pragmatic Programming. The comparison of above mentioned models including both positive aspects as well as negative aspects of all the models and their usage regarding software nature and type of project in which these models are to be used. This book refers agile models as a solution to the rapid changing environment and trends in software industry. This book also states that the research done in the agile models is still less and many of the flexibility of agile models in still unknown and hidden from many practitioners.

I have chosen this book in my studies for the reason that this book is very good for the understanding of different agile models like what is the process, what are the roles and responsibilities of project team, what are the practices that this model follows, what is its scope of use and last but not the least how and which type of projects are more feasible to be using this particular type of agile model. For me, the most important of all the things is the nature of project and feasibility of use of that model. Also, I will be working on and comparing all the above mentioned agile models in my studies.

Abrahamsson et al. (2003) focuses on comparison based study of different agile model including based on model's life cycle coverage, project management support, type of practical guidance, fitness-for-use, and empirical use of analytical lenses. In this research, researcher states that the use of agile models have now a days become famous due to their working benefits and flexibility towards the project nature as well as projects team and client.

I have chosen this research for my studies for the reason that this study put a light on the different agile models and their fitness-for-use feature. This feature will tell us which model best fit which kind of project i.e. project nature, my study also include the which agile models best fit for which kind of project depending upon project's nature also the processes used in that project.

Strode et al. (2009), focuses on one of the hidden reasons of the project failure which is the project team culture. In software industry it is very common for a scenario that there is graphically distributed team and in such scenario it is very difficult to choose a model which best suits all the project members according to their culture. It is sometimes the project team's culture which makes project either a success or a failure. In this study the author is trying to map the relation between both the project team's culture and the type of agile model used so as to avoid this fact of software development models. Case study approach was taken between the organizational culture and agile method used.

I have chosen this research studies as it have a relationship with my studies. This research will help me in choosing the agile model which not only helps the project team but also the project itself to reach success. Processes are implemented so that people may use them to get ease and to work in more comfortable environment.

Boehm (2002), focuses on comparisons of agile models with traditional plan driven models and elaborate both the positive as well as negative side of them. This study also compare different agile models on some set grounds like Developers, Customers, Requirements, Architecture, Refactoring, Size and Primary Objectives of the project being undertaken. This publication helps in identifying the steps required for using agile in different types and nature of large software projects in which people have been using traditional models or heavy weight models like waterfall model, spiral model and others.

I have chosen this article for my studies as it is bridging up the gap which I had identified in my work i.e. usage of agile models like Scrum, Extreme Programming, Feature Driven Development and others in software projects which are termed as large and heavy. This will help me in selecting my methodology and data sources for my research as it has somehow the other implemented the same.

Awad (2005), focus in comparison of both agile models as well as traditional software development models. This report not only categorizes the models in either heavy weight or as light weight. Heavy weight models are those which have a comprehensive planning, lot of documentation, and expansive design, whereas, on the other hand a light weight model does not have a detail planning, does not have a lot of documentation involved and does

not either have so much depth of design involved. In this report, the author has discussed both the pros and corns of agile model as well as traditional software development models. The suitability of software development model has also been discussed in this report. Questionnaire method was used for comparison between agile and traditional software development technologies.

I have chosen this report as it has a part which overlaps my work, the suitability of software development model depending upon size of the project and nature of the project which is being undertaken. This also contain the comparison of different software development models including both the heavy weight models like Waterfall model, Spiral model, Incremental model and others to agile models like Scrum, Extreme Programming, Adaptive Software Development and others.

Paetsch et al. (2005), put a light on one of the most important and critical phase of software development which is requirement gathering from client which is most of the time a non-technical person. This study put both traditional way of gather requirements and focuses more on Prototype model which is a heavy weight model, which is considered to be the best for requirement gathering. As agile models have opened new dimensions of software development, it has also unlocked and override the traditional way of gathering requirements from clients and other stakeholders. This research has also compare different agile models in context of requirement gathering where project nature and client nature matters a lot.

I have chosen this research as it will help in nominating and putting forward that particular agile model that has good approach not all in development but also in requirement gathering, which would be a complete package for any project team to pick that model and its associated tools for the development of software starting from initialization till the delivery of the actual product.

Strode (2005), conducted the environmental factors which were suitable for the implementation of agile methods, according to the author, agile models cannot be implemented only just viewing the project nature. There are a lot many things that should be kept in mind before the implementation of any software development technique like

agile model. The author compared five earliest published agile models and decomposing them into parts like scope, input, output and the extent to which those models have been in practice. A mixed research methodology was adopted by author, both qualitative as well as quantitative techniques were used as both the questionnaire and interviews were conducted to gather data and non-parametric quantitative data analysis was performed which was to determine a relation between environmental factors and usage of agile methodology. Questionnaire method was used to compare different agile methods and the environment in which they are best to use.

This article relates with my research in the way that I will be comparing the agile models with the project nature, few of my research and Strode research overlap is the characteristics of the agile models like the team size, project complexity as well as usage of new technology in industry. Unlike Strode, I will be gathering data only through questionnaire and then apply different test and run different analysis to generate the final outcome.

Huo et al. (2004), study the impact of traditional software development methodologies like waterfall model with agile models in the context of software quality assurance. According to author, the author takes the side that the agile models can develop software faster than other traditional models. The point of discussion is that the software developed through traditional models maintains a level of quality the main reason behind this is static requirements. On the other hand in agile, the requirements are always changing according to client so it would become more difficult for the model to handle the requirements.

This article relates with my studies in the context of the development of quality software using agile methodologies. Quality being the key success factor of any system which is under construction. Agile methods are not only models of development of any software rather it has list of models which has their own dimensions and they also work upon the quality of software.

Dyba et al (2008), studied a system review of the empirical study of the agile software development. The study was categorically divided into four groups, namely, introduction, social factors, perception of agile methods and comparison. The study enlisted the strength

and weakness of agile methods. According to researcher, '*No systematic review of agile software development research has previously been published. The existing reviews that were presented in the previous section only partially cover the empirical studies that exist today. Further, the previous reviews do not include any assessment of the quality of the published studies, as in this systematic review.*'

This article relates with my studies in the context as it compare different agile methods on different aspects. Quality has been maintained in data gathering by using a simple quality assessment form.

Chapter # 3

Research Methodology

3.2 Questionnaire Categories

Questionnaire was kept easy and small keeping in view that the people of software industry does not have enough time for long and complex questions. The questions of the questionnaire were categorized into four different categories.

3.2.1 General Questions – this category contains questions related to every individual respondent. Like his/her designation, company, work experience.

3.2.2 Project Nature – this category deals with questions related to project's nature. Questions of this category will decide if the project is complex and heavy.

3.2.3 Software Development Questions – this category contains questions related to software's nature and model's nature used.

3.2.4 Effect of model with triple constraints – this category deals with questions related to the impact of model which is being used on cost, quality and schedule.

3.3 Unit of Analysis

Employees from different software houses with different rank like, General Manager, Senior Manager, Project Manager, Team-leader, Developer, Quality Assurance, Designer and Business Analyst are unit of measure in my research.

3.4 Instrument Reliability

Reliability of questionnaire was calculated using Cronbach's Alpha, and the average of the survey questionnaire were 0.866. This value is greater than 0.7 factor, which indicates that questionnaire is reliable.

3.5 Content / Face Validity

Before the questionnaire was given to public for filling it, its standardization was ensured by reviewing it to multiple people of different domains. People from industry as well as people for academics reviewed it before finalizing.

3.6 Data Collection

The mode of data collection was questionnaire, which was created on google forms off which have scale of 5 values, starting from 'Very Low' to 'Very High', other questions having different values so as to get the values and reviews of different people.

Chapter # 4

Data Analysis and Results

4. Data Analysis and Results

4.1 Questionnaire Results

4.1.1 Please specify your designation.

Please specify your designation

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	General Manager	2	1.8	1.8	1.8
	Senior Manager	7	6.3	6.3	8.0
	Project Manager	17	15.2	15.2	23.2
	Team Leader	30	26.8	26.8	50.0
	Developer	31	27.7	27.7	77.7
	Quality Assurance	12	10.7	10.7	88.4
	Designer	8	7.1	7.1	95.5
	Business Analyst	5	4.5	4.5	100.0
	Total	112	100.0	100.0	

Table 4.1.1 (a)

This table shows the demographic of the respondent. The frequency column shows the number of respondent for each category. The total were 112 respondents out of which 2 were 'General Manager', 7 were 'Senior Manager', 17 were 'Project Manager', 30 were 'Team Leader', 31 were 'Developer', 12 were 'Quality Assurance', 8 were 'Designer' and 5 were 'Business Analyst'.

