Master Thesis
Software Engineering
Thesis no: MSE-2008:15
August 2008

Challenges in understanding software requirements in agile based offshore development

Muhammad Omair

University advisor(s):
Bengt Carlsson
Department name

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

This thesis is submitted to the School of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Muhammad Omair
E-mail: khanomair82@hotmail.com

University advisor(s):
Dr. Bengt Carlsson
School of Engineering

School of Engineering
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Internet : www.bth.se/tek
Phone : +46 457 38 50 00
Fax : + 46 457 271 25

ABSTRACT

Agile based development seems to become a favorable model for offshore development. It allows both on and offshore team to work in small iterations minimizing the effect of change in software requirements and at the same time developing regular communication between them. However different factors such as physical distance and lack of communication between on and offshore team becomes a hurdle between them leading to misunderstandings about software requirements.

This research work gives an insight about these challenges from the software industry by presenting and discussing the responses of four software companies located in different countries, collected through an online questionnaire.

The authors found that lack of communication between on and offshore site is seen as a major challenge for better understanding of software requirements. Shorter iterations at the offshore site require more communication with the onshore site. The language problem seems to exist only when both on and offshore site who are non-English speakers communicate in English. Regular long distance meetings would help in better understanding of software requirements. Previous domain and product knowledge is helpful in better understanding of software requirements. This research work would allow different stakeholders within agile based on/offshore setting to better understand these challenges and deal accordingly with them.

Keywords: lack of communication, knowledge sharing, physical distance, iteration duration.

ACKNOWLEDGEMENTS

I am thankful to Allah for giving me the ability to finish my thesis. Without His blessings I would not have been able to finish this research work.
I am extremely thankful to my advisor Bengt Carlsson who had put a lot of effort into this thesis. He had been extremely patient and provided invaluable feedback and support during every step of this thesis work. I would also thank Dejan Baca of Ericsson who gave us his precious time to get responses from Ericsson.
Lastly I would thank my family who had been supportive during my studies and had been the key source of motivation.

CONTENTS

1 INTRODUCTION

This section provides an introduction to the thesis document. In section 1.1, background to the study is given providing motivation for the research area chosen. The aims and objectives are discussed in section 1.3. The research questions and expected outcomes are given in section 1.4 and 1.5 respectively. Section 1.5 briefly describes research methodology. Section 1.6 provides outline of the rest of the thesis document.

1.1 Background

In software development, there are multiple sources of requirements that are both internal (team members) and external (customers, partners) to the software project. Software requirements could be generated from marketing department during a competitor analysis or a market survey to project managers, testers or developers who had been working on the product for some time [4] resulting in continuous flow of requirements [3]. To support this continuous flow, a software development methodology was needed that could easily satisfy these demanding needs. Agile is the best answer where you work in small iterations and deliver working software at the end of iteration [5]. The diverse collection of customers along with tough competition with its competitors puts a lot of pressure on the software requirements of a particular software product. Agile methodologies are very helpful in this regard as it welcomes changes, especially late changes [4].
The core concept in agile development is that the team can be increasingly effective when the time and cost of moving information and decisions between people is reduced [4]. For this purpose the development team should be co-located thereby increasing communication. Also included are close collaboration with customer and other stakeholders. Another aspect of agile is decrease in documentation and relying only on just enough documentation [5].
Then there is another concept of offshore software development where your team is distributed in different geographical locations taking advantage of cheaper labour, facilities and talented workforce [2]. Within global software development the activities are divided into two areas: onshore activities and offshore activities as shown in the table 1 taken from [2].

Onshore activities	Offshore activities
Project inception	Design
Requirements specification	Implementation
Requirements change management	Testing
Analysis	Maintenance
Integration Testing	Change Implementation
Deployment	
Project Monitoring	

Table 1: Typical division of activities between onshore and offshore sites taken from [2]

The biggest advantage of offshore development is reduced cost of development [7]. The significant challenge of offshore development is hurdles in communication between the onshore and offshore team. It is more difficult for the client and onshore team to communicate with the offshore project team because of increase in distance that leads to difficulty in face to face meeting (one of primary principles of agile) and also differences in time zones. Both of which can increase the probability of developing the incorrect functionality as misunderstandings and interpretations occur over software requirements [6]. Some of the main issues and challenges in global software development are: inadequate communication, knowledge management, cultural diversity, time differences [8]. These

issues become more evident in market driven software products where the source of requirements are not one or few individuals but many, making it difficult for the offshore team to understand software requirements.
Agile practices are successful in many of the software domains and projects. Their utilization in offshore projects could help achieve the advantages of agility (flexibility) and off shoring (maximize cost savings). However agile extremely focuses on communication and feedback thereby demanding a development team that is physically co-located and in close collaboration with its customers or other important stakeholders. It is not easily achievable in distributed setting [6].
Agile based offshore development methodologies are a relatively new research area as compared to plan driven offshore development methodologies [11] such as waterfall model. The focus of this research is on requirements phase because in agile this phase is iterative and therefore lot of communication is required between on and offshore team. The main challenges in agile based offshore development are lack of communication and documentation regarding software requirements. Most of the research work had focused on these challenges but does not provide deep analysis and different perspectives such as system analyst/developer, on/offshore perspective. This research work tried to analyze these perspectives and also identify some new insights into these challenges by having open ended question in the online questionnaire survey. This thesis discusses challenges and their solutions but more focus is on former.

1.2 Aims and objectives

The main objective of this thesis is to identify challenges in understanding software requirements in agile based offshore setting to decrease overall software development time by providing solutions for them. Following objectives are defined to achieve this goal:

- Identify challenges in understanding software requirements in agile based offshore development through literature review.
- Get response from the industry about these challenges and indentify new ones.
- Provide solutions for these challenges.

1.3 Research Questions

The research questions are given below.

1. What are the challenges in understanding software requirements in agile based offshore development?
2. How can these challenges be resolved to improve understanding software requirements in agile base offshore development?

1.4 Expected outcomes

Expected outcome is given below:

- An understanding and explanation of different challenges affecting understanding software requirements in agile based on/off shore setting.
- Providing solutions for the challenges.