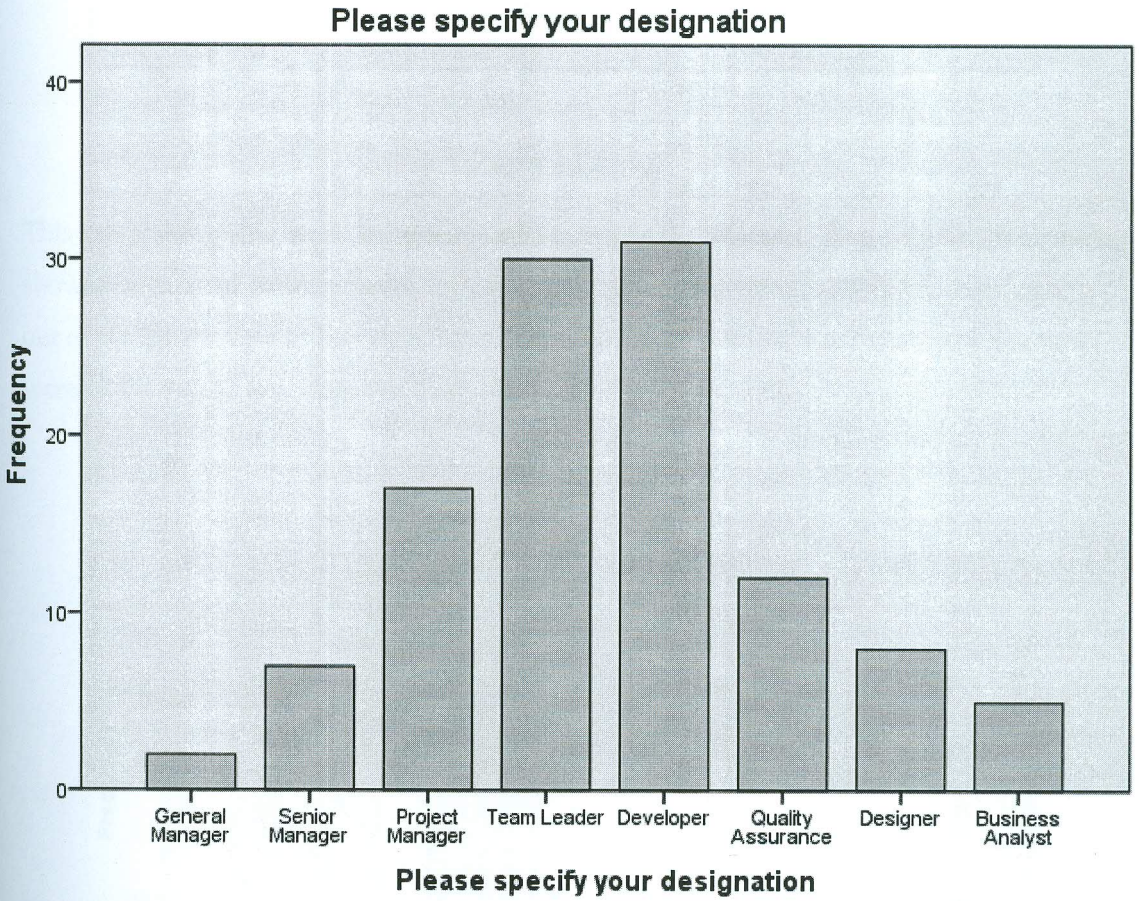


Figure 4.1.1 (b)

The above bar chart depicts the same information but in graphically form, so that it can be easily understood.

4.1.2 Your work experience

Your Work experience

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Less than 5 year	29	25.9	25.9	25.9
	5 - 7 Years	29	25.9	25.9	51.8
	7 - 9 Years	27	24.1	24.1	75.9
	More than 9 Years	27	24.1	24.1	100.0
	Total	112	100.0	100.0	

Table 4.1.2 (a)

This table shows the work experience of the entire respondents. The column frequency shows the number of respondent for each category. There were total of 112 respondents, out of which 29 were of less than 5 year experience, 29 were of 5 – 7 year experience, 27 were of 7 – 9 year experience and remaining 27 were of more than 9 year experience.

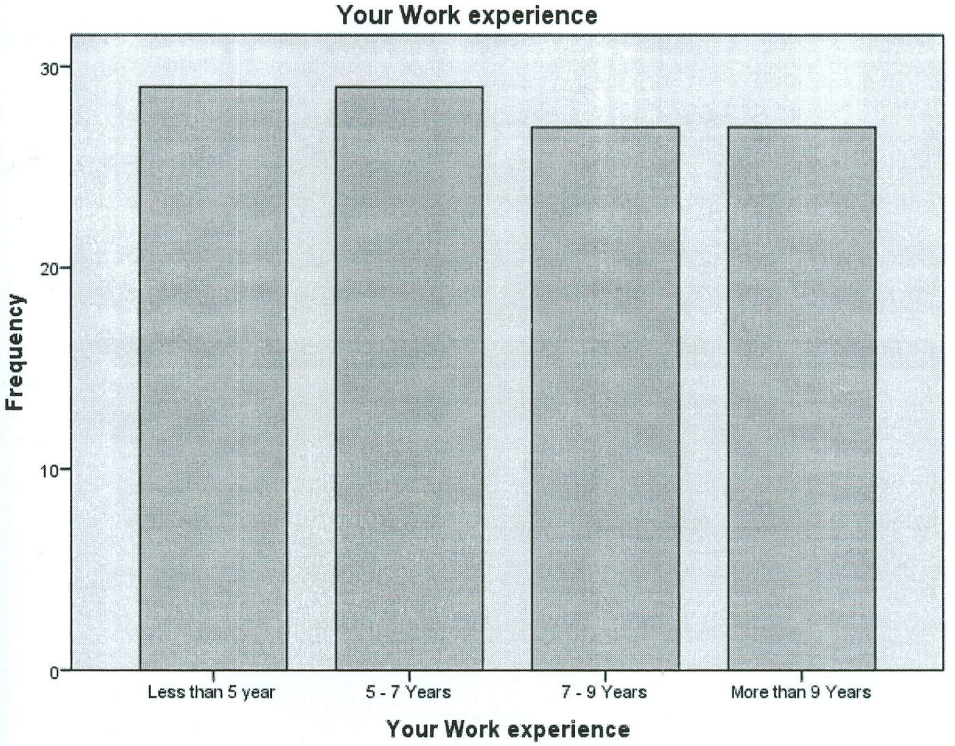


Figure 4.1.2 (b)

The above bar-chart represents the same information but in graphically form so that it is easy to understand.

4.1.3 Are you applying any Agile Model in current project

Are you applying any Agile Model in current project

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	No	21	18.8	18.8	18.8
	Yes, but I have worked in my past project	18	16.1	16.1	34.8
	Yes	73	65.2	65.2	100.0
	Total	112	100.0	100.0	

Table 4.1.3 (a)

This table shows that out of 112 project, 21 respondents have never used any agile models, whereas 73 respondents were currently using agile models in their projects and remaining 18 respondents have used agile models in their past projects.

The user of agile methods is about 65.2%, which is clear that currently agile methods are being used more than any other traditional method.

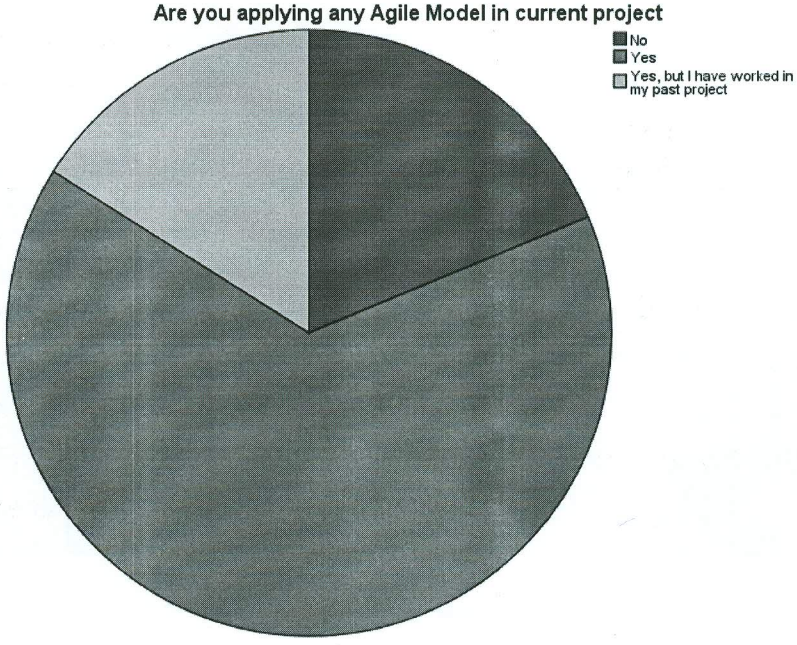


Table 4.1.3 (b)

The above pie-chart depicts the same information as that of table 4.3 (a), but the graphically nature of the pie-chart makes it easy to understand.

4.1.4 Your current Project team size

Your current Project team size

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	2	8	7.1	7.1	7.1
	3	27	24.1	24.1	31.3
	More than 3	77	68.8	68.8	100.0
	Total	112	100.0	100.0	

Table 4.1.4 (a)

This table shows that out of 112 projects taken as sample, 8 projects have team size of 2 developers, 27 projects were having a team size of 3 developers and 77 projects were having team size more than 3. According to my research teams whose size is 3 or more are considered as complex projects. Therefore, 92.9% projects being conducted are complex in nature.