1.5 Research Methodology

State of the art literature related to the research questions was studied. This literature survey gave us the latest knowledge about agile based offshore software development. This literature survey was studied from credible sources that include the following:

- Electronic databases
- Books
- Journals
- Conferences
- The internet

Literature review was used only to get knowledge about this research area. It formed the basis for conducting an online questionnaire survey at four software companies on challenges related to understanding software requirements. This survey gave us a hands-on knowledge about these challenges within these software companies and also answering the research questions by using quantitative research method. The solutions are discussed along with the challenges in the discussion section as they are connected to them. Figure 1 below gives an overview of the research process.

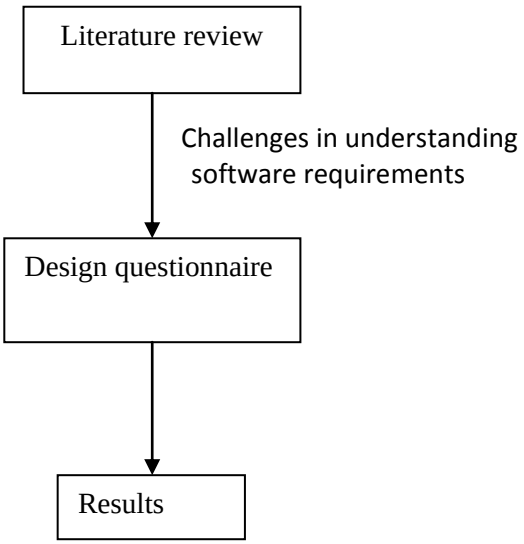

Figure1: Research methodology

1.6 Thesis Outline

This section gives an outline of the thesis document.
In Chapter 2 agile software development is explained. A discussion is made about agile principles and why should agile methodologies be used focusing on its different features such as adaptability, welcoming change, iterative development and strong emphasis on communication.
In Chapter 3 an introduction to requirements engineering process is given and what parts of the agile methodologies are similar to this process.
Chapter 4 discusses agile based offshore development. Many companies are moving offshore and are finding it difficult to manage a geographically distributed project team. This chapter advocates using agile process in this on/offshore setting
Chapter 5 highlights different challenges in understanding software requirements in agile based on/offshore setting. The three most important challenges are lack of communication, less detailed software requirements specification and lack of domain and product knowledge.
In chapter 6 a questionnaire is designed based on challenges identified in Chapter 5. This questionnaire is then made available online to be filled in by four software companies.
Chapter 7 gives an analysis of data collected through the online questionnaire. A graphical presentation of the data is provided to allow the reader to understand in a much easier way.

In Chapter 8 collected data is discussed in detail. This discussion involves of making different groups of respondents to see if there are any similarities or differences in their answers.

2 AGILE SOFTWARE DEVELOPMENT

This chapter discusses what is agile. How does it work? What are its advantages over other development methodologies?

2.1 What is Agile?

Agility can be defined as "Agility is dynamic, context-specific, aggressively change embracing, and growth-oriented. It is not about improving efficiency, cutting costs, or battening down the business hatches to ride out fearsome competitive "storms." It is about succeeding and about winning: about succeeding in emerging competitive arena, and about winning profits, market share, and customers in the very center of the competitive storms many companies now fear" [cited by 28]. The core concept in agile is quick response to change [6]. Any methodology should be quicker and cheaper to implement changes in requirements where you cannot lock or freeze requirements in earlier stages [41]. The focus of agile is always on the team and concludes that to effectively respond to change one have to reduce the cost of moving information between people along with reduction in the elapsed time between making a decision to seeing the outcomes of that decision [6]. Below in Figure 2 are the values of agile manifesto and waterfall model taken from [5].

Agile	**Waterfall**
Individuals and interactions	processes and tools
Working software	comprehensive documentation
Customer collaboration	contract negotiation
Responding to change	following a plan

Figure 2. Values of agile manifesto and water fall model

These values form basis for the principles given below in figure 3 taken from [5]. Principles are a more detail descriptions of these values mentioned above. These principles should not be confused with concrete practices as each agile methodology adopts these principles according to its own priorities.

The values given in figure 2 show a comparison of agile with waterfall model. In waterfall model software development activities are divided into five distinct and linear stages [50]. It has some drawbacks such as very formal and inflexibility to changing software requirements [cited by 50]. Due to changing nature of software requirements waterfall model does not fully understand software requirements and resources are prematurely committed [44] leading to greater cost than estimated as the project progresses.

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Figure 3. Principles of Agile Manifesto

There are a number of agile methodologies following the above mentioned principles given in figure 3. These will not be discussed as every methodology has its own characteristics but only their names will be mentioned along with references that explain these methodologies. A more detailed description of different agile methodologies can be found at [12]. But it is important to note that each of them follows the above mentioned values and principles. These methodologies are:

i. Extreme Programming (XP) [47, 48, 49].
ii. Feature driven development (FDD) [40].
iii. Agile Modelling (AM) [38].
iv. Scrum [36].
v. Crystal Methodologies
vi. Dynamic Systems Development Method (DSDM)
vii. Adaptive Software Development (ASD)

2.2 Why Agile?

Requirements of software products are continuously changing due to its different nature [3] and diverse set of stakeholders compared to a fixed customer [4]. The diverse collection of customers along with tough competition with its competitors puts a lot of pressure on the requirements of a particular product. Agile methodologies are very helpful in this regard as it welcomes changes especially late changes [6].
The core concept in agile development is that the team can be increasingly effective when the time and cost of moving information and decisions between people is reduced [6]. For this purpose the development team should be co-located thereby increasing communication [7]. For example in case of XP if the development team (mainly programmers) is scattered in two rooms it creates problems for successfully implementing it [29]. Also included are close collaboration with customer and other stakeholders. Another aspect of agile is decrease in documentation and relying only on just enough documentation [7]. In software development there is a continuous flow of requirements. The aspect of welcoming late changing of software requirements in agile [6] makes it suitable for the demanding needs of software development. All of these aspects will be discussed in further detail below.

2.2.1 Adaptability

Agile methodologies can be blended with existing software development practices [7, 30]. However it is important to remember that this blending should be done with care [30]. By care it means that all the project or product members should be informed in detail about it. They should be convinced and brought onboard as it is important for any process change to succeed [31]. It is equally important to note that this adaptability gives agile its strength. So a project team can tailor the features of a particular agile method with their existing methods and use it effectively [30]. It gives freedom to management and other team members who are at higher level to focus on their core responsibilities, for example project manager will become more of a facilitator rather than a manager in strict sense. However it does not mean that managers and other members should ignore their responsibilities. They have to keep track of progress of adopted agile practices and their effect on project and project team. Thus they can focus on their other and sometimes main responsibilities. Since there are many sources of requirements therefore the world of shared responsibility and understanding [30] (its basic principle [5]) facilitate this phenomenon. The adaptability factor will be further discussed in the chapter of offshore development.