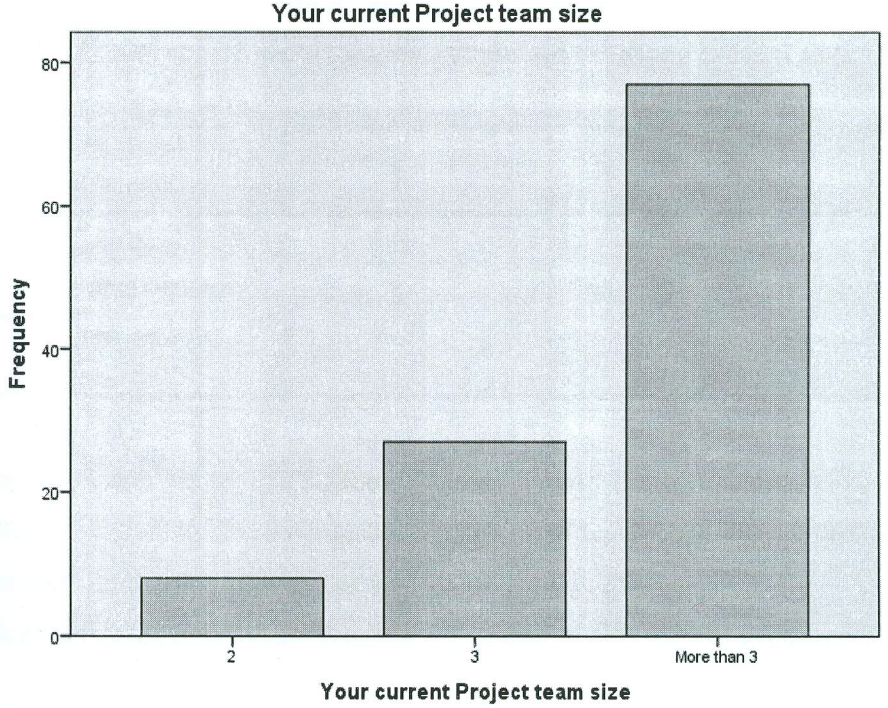


Table 4.1.4 (b)

This bar chart displays the same information as table 4.4 (a) does, but the graphically nature of bar chart makes it easy to understand.

4.1.5 Studying the relationship between the team size and the usage of agile methods.

Your current Project team size * Are you applying any Agile Model in current project Cross tabulation

Count

		Are you applying any Agile Model in current project			Total
		No	Yes, but I have worked in my past project	Yes	
Your current Project team size	2	3	1	4	8
	3	10	2	15	27
	More than 3	8	15	54	77
Total		21	18	73	112

Table 4.1.5 (a)

This table shows the relationship between the project team size and the usage of agile models. This indicates that the larger the team size the more usage of agile methods, for the reason that agile methods make it easy to manage the project.

4.1.6 Your current Project contains, estimate, how many LOC (Lines of Code)

Your current Project contains, estimate, how many LOC (Lines of Code)

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Less than 10000	9	8.0	8.0	8.0
	10000 – 50000	31	27.7	27.7	35.7
	More than 50000	72	64.3	64.3	100.0
	Total	112	100.0	100.0	

Table 4.1.6 (a)

This table shows that the out of 112, there are only 9 projects in the sample which has less than 10000 lines of code, and 31 projects whose line of code range from 10000 to 50000 and 72 projects whose lines of code exceeds more than 50000.

This indicates that the projects have usually more than 50000 lines of code.

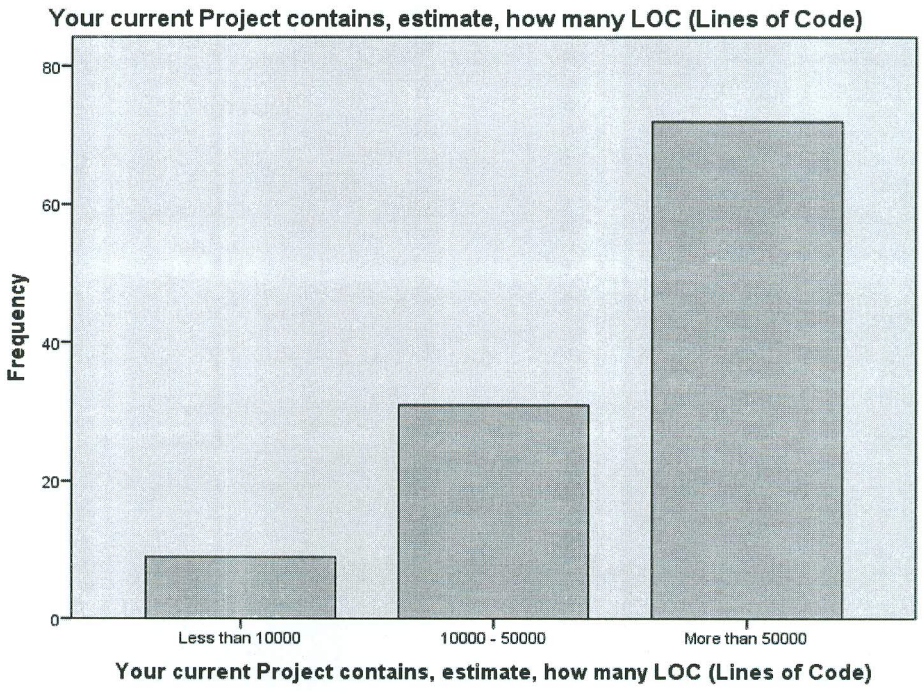


Table 4.1.6 (b)

This bar-chart displays the same information as the table 4.5 (a), but the graphically nature of bar-chart makes it easy to understand.

4.1.7 Which Model from following Agile Model do you feel comfortable in your project

Which Model from following Agile Model do you feel comfortable in your project

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Scrum	54	48.2	59.3	59.3
	Extreme Programing	14	12.5	15.4	74.7
	Adaptive Software Development	23	20.5	25.3	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.7 (a)

This table shows the usages of agile models in different projects i.e. out of 91 projects 54 projects apply scrum, 14 projects apply extreme programming and 23 projects apply adaptive software development.

This also shows that scrum is used in almost 59.3% projects and adaptive software development is used in almost 25.3% projects.

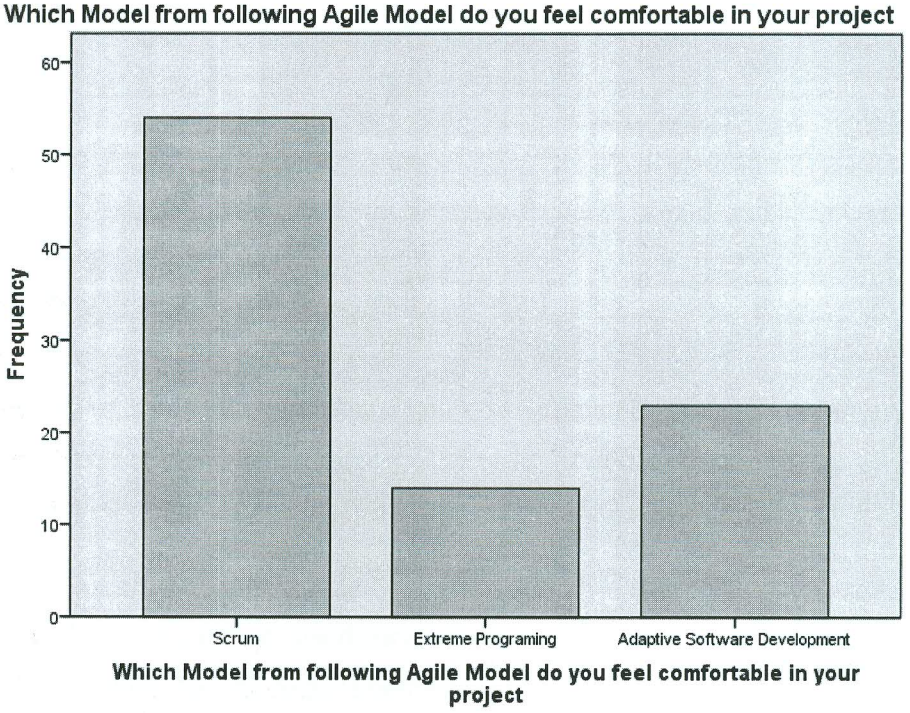


Table 4.1.7 (b)

This bar-chart displays the same information as table 4.1.7 (a), but the graphically nature of bar-chart makes it easy to understand.

4.1.8 How many request for changes to new functional requirements been received

How many request for changes to new functional requirements have you received in the past two weeks?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very Low	7	6.3	7.7	7.7
	Low	20	17.9	22.0	29.7
	Medium	32	28.6	35.2	64.8
	High	25	22.3	27.5	92.3
	Very High	7	6.3	7.7	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.8 (a)

This table shows the number of change request in different projects. It is clear that out of 91 projects 7 has very low change request, 20 projects have low change request, 32 projects have medium amount of change request, 25 projects were having high change request and 7 were projects which were having very high change request.

This mean change request is mandatory no matter how much the requirements were defined. The medium has the highest percentage of change request with 35.2% and high being the second highest percentage of change request with 27.5%. The major reason of stressing with the change request is that in traditional models the change request is not acceptable once the requirement phase has officially ended, to encounter change request the model should be light weight so as to move the iteration to and fro.

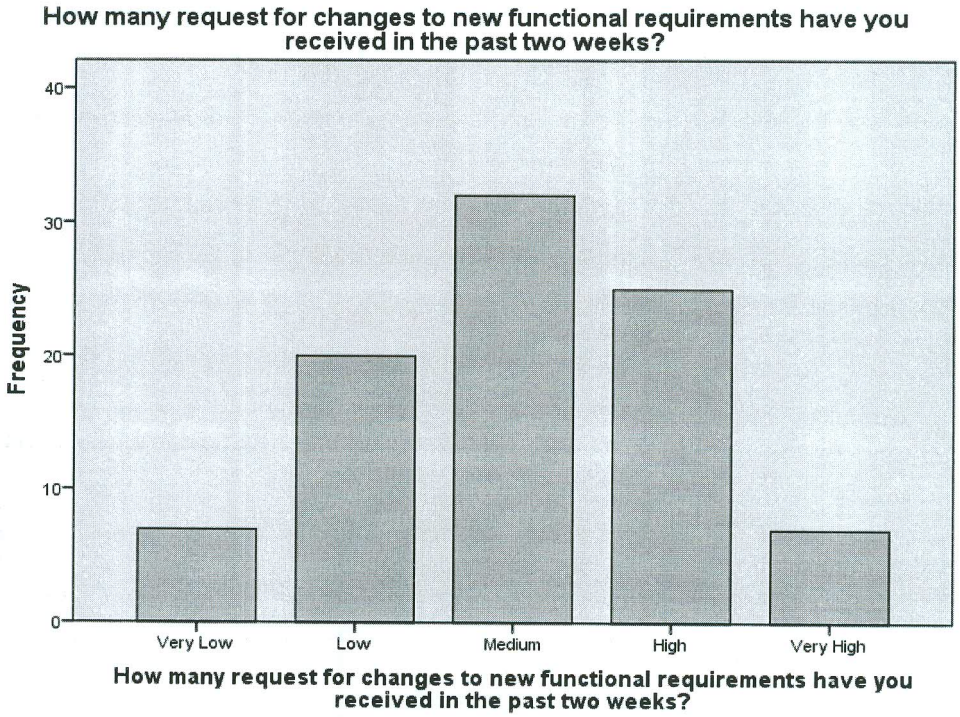


Table 4.1.8 (b)

The bar-chart indicates that change request for functional requirements made by either client or the team due to feasibility.

4.1.9 Studying the relationship between the model and new request in requirement?

How many request for changes to new functional requirements have you received in the past two weeks? * Which Model from following Agile Model do you feel comfortable in your project cross tabulation

Count

		Which Model from following Agile Model do you feel comfortable in your project			Total
		Scrum	Extreme Programing	Adaptive Software Development	
How many request for changes to new functional requirements have you	Very Low	4	2	1	7
	Low	14	1	5	20
	Medium	18	3	11	32

received in the past two weeks?	High	12	8	5	25
	Very High	6	0	1	7
Total		54	14	23	91

Table 4.1.9 (a)

This table indicates that 54 projects out of 91 use scrum and 14 use extreme programming while the remaining 23 uses adaptive software development. But on the other hand, this can be depicted, change request is mandatory, so many of the project managers and team leaders prefer the model which can easily handle the changes.

4.1.10 What makes you feel comfortable to use current model?

What makes you feel comfortable to use the current model?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Easy to understand	31	27.7	34.1	34.1
	Iterative	36	32.1	39.6	73.6
	Stability	24	21.4	26.4	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.10 (a)

This table indicates the characteristics of the models being used. Three characteristics namely, easy to understand, iterative and stability were used from respondents. Out of 91 respondents received 31 admitted that the model they were using is easy to understand, 36 admitted that the model they used is iterative and remaining 24 considered the model they used is stable. Most of the respondent chose the characteristic of iterative, which mean the feature of agile of being iterative is most popular of all.

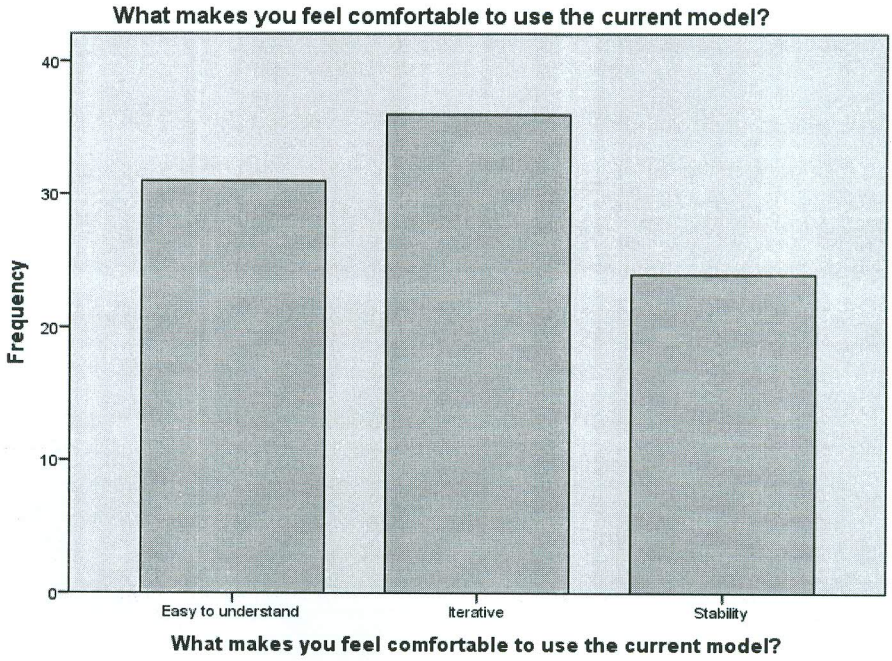


Table 4.1.10 (b)

This bar-chart depicts the same information as table 4.1.10 (a), but the graphically nature of the bar-chart makes it easy to understand things.

4.1.11 How does your model help in managing project?

How does the model you chose helps in managing the project?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Improved communication with customer	30	26.8	33.0	33.0
	Reduction of effort	6	5.4	6.6	39.6
	Improved quality of deliverable system	20	17.9	22.0	61.5
	Improved team morale	3	2.7	3.3	64.8
	Fast development	9	8.0	9.9	74.7
	More productive team	11	9.8	12.1	86.8
	Better control of development Schedule	12	10.7	13.2	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.11 (a)

This table indicates the few characteristics of model that helps managing the project. Out of 91 projects 30 respondents agreed that the model improved communication with client, 6 respondent consider that model had reduced the effort of team, 20 respondent considered that model has increased the quality of delivery system, 3 considered that model helped to improve the team morale, 9 considered model led to fast development, 11 respondent considered model led to more productive team and 12 respondent considered model helped in better control of development schedule

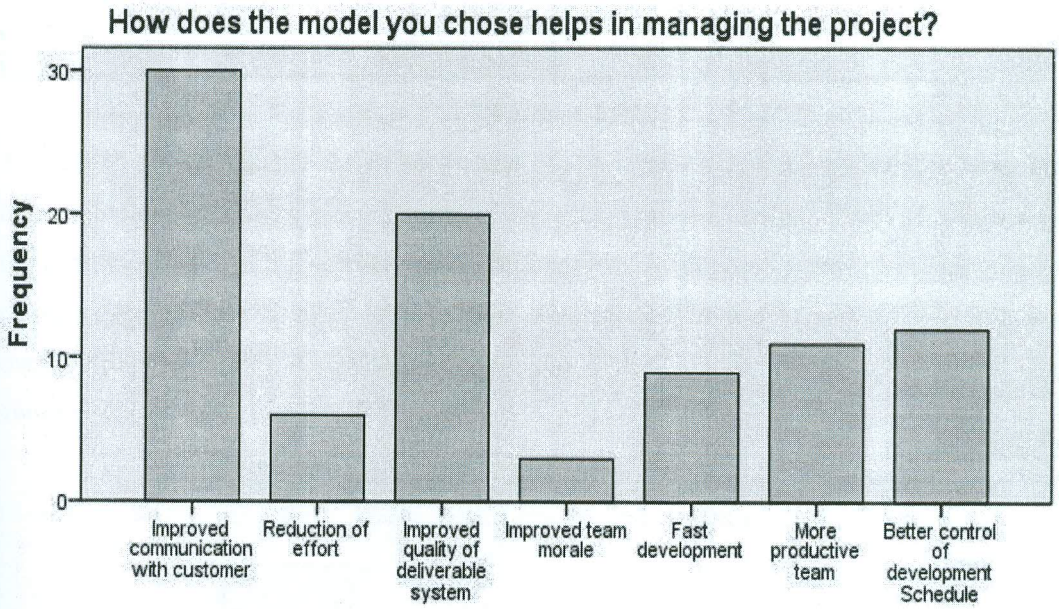


Table 4.1.11 (b)

This bar-chart clearly shows that improved communication with client is the key point where maximum respondents agree that the model has helped them in. Bar-chart depicts the same information, but the graphically nature of the bar-chart helps to understand it easily.