2.2.2 Welcoming change

In waterfall software development model the software requirements are frozen before moving to the next stage and following the same freezing principle when commencing to the next stage. Agile on the other hand allows the freedom of making changes in software requirements specification even late during the software development [41]. Satisfying the customer at delivery of the software and not during the initial phases has now taken precedence [28]. It means that requirements change during different phases of software development and thus requires flexibility and accommodating change. Requirements cannot be specified absolutely correctly in the specifications. It is a reality with which you have to deal with [37]. Therefore you have to deal with changes in one way or another. So it is better to welcome these changes instead of trying to put more than required effort in the analysis phase. A particular example is of market driven software products in which there are many sources of requirements meaning many requirements at most occasions that leads to continuous changing of requirements [3]. For this purpose agile methodologies would be very helpful as they welcome changes especially late changes in software requirements [5, 41].

2.2.3 Iterative development

Iterative development of agile methodologies helps in giving a better understanding of software requirements by the project team. During early iterations changes in software requirements occur that will require its reassessment. These early iterations will remove the ambiguities in software requirements and will minimize the chances of implementing software requirements that will prove costly when changing them later in the software development lifecycle [14]. It is particularly useful in offshore setting since these iterations help in understanding software requirements of the system. In this way the offshore team can be involved more in the overall process that will lead to increase in productivity. Thus agile will be very useful for software products where software requirements changes frequently. For this purpose iterative development can show the software requirements in a quantifiable manner for example in form of prototypes in early iterations [11]. If these software requirements are realized as not implementable or conflicting late in the development lifecycle then managing them would prove costly; the late a problem arises the more difficult and costly it becomes to manage.

2.2.4 Regular communication with end-user and other stakeholders

According to [44] software projects fail due to a number of reasons. Some of which are mentioned below:

i. Not clearly communicating the requirements
ii. Business problem is not solved by the requirements
iii. Changing nature of requirements
iv. Incorrect requirements
v. Committing resources before fully understating the requirements

For these issues to resolve in context of agile methods it requires regular communication with end-user and other stakeholders involved in the project [6]. It is important because there is less documentation and more face to face communication. This communication between different stakeholders brings them to a common understanding of software requirements. It becomes more important in case of market driven software products where you have many stakeholders and there is the issue of bringing them on a common understanding about software requirements. These different stakeholders have different backgrounds and thus interpret requirements in different ways which requires regular communication and feedback meetings. The technical knowledge of these stakeholders may not be at the same level making face-to-face communications more important for example a software requirement that comes from marketing department is more abstract and when is made concrete or understandable to the development team, its meaning and context may change. Thus marketing department then have to be brought into the communication process to confirm that these software requirements which are now in a new shape have maintained the same meaning. This activity cannot be performed without communicating with them. In case the technical team does not communicate this early on then it can create difficulties for the development team to accommodate these changes later on. By difficulties it is meant that increase in development time and resources spent on the product. In agile practices software requirements are more informal and kind of stories [43] that can be interpreted differently by differently people. This characteristic of informality in software requirements will be discussed in later chapter of challenges in understanding software requirements in agile based offshore development.

3 REQUIREMENTS ENGINEERING PROCESS (REP) AND AGILE SOFTWARE DEVELOPMENT

This chapter explains requirements engineering process and its similarities to agile methodologies. It would help readers in understanding the different aspects of agile and waterfall development methodologies. Waterfall development methodology was selected because it is most widely used for offshore development to avoid communication challenges. Kotonya and Sommerville [15] had given a complete and concise definition of REP "A requirements engineering is a structured set of activities which are followed to derive, validate and maintain a systems requirements document. Process activities include requirements elicitation, requirements analysis and negotiation and requirements validation". A simplified version of this process taken from [16] is given below in figure 4.

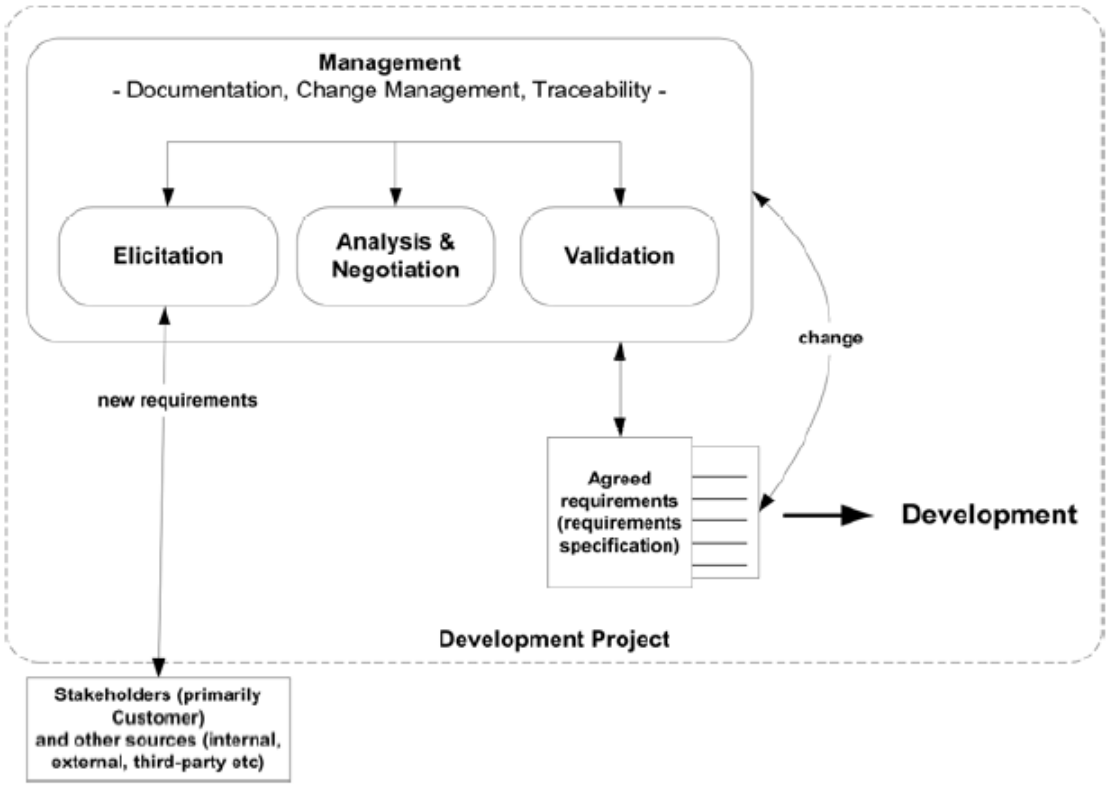

Figure 4. Overview of RE process activities taken from [16]

A brief and comprehensive understanding of these activities in general and agile practices is provided below.