4.1.12 How your model does helps in avoiding cost overruns?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Timely deliverable	36	32.1	39.6	39.6
	better resource utilization	14	12.5	15.4	54.9
	Better project planning	32	28.6	35.2	90.1
	Excellent scheduling	9	8.0	9.9	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.12 (a)

This table shows that the factor or feature of the model that the respondents have been using which helped them in avoiding cost overruns. Out of 91 respondents, 36 claimed that the timely delivery of the deliverable made it easy to avoid cost overrun, 14 claimed better resource utilization, 32 were of view that better project planning saved them from cost overrun and remaining 9 respondents considered excellent scheduling that helped them avoid cost overrun.

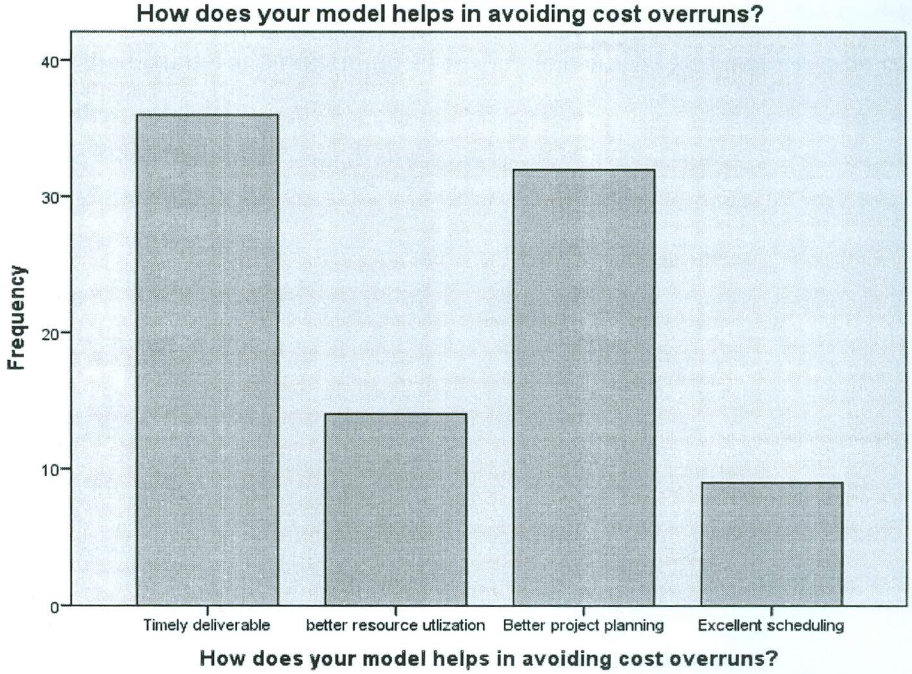


Table 4.1.12 (b)

The bar-chart displays the same information as table 4.1.12 (a), but the graphically nature of the bar-chart makes it easy to understand.

4.1.13 What is the effect on product quality with the chosen model?

What is the effect on product quality with the chosen model?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very low	3	2.7	3.3	3.3
	Low	3	2.7	3.3	6.6
	Medium	41	36.6	45.1	51.6
	High	33	29.5	36.3	87.9
	Very High	11	9.8	12.1	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.13 (a)

This table shows that the quality parameter associated with the model has an effect on project. The numeric values in the table clearly shows there were only 3 respondents who think that the quality was very low, 3 respondent consider that the quality remained low, 41 respondent considered quality to be medium, 33 respondent considered quality to be high by using their model and remaining 11 respondent considered quality to be very high by the use of their model.

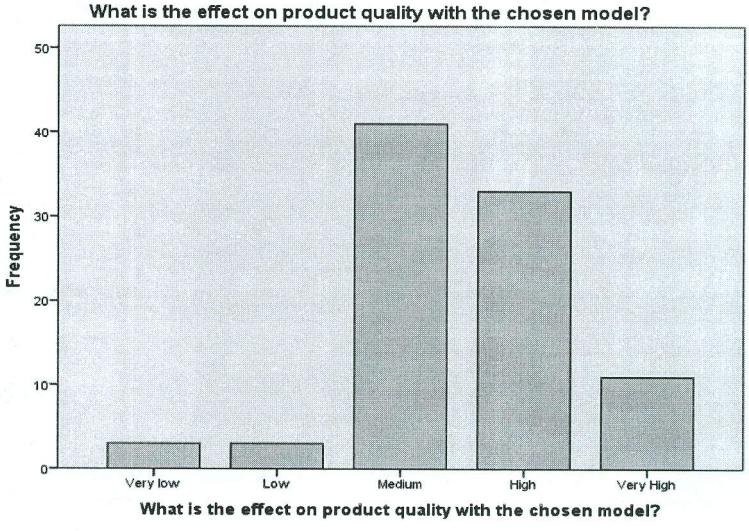


Table 4.1.13 (b)

The above bar-chart indicates the same as table 4.1.13 (a), but due to graphically nature of the bar chart it is easy to understand.

4.1.14 How much comfortable are you in your current model?

How much comfortable are you in your current model

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very low	1	.9	.9	.9
	Low	5	4.5	4.5	5.4
	Medium	45	40.2	40.2	45.5
	High	53	47.3	47.3	92.9
	Very High	8	7.1	7.1	100.0
	Total	112	100.0	100.0	

Table 4.1.14 (a)

The table indicates, the respondent's comfort level with the model being used in the project. The numeric figures clearly shows that out of 112 respondents only 1 respondent consider his/her comfort level to very low, 5 respondents considered their comfort level to low, 45 considered their comfort level to medium with the model, 53 respondents considered their comfort level to be high with the type of model they have been using their project and remaining 8 respondents considered their comfort level to be very high with the model.

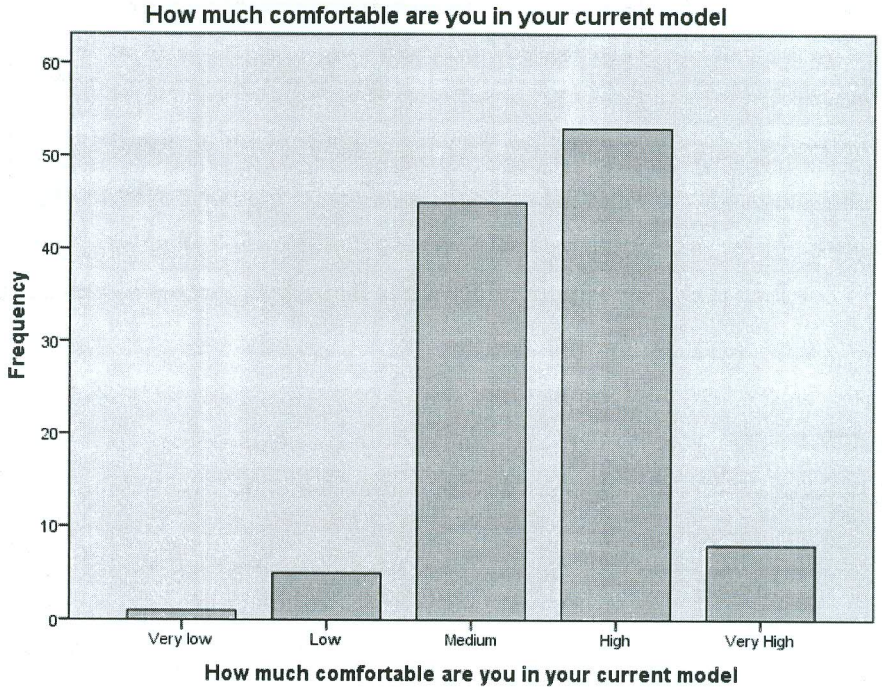


Table 4.1.14 (b)

The above bar-chart depicts the same information as in table 4.1.14 (a), but the graphically nature of the bar-chart makes it easy to understand.

4.1.15 How well defined were the requirements defined?

How well defined were the requirements (business and user requirements) when programming began at the start of the project?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Not defined	2	1.8	2.2	2.2
	Minimal requirements were provided	8	7.1	8.8	11.0
	Some requirements were provided	32	28.6	35.2	46.2
	Adequate requirements were provided	32	28.6	35.2	81.3
	Requirements were full defined	17	15.2	18.7	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.15 (a)

This table indicates the availability of requirements for the development team. Out of 91 projects only in 2 projects the requirements were not defined, 8 respondents consider the minimal requirements provided to them, 32 respondents considered some requirements were provided to them, 32 respondents adequate requirements were provided and remaining 17 respondents considered that the requirements were fully defined.

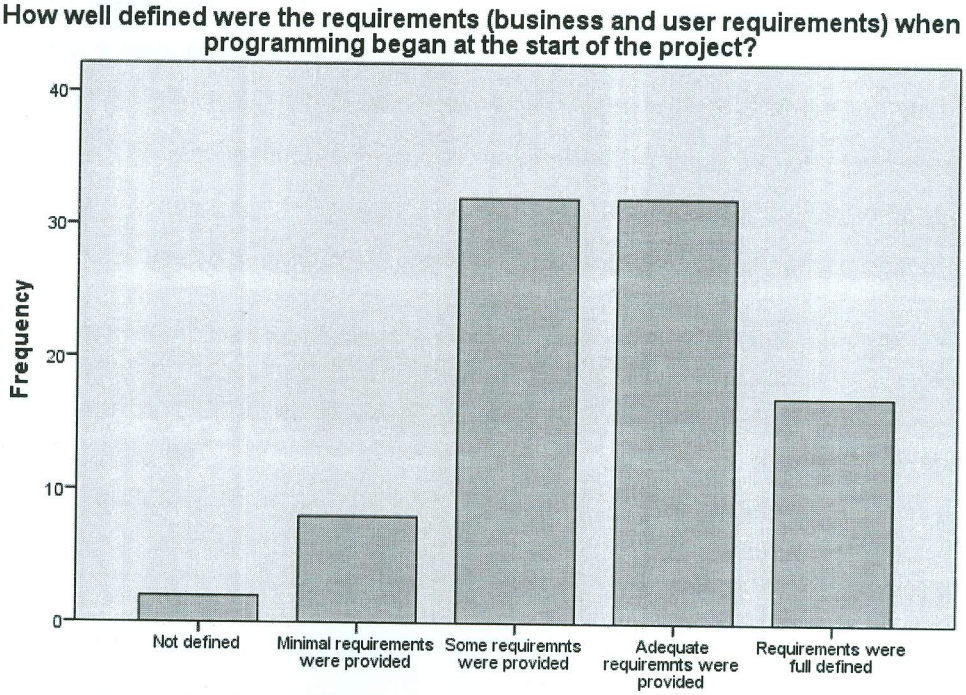


Table 4.1.15 (b)

The above bar-chart translates the same information as table 4.1.15 (a), but the graphically nature of the bar-chart makes it easy to understand.

4.1.16 How many request for changes to existing functional requirements?

How many request for changes to existing functional requirements have you received in the past two weeks?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very low	6	5.4	6.6	6.6
	Low	23	20.5	25.3	31.9
	Medium	39	34.8	42.9	74.7
	High	22	19.6	24.2	98.9
	Very High	1	.9	1.1	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.16 (a)

The above table indicates the change request for existing functionality. Out of 91 respondents, 6 respondents have experienced a very low amount of change in requirements for existing functionality, 23 respondents experienced a low amount of change in requirements, 39 respondents experienced a medium amount of change in existing functionality, 22 respondents experienced a high amount of change in functionality and 1 respondent experienced very high change in the existing functionality of the system.

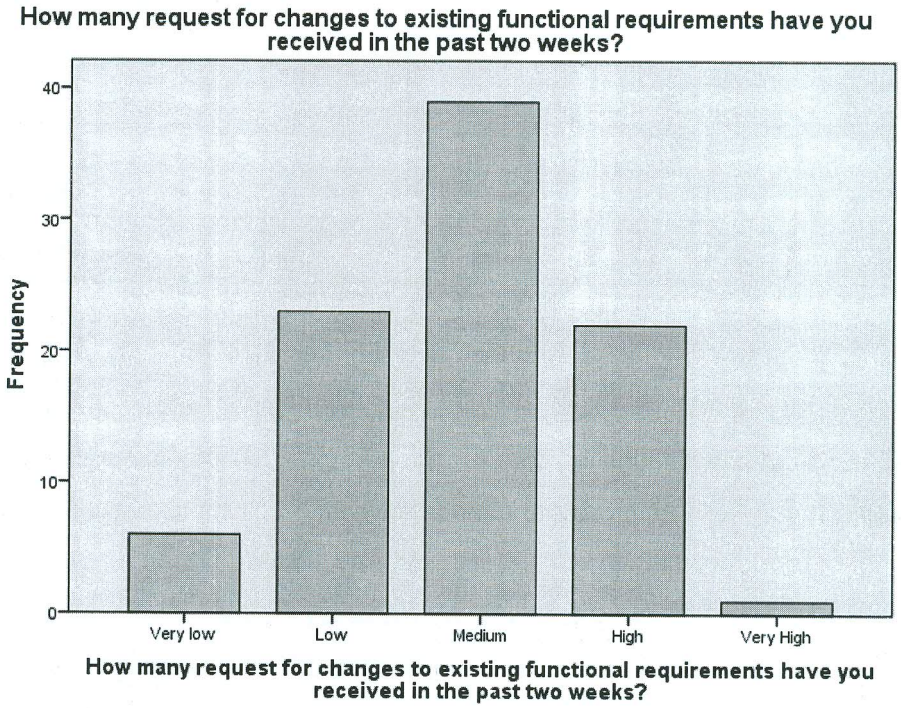


Table 4.1.16 (b)

The bar-chart depicts the same information as table 4.1.16 (a), but the graphically nature of the bar-chart makes it easy to understand.

4.1.17 How many request for changes to other project factors?

How many request for changes to other project factors have you received in the past two weeks?

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Very low	16	14.3	17.6	17.6
	Low	33	29.5	36.3	53.8
	Medium	34	30.4	37.4	91.2
	High	7	6.3	7.7	98.9
	Very High	1	.9	1.1	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4.1.17 (a)

The above table describes that the change in other project factors other than the functional requirements of system being developed, like for instance, deployment date, budget, resources changes and etc. out of 91 projects, 16 projects have a very low rate i.e. hardly 1 – 10 request have been made, of changes of other project factors, 33 projects have a low rate i.e. in between 11 – 20 request have been made, 34 projects have a medium rate i.e. 21 – 30 request have been made, 7 projects have a high rate i.e. 31 – 40 request have been made and only 1 project out of 91 showed a very high rate i.e. 41 – 50 request have been made for change in other project factor. From this statistics, it is clear that in agile most of the projects have a low to medium change in the other project factors.

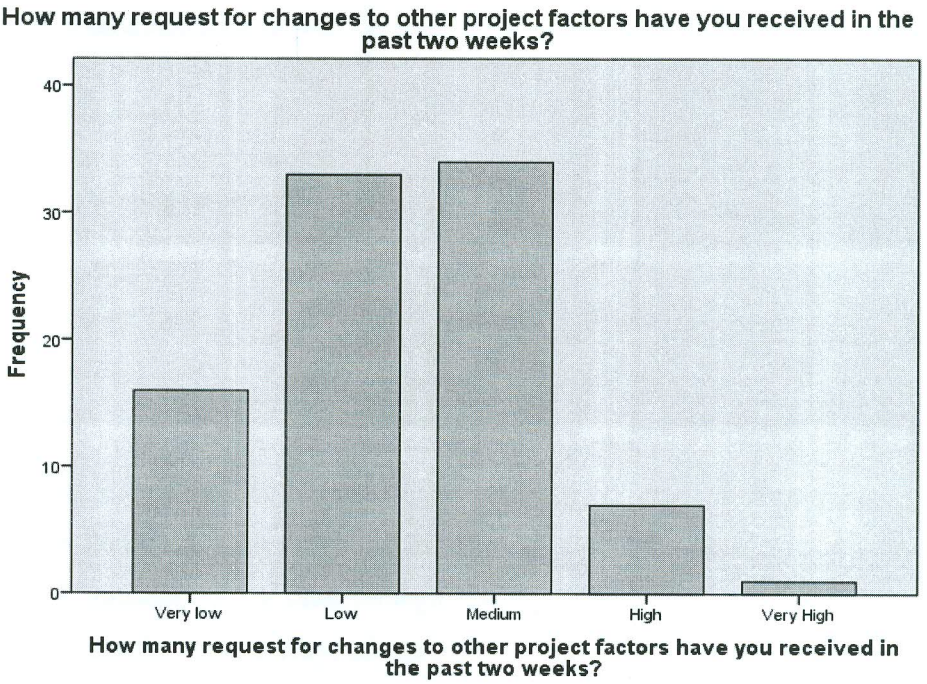


Table 4.1.17 (b)

The above bar charts depicts the same information as table 4.1.17 (a), but the graphically nature of the bar chart makes it easy to understand.

4.1.18 Disadvantages of using agile method

In your opinion what are the disadvantages of using agile method

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Reduce design document	24	21.4	26.4	26.4
	Gaining the support of the management for agile techniques are difficult	8	7.1	8.8	35.2
	Some techniques are not acceptable to the team members	5	4.5	5.5	40.7
	Controlling iteration is difficult	17	15.2	18.7	59.3
	Requirement change is difficult to handle	16	14.3	17.6	76.9

	Clients want development to meet a fixed price contract	10	8.9	11.0	87.9
	Clients cannot prioritize requirements	5	4.5	5.5	93.4
	Clients are not prepared to drop minor requirements when new requirements emerge	6	5.4	6.6	100.0
	Total	91	81.3	100.0	
Missing	System	21	18.8		
Total		112	100.0		

Table 4. 1.18 (a)

The above table shows a list of disadvantages of using agile models. Out of 91 respondents, 24 consider that the disadvantage of using agile models is that it reduce the design document, 8 consider that the disadvantage of using agile method is gaining the support of the management for agile, 5 respondents consider some techniques are not acceptable to the team members as disadvantages of implementing agile, 17 considered controlling the iteration of agile models is the disadvantage of using agile, 16 considered requirements change is difficult to handle as the disadvantage of using agile model, 10 considered clients want development to meet a fixed price contract as disadvantage of using agile model, 5 respondents marked clients cannot prioritize requirements, also the reason might be lack of technical how know of the client, 6 respondents considered that clients are not prepared to drop minor requirements when new requirements merge as disadvantage of using agile model.