3.1 Requirements Elicitation

Requirements Elicitation is about finding requirements through consultation and communication with different stakeholders, studying already existing system documents and using domain knowledge [15, 16, 18, 20]. In other words it also identifies the system boundaries [19] i.e. what the system should do and not so. Particular questions that need to be answered are: what activities come inside the scope of the system and that had to be put as software requirements and what activities are outside the scope of the system. This activity could be said as the first step in a RE process.

System Engineers and developers work in close coordination with customers and end users to understand what needs to be fulfilled by the software product [15]. There are other sources for eliciting software requirements including documentation, standards, external products that will be interacting with the software being developed, operating manuals of products to be replaced, laws and regulations etc [16]. It is important to note that elicitation is not an easy process. Requirements elicitation does not mean that requirements are just there to be collected but it includes a more complex activity of capturing requirements. Some stakeholders do not know how to express their requirements for the software to be developed. Therefore requirements sometimes have to be extracted rather than simply collecting them [17]. A few of the elicitation techniques are described below:

3.1.1 Conventional techniques

It consists of interviews, surveys and questionnaires to gather information from stakeholders. Also analyzing existing documentation for example standards and manuals for existing systems can be studied. Scenarios and use cases can be used to quantify these abstract descriptions [16] into real implementable and understandable software requirements. These are most commonly used elicitation techniques.
In agile software development customer involvement is one of its primary goals. Through interviews and other techniques agile team can extract software requirements for development. This involvement also allows developing trust between the customer and project team [7].

3.1.2 Observation (Contextual techniques)

On some occasions stakeholders cannot describe what they want even though they perform the task every day. In this case observations can be used for elicitating how tasks are performed rather than to ask stakeholders to explain it. Furthermore it is also important to note that context and environment in which the stakeholders perform their tasks should not be ignored [16].

3.1.3 Reusing existing requirements and knowledge

In this process existing knowledge is used to develop the new system [16]. The existing knowledge can be developed with working similar kind of systems or domains. Software requirements can be reused once they are specified and validated [16].

3.1.4 Group elicitation techniques

These techniques encourage stakeholders' agreement and commitment along with using team dynamics to elicitate better understanding of their needs. Brainstorming, focus groups and RAD/JAD2 workshops are some examples of group elicitation techniques [21]. Sometimes different stakeholders have requirements that contradict with each other so these group elicitation techniques facilitate in bringing them to a common ground.
Similar technique is used in agile methodologies such as in ASD to increase customer involvement. At start of the project Joint application development sessions are used to get an understanding of the system [7].

3.2 Requirements Analysis and Negotiation

According to [15] requirements analysis and negotiation are "activities which aim to discover problems with the system requirements and reach agreement on changes to satisfy all system stakeholders". The main goal of requirements analysis is to identify possible conflicts, overlaps, dependencies, inconsistencies or omissions when software requirements are elicited and specified [16]. It is an ongoing process in which all the stakeholders come together to arrive on a concrete set of requirements [23].
Some analysis of software requirements also takes place during the elicitation phase. This is the case when problems with software requirements can be identified as soon as they are expressed. However the extended analysis takes place after the initial version of software requirements document produced [15]. Requirements negotiation can also be included during the analysis phase. Different stakeholders give importance to a software requirement from their own perspective. They also have different levels of power over the decisions being made about particular requirements [15, 22]. The main goal of this phase within RE process is to get the right software requirements [15].
There are different ways to analyze software requirements for example prototypes, mock-ups and test cases that can be used for analyzing and refining the requirements. One of the most important activities of this analysis phase is to ensure that all stake holders from customers to engineers and developers have the same understanding of software requirements [16].
In agile based development the customer is involved in the analysis phases in every iteration. (S)his feedback and involvements leads to better understanding of software requirements.

3.3 Requirements Validation

Requirements validation is performed when you have the final draft of the software requirements [15]. It is a process through which you confirm the elicited software requirements with the actual software requirements of stakeholders [24].Validation is performed to approve that the software requirements are acceptable to be implemented. During this validation phase conflicts between stakeholders are also resolved [15]. It is important to note that execution of software requirements validation process should be done appropriately to save costs that are going to occur later on during the implementation of incorrect requirements. Requirements reviews and inspections can be used to validate software requirements. In reviews a group of people read and analyze requirements, identify any problems and issues and then find a solution for them [15].
In agile methodologies requirements validation is performed through regular review meetings and acceptance tests. Customers can use the developed software and determine which functionality is implemented and what further needs to be developed. It allows the whole team including the customer to know strength and weaknesses of the design [7].

3.4 Requirements Management

According to [15] "Requirements management is the process of managing changes to a system's requirements". Software requirements changes as stakeholders get better understanding of the system under development. These changes have to be documented and control in an efficient way to decrease the probability of costly modifications. A basic requirements management should contain the following activities [15]:

i. Change management of agreed requirements.
ii. Management of interrelationships between those requirements
iii. Management of dependencies between requirements document and other documents produced during the overall software engineering process.

Requirements traceability is a vital activity within requirements management without which it cannot be performed effectively. This traceability information is used to identify what other requirements are/might be affected by making these propose changes [15].

In agile methodologies software requirements are written on index cards or maintained in a product backlog or feature list. The main difference with traditional requirements management is the level of detail in which the software requirements are specified [7].

4 AGILE BASED OFFSHORE SOFTWARE DEVELOPMENT

Off software development occurs when the supplier of software is from a different country then the company who outsourced its development [1]. A Swedish software company could shift some of its software development activities to Pakistan, India or China. Some activities are onshore or the original birth place of the software (to be developed) and some are offshore. The major decision making mainly takes place at onshore site where inception of the project takes place. Offshore site is mainly executioner of high level plans of onshore site for example software requirements specification is developed at onshore site primarily because of the software is developed for their clients. Similarly designing of the software is mostly done offshore because the software development or coding takes place at the offshore site. Software components developed at different offshore sites are brought together at onshore as they have the bigger and holistic picture of the software product.

4.1 Why offshore Software Development

The main reasons for companies moving offshore are reduce cost of development due to lower wages, skilled labor and round the clock development [8]. The different reasons for moving offshore are given below.

4.1.1 Improvement in telecommunications

Globalization of IT and improvement in telecommunication technologies had led to the trend of off-shoring. Offshore sites benefit from bringing together their equipment and communications infrastructure in their home country [9]. Along with it global digital networks had made offshore software development possible because it allows offshore workstations to appear as if they were on the local office network [10]. It does not cost much in terms of time and money to ship software code from offshore to onshore site. The developed software can be transfer within minutes through internet using broadband line.

4.1.2 Favorable government policies

Software companies can take advantage of favorable governmental policies and less tax payment. For example India had developed and encouraged the infrastructure required for building a strong software industry. Also the knowledge market had also been strongly developed by having good educational schools and universities. Along with it they have an open policy for foreign companies to open subsidiaries in India [10].

4.1.3 Skilled and cheap manpower

However the success of this onshore/offshore model depends most importantly on skilled manpower at offshore site [9]. Software development is a labor intensive industry and requires little capital to set up the infrastructure for software development [42].Most companies move offshore because of these direct cost-savings through cheaper labor for example the cost of software developer offshore can be considerably less than the developer onshore.

4.1.4 Less cost of setting offshore site

Also setting up offshore site is not costly as compared to a manufacturing factory. You can rollback all the setup in short time and move it to another place. This is true because of the

intangible nature of software. Along with it the cost of computer related hardware and software have also decreased considerably over the years.

4.2 Why Agile based offshore development

Many of the characteristics that makes suitable agile for software development in general are similar to the suitability of agile for offshore development. In section 3.2 we had mentioned these suitability characteristics in detailed. Over here we discuss them from the perspective of offshore development. Some of the discussion may be a repetition of the discussion in section 3.2
Traditionally for offshore projects plan driven approach is preferred. Detail requirements and design specification are sent to offshore site where they can develop the software [11]. This approach is useful to minimize the communication hiccups that are a basic product of offshore development. But it's almost impossible to get software requirements right in one go. Thus a more flexible process is required that deals more effectively with this changing and evolving nature of software requirements. It is also time consuming as more time is spent on freezing the software requirements. At most of the occasions people do not realize that software requirements evolve and change during software development lifecycle [37] particularly during initial stages. In offshore setting the principles of agile can be successfully executed. Most of the agile principles are based on giving space for decision making to the software development team. This will create more confidence particularly in the offshore team. Also more thinking will be done at offshore bringing in more understanding about the software requirements increasing ownership of the software product. Also more communication takes place between the onshore and offshore teams resulting in much better job performance [13].
In the following section characteristics of agile based offshore development are discussed that can be effective in on/offshore setting:

4.2.1 Iterative development

Agile processes have flourished because they are able to achieve not only high quality software that meets the customers' requirements at the delivery time but also achieve this feat in time and save costs [45]. This is achieved through iterative development. Iterative development of agile methodologies helps in giving a better understanding of requirements especially by the offshore team. Normally iteration is of few weeks containing analysis, design, implementation and testing phases [46]. These phases are not as strictly differentiated as in the case of waterfall model. During the early iterations changes in software requirements occur that will require their reassessment. These early iterations will remove the ambiguities in software requirements and will minimize the chances of implementing software requirements that will prove costly when changing them later in the software development lifecycle [14]. It is particularly useful in offshore setting since these iterations help in understanding the software requirements of the system. This way the offshore team can be involved more in the overall process that will lead to increase in productivity. Thus agile will be very useful for software development where software requirements changes frequently [11].

4.2.2 Regular communication with end-user or customers

Agile puts great focus on communication with the end-user or customer. This communication can help in developing trust amongst both the onshore and offshore teams [13]. Regular long distance meetings between on and offshore site develops better understanding of software requirements [11].

5 CHALLENGES IN UNDERSTANDING SOFTWARE REQUIREMENTS IN AGILE BASED OFFSHORE DEVELOPMENT

It is apparent that agile is most suitable for offshore development. But it also creates different hurdles in offshore development that can negate the advantages of agile offshore development. This chapter explains these challenges in detail through literature review.

5.1 Communication and knowledge sharing

One of the main issues in off-shoring is communication. Agile puts more focus on team effort where the teams are physically co-located. But when teams are geographically distributed especially with large time zone differences then it becomes difficult to co-ordinate. The primary reason is that they can no longer talk face to face every day. Although telephones and teleconferencing allow communicating synchronously but still it takes longer to resolve an issue when teams are at a distance from each other [35]. A common problem in communication is that team at one location is waiting for a response from the team at other location creating misunderstandings and irritation with each other [34].
Knowledge sharing is the fundamental principle of agile methodologies. Since there is less focus on documentation and more focus on communication then more of the knowledge is un-documented. Communication is not the same as physically co-located teams compared to on and offshore teams therefore knowledge sharing becomes more difficult. Although any kind of software requirements specification acts as a mean of knowledge sharing but not everyone in the team will clearly understand the specification and will have questions in (s)his mind. Although new communication technologies help to an extent in knowledge sharing but the deep knowledge embodied in the processes is difficult to share [35].

5.2 Domain and product knowledge

In offshore development activities are divided amongst on and offshore site. More of the development is done offshore and the role of offshore team has much greater importance. According to agile principles every team member has a say and a much greater role than the waterfall model [6]. A good example is that software developer is not just considered as developers but professionals who can provide intelligent solutions to problems that are faced by their team. This means that they should have knowledge about the domain and product that is not available in the software requirements specification or develop it. But generally the countries to which the projects are moved offshore have a highly volatile job market meaning that software professionals switch jobs between companies regularly; the primary reason is to get high salaries. Since in agile there is less focus on documentation and more on developing code [5] therefore a lot of knowledge that is developed walks out with the person when (s)he leaves the company.