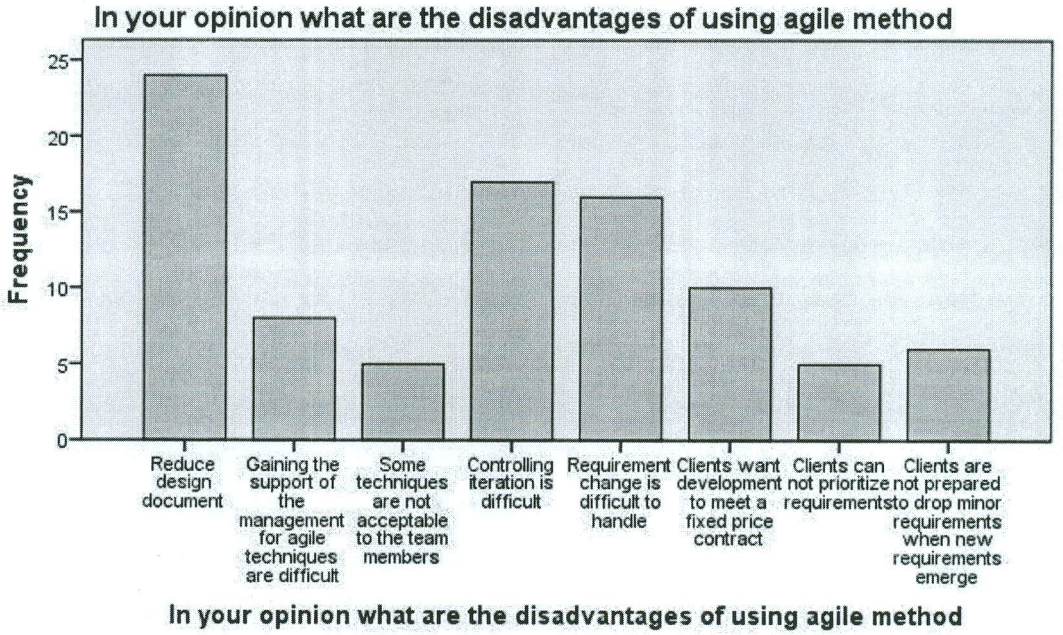


Table 4.1.18 (b)

The above bar chart displays the same information as that is mentioned in table 4.1.18 (a), but the graphically nature of bar chart makes it easy to understand.

4.1.19 Studying the relation between lines of code and use of agile model used

Count

		Are you applying any Agile Model in current project			Total
		No	Yes, but I have worked in my past project	Yes	
Your current Project contains, estimate, how many LOC (Lines of Code)	Less than 10000	1	0	8	9
	10000 - 50000	9	7	15	31
	More than 50000	11	11	50	72
Total		21	18	73	112

Table 4.1.19 (a)

The above table shows the relationship between the use of agile model and LOC (Lines of Code). The numeric values make it easy to understand. Total of 9 projects are such that their line of code (LOC) is less than 10,000 lines out of which 1 project whose LOC is less

than 10KLOC do not use any agile model and 8 respondents have still used agile model in small project. Which makes a total of around 88.8% small projects are using agile models. Similarly, total of 31 projects whose lines of code (LOC) ranges from 10KLOC to 50KLOC amongst which there were only 9 projects which were not using any agile model in the project. 22 respondents did used agile model in such projects which LOC ranged from 10,000 to 50,000 LOC. This makes a total of around 71% of the projects which are using agile model which 10KLOC to 50KLOC. Similarly, total of 72 projects were analyzed and agreed to have more than 50KLOC or 50,000 LOC or lines of code. Out of 72 projects, 11 projects did not use any agile model in their project, and remaining 61 projects used somehow the other used agile models. The range of lines of code which was more than 50KLOC makes the project large and complex. This means that around 85% of projects of software industry, which are large and complex projects i.e. their lines of code of are more than 50 KLOC, are using different agile as well as traditional models for handling of the projects.

Overall, from the above statistics it is very clear that no matter the project is small i.e. the lines of code is less than 10KLOC, or the project is medium sized project i.e. the lines of code ranges from 10KLOC to 50KLOC, or the project is large i.e. the lines of code exceeds 50KLOC and more, all uses agile models to take good control of the projects right from the kick off meeting. This also shows that in every size related project

4.1.20 Studying the relation between designation and agile model used

Please specify your designation * Which Model from following Agile Model do you feel comfortable in your project Cross tabulation

Count

		Which Model from following Agile Model do you feel comfortable in your project			Total
		Scrum	Extreme Programing	Adaptive Software Development	
Please specify your designation	General Manager	1	0	1	2
	Senior Manager	6	0	1	7
	Project Manager	14	0	2	16
	Team Leader	14	4	8	26

	Developer	10	6	9	25
	Quality Assurance	5	2	2	9
	Designer	1	0	0	1
	Business Analyst	3	2	0	5
Total		54	14	23	91

Table 4.1.20

The above table indicates that the relationship between the model used and the designation. Out of 91 total projects, there were 2 General Managers, 7 Senior Managers, 16 Project Managers, 26 Team Leader, 25 Developers, 9 Quality Assurance, 1 Designer and 5 Business Analyst. Out of 2 General Managers 1 opted for Scrum and other opted for Adaptive Software Development, out of 7 General Managers, 6 opted for Scrum and 1 opted for Adaptive Software Development, out of 16 Project Managers, 14 opted for Scrum and 2 opted for Adaptive Software Development, out of 26 Team Leads 14 opted for Scrum, 8 opted for Extreme Programming and 8 opted for Adaptive Software Development. Out of 25 Developers, 10 opted for Scrum, 6 opted for Extreme Programming and 9 opted for Adaptive Software Development. Out of 9 Quality Assurance 5 opted for Scrum, 2opted for Extreme Programming and 2 opted for Adaptive Software Development. In case of designer, no of sample size was one in which designer opted for Scrum. Last but not the least, out of 5 Business Analyst 3 opted for Scrum and 2 opted for Extreme Programming.

4.1.21 Studying the relation between model used and the avoiding cost overruns.

		Which Model from following Agile Model do you feel comfortable in your project			Total
		Scrum	Extreme Programing	Adaptive Software Development	
How your model does helps in avoiding cost overruns?	Timely deliverable	24	5	7	36
	better resource utilization	8	3	3	14
	Better project planning	18	5	9	32
	Excellent scheduling	4	1	4	9
Total		54	14	23	91

Table 4.1.21

The above table shows the relation between the model used and the how the model helped in avoiding cost overruns. Total of 91 projects were found matching the criteria out of which 56 applied Scrum, 14 applied Extreme Programming and 23 applied Adaptive Software Development. Out of 54 which applied scrum, 24 opted for timely deliverable as an option for avoiding cost overrun, 8 opted better resource utilization, 18 opted for better project planning and remaining 4 opted for excellent scheduling. Similarly, out of 14 which applied Extreme Programming 5 opted for timely deliverable as an option for avoiding cost overrun, 3 opted better resource utilization, 5 opted for better project planning and remaining 1 opted for excellent scheduling. Similarly, out of 23 which applied Adaptive Software Development 7 opted for timely deliverable as an option for avoiding cost overrun, 3 opted better resource utilization, 9 opted for better project planning and remaining 4 opted for excellent scheduling. This means that timely deliverable is the highest rated option.

4.1.22 Studying the relation between model used and the managing project.

		Which Model from following Agile Model do you feel comfortable in your project			Total
		Scrum	Extreme Programing	Adaptive Software Development	
How does the model you chose helps in managing the project?	Improved communication with customer	21	5	4	30
	Reduction of effort	3	0	3	6
	Improved quality of deliverable system	10	1	9	20
	Improved team morale	1	2	0	3
	Fast development	3	2	4	9
	More productive team	8	1	2	11
	Better control of development Schedule	8	3	1	12
Total		54	14	23	91

Table 4.1.22

The above table indicates the relationship between the agile model applied and the how does the model help in managing the project as a whole. Out of 54 project which

were applying scrum 21 reported that increased communication with customer helped in managing the project as a whole, 3 opted that the model applied had reduced the effort, 10 opted that the model used has increased the quality of deliverable system, 1opted that the model has improved the team morale, 3 opted that the model helped them in fast development of the product, 8 claimed teams were more productive and remaining 8 opted for better control of development schedule. Out of 14 project which were applying extreme programming 5 reported that increased communication with customer helped in managing the project as a whole, 0 opted that the model applied had reduced the effort, 1 opted that the model used has increased the quality of deliverable system, 2 opted that the model has improved the team morale, 2 opted that the model helped them in fast development of the product, 1 claimed teams were more productive and remaining 3 opted for better control of development schedule. Out of 23 project which were applying adaptive software development 4 reported that increased communication with customer helped in managing the project as a whole, 3 opted that the model applied had reduced the effort, 9 opted that the model used has increased the quality of deliverable system, 0 opted that the model has improved the team morale, 4 opted that the model helped them in fast development of the product, 2 claimed teams were more productive and remaining 1 opted for better control of development schedule.

4.1.23 Studying the relation between model used and the effect on quality.

		Which Model from following Agile Model do you feel comfortable in your project			Total
		Scrum	Extreme Programing	Adaptive Software Development	
What is the effect on product quality with the chosen model?	Very low	2	0	1	3
	Low	2	0	1	3
	Medium	23	9	9	41
	High	21	3	9	33
	Very High	6	2	3	11
Total		54	14	23	91

Table 4.1.23

The table above indicates the model applied and the effect of the model on software quality. Out of 54 projects which were applying Scrum, 2 opted for very low quality has been achieved, 2 opted for low quality with the model that has been applied, 23 claimed to be medium quality achieved with the model, 21 opted that high quality can be achieved and remaining 6 opted for very high quality achieved by applying that specific model. Out of 14 projects which were applying Extreme Programming, 0 opted for very low quality has been achieved, 0 opted for low quality with the model that has been applied, 9 claimed to be medium quality achieved with the model, 3 opted that high quality can be achieved and remaining 2 opted for very high quality achieved by applying that specific model. Out of 23 projects which were applying Adaptive Software Development, 1 opted for very low quality has been achieved, 1 opted for low quality with the model that has been applied, 9 claimed to be medium quality achieved with the model, 9 opted that high quality can be achieved and remaining 3 opted for very high quality achieved by applying that specific model.

Chapter # 5

Results and Discussion

5. Results and Discussion

5.1 Discussion

The aim of this research was to identify the relationship between the project's nature and the agile model used. Different models like Scrum, Extreme Programming and Adaptive Software development and Dynamic System Development approaches has been in continuous progress.

Different models have their own property, their own advantages and their own disadvantages. Though the above mentioned all models are agile and light weight models but yet if a wrong model is applied which does not match the project's nature, the chance of fail for the project increases.

This study relies on finding a reasonable relationship between the project's nature and the type of model used. The factor which will be describing the project nature is the complexity of the project. By complexity I simply mean, the lines of code are above 10,000 and the size of development team is either more than or equals to 3.

Overall, out of 91 projects that were analyzed and evaluated, which fulfilled the criteria of complex project, 54 projects was applying Scrum, i.e. 59% approximately, and 14 projects applied Extreme Programming, i.e. 15% approximately, and the remaining 23 projects were applying Adaptive Software Development, i.e. 25% approximately.

The two major variables, the size of the development team and the lines of code have an impact on the model applied by the team. This relation was identified and results were recorded in table 4.1.5 (page 34).

5.2 Conclusion

To conclude this study and to sum this up as a whole, it is pointed out that a very little work has been done in this regard as the conventional school of thought still thinks that the usage of agile models for large and complex projects is not good enough, as agile models provide a very little documentation. New trends and dimensions have been opened as industry has taken a step forward towards agile for small as well as for large and complex projects. Out of 112, the total sample size, 91 projects which matched the criteria of being complex and heavy were using agile model, which shows that out of total 100, 81 project those were large and complex were applying agile model.

Table 4.1.21, table 4.1.22 and table 4.1.23, clearly indicates that the model being used by most of the practitioner is Scrum and reason can also be reviewed in the table as mentioned above. Scrum being the highest users is at top, Adaptive Software Development whose user are 23 rests at 2nd position and Extreme Programming with a user strength of just 14 remains at 3rd position.

Reference:

Abrahamsson, P., Salo, O. & Ronkainen, J. (2002). *Agile software development methods, review and analysis*

Abrahamsson, P., Warsta, J., Siponen, T., & Ronkainen, J. (2003). *New Directions on Agile Methods: A Comparative Analysis*

Ambler, S., (2005). *When Does (n't) Agile Model Make Sense.*

Awad, A. (2005). *A Comparison between Agile and Traditional Software Development Methodologies*

Beck, K. (1999). *Embracing change with Extreme Programming*

Beck, K. (2000). *Extreme Programming explained: Embrace Change*

Boehm, B. (2002). *Get Ready for Agile Methods, with Care.*

Charette, R. (2001). *The decision is in: Agile versus heavy methodologies.*

Cockburn, A. (2002). *Agile Software Development.*

Copeland, L., (2001). *Extreme Programming moves slowly into the Enterprise.*

Cunningham. W., (2004). *Agile Manifesto. http://www.agilemanifesto.org/*

Dyba, T., Dingsoyr, T. (2008). *Empirical studies of agile software development: A systemic review*

Erickson, J., Lyytinen, K., & Siau, K., (2005). *Agile Modeling, Agile Software Development and Extreme Programming: The State of Research.*

Highsmith, J. (2002). *Agile Software Development ecosystem*

Highsmith, J., Cockburn, A. (2001). *Agile Software Development: The Business of Innovation.*

Highsmith, J. A., *Adaptive Software Development: A Collaborative Approach to Managing Complex System*

Huo, M., Verner, J., Zhu, L & Baber, A. (2004). *Software Quality and Agile Methods*

Krebs, J. (2005). RUP in the dialogue with Scrum. *developersWorks. http://www-106.ibm.com/developerworks/rational/library/feb05/krebs/index.html*

Lassenius, C. (2008). *Distributed Agile Development: Using Scrum in large project*

Mar, K., & Schwaber, K. (2005). *Experiences of using Scrum with XP.*

Appendix – Questionnaire

Comparison of different agile methodologies and determining which is more effective for heavy and complex projects of software industry

1. Please specify your name.

   _____

2. Please specify your designation
   a. General Manager
   b. Senior Manager
   c. Project Manager
   d. Team Leader
   e. Developer
   f. Quality Assurance

3. Please Specify your email or Phone

   _____

4. Your Work experience
   a. Less than 5 Year
   b. 5 - 7 Years
   c. 7 - 9 Years
   d. More than 9 Years

5. Your current Project team size
   a. 1
   b. 2
   c. 3
   d. more than 3

6. Your current Project contains, estimate, how many LOC (Lines of Code)
   a. Less than 10000
   b. 10000 – 50000
   c. More than 50000

7. How much comfortable are you in your current model
   1 = very low, 2 = low, 3 = medium, 4 = high, 5 = very high
   a. Very low
   b. Low

c. Medium
d. High
e. Very High

8. Are you applying any Agile Model in current project
   a. Yes
   b. No
   c. Yes, but I have worked in my past project

9. Which Model from following Agile Model do you feel comfortable in your project
   a. Extreme Programming
   b. Scrum
   c. Adaptive Software Development
   d. Feature Driven Development
   e. Dynamic System Development Method

10. How well defined were the requirements (business and user requirements) when programming began at the start of the project?
    a. Not defined; there were no stated requirements
    b. Minimal requirements were provided; there was no detail
    c. Some requirements were provided; they were not enough to carry the whole project
    d. Adequate requirements were provided; they were adequate to carry out most of the project
    e. Requirements were fully defined before the software development began

11. How many request for changes to new functional requirements have you received in the past two weeks?
    1-10 = Very low, 11-20 = Low, 21-30 = Medium, 31-40 = High, 41-50 = Very High
    a. Very low
    b. Low
    c. Medium
    d. High
    e. Very High

12. How many request for changes to existing functional requirements have you received in the past two weeks?
    Like: use of new technology or database; 1-10 = Very low, 11-20 = Low, 21-30 = Medium, 31-40 = High, 41-50 = Very High
    a. Very low
    b. Low

c. Medium
d. High
e. Very High

13. How many request for changes to other project factors have you received in the past two weeks?
Like: delivery date, budget, staff changes; 1-10 = Very low, 11-20 = Low, 21-30 = Medium, 31-40 = High, 41-50 = Very High
    a. Very low
    b. Low
    c. Medium
    d. High
    e. Very High

14. What makes you feel comfortable to use the current model?
Like: helps in communication, requirements change, etc.
    a. Easy to Understand
    b. Iterative
    c. Stability

15. How does the model you chose helps in managing the project?
    a. Improved communication with customer
    b. Reduction of effort
    c. Improved quality of deliverable system
    d. Improved team morale
    e. Fast development
    f. More productive team
    g. Better control of development Schedule

16. How your model does helps in avoiding cost overruns?
    a. Timely deliverable
    b. better resource utilization
    c. Better project planning
    d. Excellent scheduling

17. What is the effect on product quality with the chosen model?
1 = Very low, 2 = Low, 3 = Medium, 4 = High, 5 = Very High
    a. Very low
    b. Low
    c. Medium
    d. High
    e. Very high

18. In your opinion what are the disadvantages of using agile method
    a. Reduce design documentation creates problem
    b. Gaining the support of management for agile techniques is difficult
    c. Some techniques are not acceptable to the team members
    d. Controlling iterations is difficult
    e. Requirement changes are difficult to control
    f. Clients want the development to meet a fixed price contract
    g. Clients cannot prioritize requirements
    h. Clients are not prepared to drop minor requirements when new requirements emerge