5.3 Less software requirements documentation

Less or just enough documentation in agile puts more focus on the real implementation of software requirements. This reduces the time spend on software requirements specification that can be utilized in doing software development. But less documentation puts more on communication within and between agile teams. If the team is physically co-located

questions regarding the software requirements could be easily resolved. But this is not the case with geographically distributed teams as many questions that arises in the minds of the offshore team they have to ask from the onshore team and vice versa [11].

6 DESIGNING QUESTIONNAIRE

This section describes the design, distribution and analysis of questionnaire. Section 6.1 explains the questionnaire design, section 6.2 distribution and section 6.3 provides an analysis of the questionnaire.

6.1 Questionnaire Design

The questionnaire was designed based on the challenges discussed in section 5. Online questionnaire was used because respondents were distributed globally residing in four different countries. It is easier to collect responses by sending a web link to a contact in the companies which they distributed within their respective companies. The anonymity of the respondents was kept to get more realistic responses. While designing the questionnaire checklist from [33] given in appendix 4, had also been taken into consideration. The questionnaire contained a total of 28 questions. 27 of them were close ended and 1 was open ended. The open ended question was included to ask the respondent to briefly mention three most common challenges regarding understanding software requirements that are faced at their site. This would allow us to get challenges that may not be included in the questionnaire. Ten of the closed ended questions were based on Likert scale that has been widely used to get the level of user agreement [cited by 26]. A four-point scale was used to compel respondents to choose any one of them rather than intermediate values. All of the questions in the questionnaire are given in table2 and 3 below. Table 2 contains questions that are not based on Likert scale and table 3 on Likert scale, making it easy for the reader to understand it. To make the tables more understandable for the reader abbreviations given in appendix 3 are used.

Sr no.	Questions	Given options									
1.	Could you mention your company name?	Close ended									
2	Could you enter the country name where your site is situated?	Close ended									
3	What is your role within software development process?	System Analyst	System designer	Software Tester	Software developer	Other					
4	Are you working at the same site as the software requirements holder (from where the requirements are coming)?	Yes	No								
5	Which of the following activities are performed at your site?	PI	RS	RCM	RCR	HLD	DD	IT	TA	PM	RI
6	Do you get any kind of software requirements specification?	Yes	No								
8	Do you get any kind of test specifications with software requirements?	Yes	No								
9	How long is average duration of iteration at your site?	2-4 weeks	4-8weeks	8-12 weeks	12-more weeks						
10	Which of the following tools are used by your site when communicating with other sites about software requirements?	TC	IM	TP	EM	Other					
11	Does your site have face to face meetings with other sites about software requirements at the start of an iteration?	In some cases	In most cases	Never							
12	Does your site have long distance meetings with other sites about software requirements at the start of an iteration?	In some cases	In most cases	Never							
13	How often do you have face to face meetings with end user or	Once every iteration	More than once every iteration	Once every project	More than once every	Never					

	customer about software requirements?				project	
14	How often do you have long distance meetings with end user or customer about software requirements?	Once every iteration	More than once every iteration	Once every project	More than once every project	Never
15	How often do you have face to face meetings with other sites about software requirements?	Once every iteration	More than once every iteration	Once every project	More than once every project	Never
16	How often do you have long distance meetings with other sites about software requirements?	Once every iteration	More than once every iteration	Once every project	More than once every project	Never
21	Which of the following individuals at your site participate in long distance meetings with other sites about software requirements?	System analyst	System designer	Software tester	Software developer	Other
26	How much is employee turn-over rate per year at your site?	1-10%	11-25%	26-50%	More than 50%	
28	Could you briefly describe three most common problems about software requirements that are faced at your site keeping in mind the on/off shore setting.	Open ended question				

Table 2. Questions that are not based on Likert scale.

Sr No.	**Questions**	**Scale**			
		1	2	3	4
7	Is current software requirements specification enough for your site to understand it?	Strongly disagree	Disagree	Agree	Strongly Agree
17	Do regular long distance meetings with other sites help in understanding software requirements by your site?	Strongly disagree	Disagree	Agree	Strongly Agree
18	Is there an increase in long distance meetings rather than face to face meetings because of geographical distance amongst your site and others?	Strongly disagree	Disagree	Agree	Strongly Agree
19	Does lack of communication between different sites creates misunderstandings about software requirements?	Strongly disagree	Disagree	Agree	Strongly Agree
20	Does physical distance between your and other sites slow down knowledge sharing about software requirements?	Strongly disagree	Disagree	Agree	Strongly Agree
22	Are face to face meetings with other sites more helpful than long distance meetings to understand software requirements	Strongly disagree	Disagree	Agree	Strongly Agree
23	Does regular exchange of team members between you and other sites increase overall understanding of software requirements by them?	Strongly disagree	Disagree	Agree	Strongly Agree
24	Does language for example English makes it difficult for your site to understand software requirements when communicating with other sites?	Strongly disagree	Disagree	Agree	Strongly Agree
25	Does knowledge about the development of software product by your sites makes it easier to understand software requirements?	Strongly disagree	Disagree	Agree	Strongly Agree
27	Will high turnover rate of employees at your site increase overall software development time?	Strongly disagree	Disagree	Agree	Strongly Agree

Table 3. Questions that are based on likert scale

The relationship of questions to the challenges mentioned in Section 5 is given below in table 4.

Challenges	**Questions**
Requirements documentation	6-8
Communication and knowledge sharing	9-24
Domain and product knowledge	25- 27

Table 4: Relationship of questions with challenges and solutions

6.2 Questionnaire distribution

The online questionnaire was distributed through emails. First software companies were selected that had agile based on/off shore setup. Then every company was sent an email to ask them whether they would be willing to participate in filling the online questionnaire. When a positive response was received then a web link to online questionnaire was send to them that they would distribute internally in their company. Each contact was requested to distribute this link to both on and off shore respondents within their company, which they did. The questionnaire was available online for two months. Utmost efforts were made to get at least ten responses from each company but it was not possible because all four of the companies could not find enough time from their busy schedules. Email reminders were sent couple of times but could not get more respondents i.e. five onshore and five offshore respondents from every single company. In total we got nine responses which are given in table 5 along with their company names and the location of respondent's site.

Number of responses	Company name	Location
2	Ericsson	Sweden and China
3	Thoughtworks	India
1	Aginity	USA
3	Shinetechchina	China

Table 5. No of respondents and their companies

Two of the respondents were working onshore in their respective company whereas seven were working offshore.

6.3 Analysis of data collected from online questionnaire

We performed quantitative analysis of data collected from the questionnaire to get concrete results. Responses of all likert based questions in terms of percentage are given in Appendix 1. Responses for question 27 which is an open ended question are given in Appendix 2. In this section we will provide distribution and analysis of answers for the key questions that were related to the three challenges.

6.4 Distribution of answers for the three challenges

The distribution of answers against each challenge is given below. Mean values of the responses for each challenge are provided to show their trend as following: 1 is strongly disagree, 2 is disagree, 3 is agree and 4 is strongly agree.

6.4.1 Requirements specification

In figure 5 below the distribution of response of respondents in percentage is shown when asked about question 7 i.e. "Is current software requirement specification enough for your site to understand it". 0% of the respondents strongly agree, 89% agree, 11% disagree and 0% strongly disagree. 2.9 is the mean value of the responses for this question which is closer to agree i.e. 3.

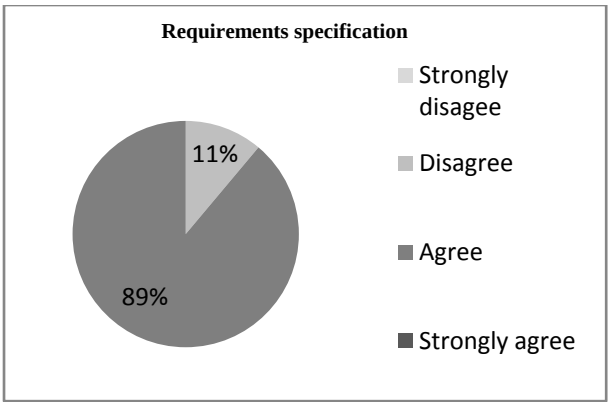

Figure 4. Response regarding the state of software requirement specification

6.4.2 Communication and knowledge sharing

In figure 6 below the distribution of responses in percentage is shown when asked about question 19 i.e. "Does lack of communication between different sites create misunderstandings about software requirements". 33% strongly agree with it, 56% agree, 0% disagrees and 11% strongly disagrees. 3 is the mean value of the responses for this question which is equal to agree i.e. 3.

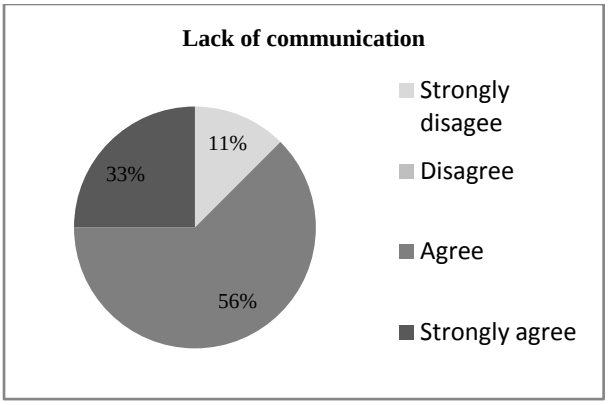

Figure 6. Lack of communication between sites regarding software requirements

In figure 7 below the distribution of responses in percentage is shown when asked about question 20 i.e. "Does physical distance between your and other sites slow down knowledge sharing about software requirements". 22% strongly agree that physical distance between sites slows down knowledge sharing, 56% agree, 22% disagree and 0% strongly disagree.
3 is the mean value of the responses for this question which is equal to agree i.e. 3.

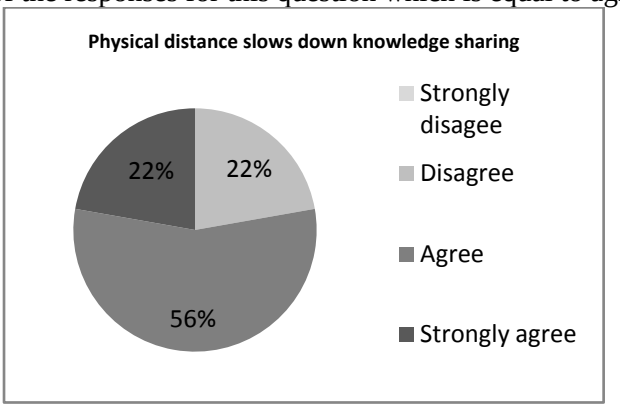

Figure 7. Physical distance between sites slows down knowledge sharing

6.4.3 Language

In figure8 below the distribution of responses in percentage is shown when asked about question 24 i.e. "Does language for example English make it difficult for your site to understand software requirements when communicating with other sites". 0% strongly agrees that language makes it difficult to understand software requirements, 22% agree, 67% disagree and 11% strongly disagree.
2.11 is the mean value of the responses for this question which is closer to disagree i.e. 2.

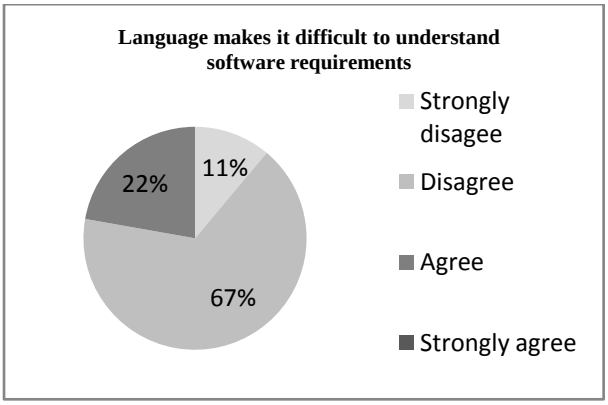

Figure 8. Language makes it difficult to understand software requirements

6.4.4 Domain and product knowledge

In figure 9 below the distribution of responses in percentage is shown when asked about question 25 i.e. "Does knowledge about the development of software product by your site make it easier to understand software requirements". 22% strongly agree, 56% agree, 11% disagree and 11% strongly disagree. 2.88 is the mean value of the responses for this question which is closer to agree i.e. 3.

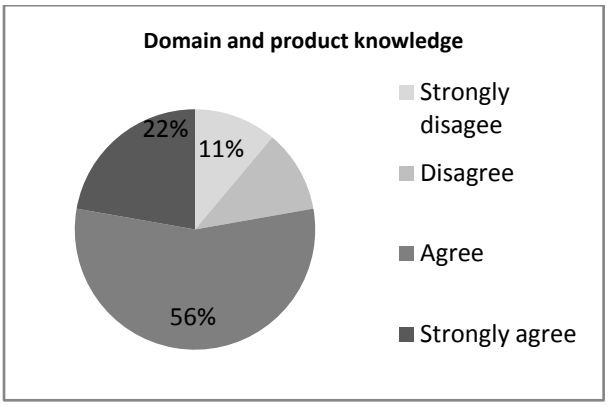

Figure 9. Domain and product knowledge

6.5 Analysis of all responses

In this section the data collected from all respondents is presented to give an overview for better understanding of the topic.
The data for all respondents is presented in table 6 below. It is important to note that only likert based questions are represented in these tables. Abbreviations are used at some places in the tables to make them more readable. These abbreviations are given in Appendix 3.

Respondents	Questions												
	1	3	4	7	17	18	19	20	22	23	24	25	27
1	Shinetechchina	SD	Yes	3	3	3	1	2	4	2	2	1	2
2	Ericsson	PM	Yes	3	3	3	4	4	4	4	3	3	4

3	Shinetechchina	PM	No	3	4	3	3	2	3	4	2	4	3
4	Shinetechchina	SD	No	2	4	3	3	3	4	3	2	4	3
5	Thoughtworks	SD	No	3	3	4	4	4	3	1	1	2	3
6	Thoughtworks	BA	No	3	3	3	3	3	3	3	2	3	3
7	Thoughtworks	SA	No	3	3	3	4	3	4	4	2	3	3
8	Aginity	CTO	No	3	3	3	3	3	3	3	2	3	3
9	Ericsson	PM	No	3	3	4	3	3	4	3	3	3	4
Mean				2,9	3.22	3.22	3.11	3	3.55	3	2.11	2.88	3,11
Standard Deviation				0,33	0.44	0.44	0.927	0.707	0.527	1	0.60	0.92	0,60

Table 6: Responses

Both the onshore respondents agree that the current software requirements specification is enough to understand it. However respondent (software developer) from Shinetechchina disagree that lack of communication between different sites creates misunderstandings about software requirements. (S)his site has face to face meetings with other sites once an iteration which decreases the chances of any type of miscommunication as all the major issues could be resolved effectively. Also their site has long distance meetings with other sites more than once every project particularly at the start of iteration.

On other hand the site of onshore respondent (project manager) from Ericsson only has face to face meetings with other sites only once when the project starts and long distance meetings only once every iteration. Their site faces problems in making a design from the software requirements specification as the requirements are not defined with enough detail. There are late changes in requirement that needs to be accommodated in their design and subsequently software development. (S)his site does not have any face to face or long distance meetings with other sites during an iteration making it difficult for them to resolve any problems. Most of the communication during iteration is done through emails and telephone. (S)his site also has difficulties in understanding software requirements when communicating with other sites because of English language and they communicate in English only when they have non-Swedish speaking team members.

The flow of information from onshore to offshore is going mainly through system analyst, system designer and software testers at the offshore site. It is so because they are involved in long distance meetings with onshore site. Therefore the software developers are mainly dependent on these sources for any kind of information that is not available in software requirements specification, which has both its advantages and disadvantages. The advantage is that they are physically co-located so the communication will be more efficient provided that they get the right answers. Also due to a large organization they need to communicate on formal channels to keep track of status of every software requirements. The disadvantage is that an extra layer is added which is against the agile principles i.e. direct communication [5]. If the answer could not be found locally then it is going to be forwarded to onshore team and the response is going to come back through the same channel. Another reason for involving less personal from offshore site in long distance meetings could be to make these meetings more effective and efficient. There could be many team members and less number of participants allows focusing on key issues and keeping duration of long distance meetings as short as possible.

The respondent from Shinetechchina disagrees to (s)his site requiring previous knowledge about a software product. It could be possible that (s)his site is working with different kinds of domain for example in one project they could be working on Financial services domain whereas in other they could be working on a project that is in educational sector. The respondent from Ericsson agrees that (s)his site needs product knowledge as each site in Ericsson develops a particular feature of software products. Another reason for (s)his agreement could be that their products are mostly market driven which have continuous flow of software requirements[3] and deadlines are much more fixed. Thus requiring the development team to have a good and detail understanding of the software requirements which in most cases will be related to software requirements of previous releases.

Six respondents who are working offshore are satisfied with the current software requirements specification. The one (software developer) at Shinetechchina who is not satisfied with it does not get any kind of software requirements specification. At (s)his site

only the system analyst is participating in long distance meetings with other sites which makes that system analyst main source of information on any kind of software requirements. It puts a lot of pressure on (s)him to remember software requirements that the offshore site has to implement. Since there are no software requirements specifications therefore there is a greater chance of implementing incorrect software requirements. Although agile methodologies discourages detail software requirements specification but still it does support just enough software requirements specification [5].

All of the offshore respondents agree that there exists a lack of communication between on and offshore teams which leads to misunderstandings about software requirements. This lack of communication could be due to a number of reasons such as slow response due to time zone differences.

Seven of the respondents disagrees that communication language for example English makes it difficult to understand software requirements when communicating with other sites. But two respondents who are project managers at Ericsson agree that communication language which in their case is English does create difficulties in understanding software requirements when communicating with other sites. More time could be spent on explaining what they are asking about leading to increase in overall software development time. It could also create difficulties in informal communication through telephones or emails. In case of Ericsson it could put pressure on the offshore team because they could implement incorrect requirements especially during the later stages because of the language. But they can negate this disadvantage by having more face to face or long distance meetings with other sites to decrease the language barrier. System analyst and designer and software testers are involved in their long distance meetings with other sites. Software developers and other team members who do not participate in long distance meetings with other sites can ask questions from their system analyst and designer or software testers in their own local language.

Six of the offshore respondents agree that previous knowledge about the development of software product makes it easier for the team members to understand software requirements. Since agile puts less focus on documentation and more on developing code [5] therefore a lot of knowledge is developed and stored in their mind as opposed to Waterfall development where heavy documentation is given a high priority. For example at Ericsson each site develops a particular feature of the software product therefore a new release of the same product means that lot of software requirements from the previous releases are also included. So it is important to have previous knowledge about the software product.

At Ericsson China they face problems in implementing software requirements due to problems in requirements dependency. Such problems need to be discussed with the onshore team which they already do. There is large number of requirements in market driven products [3]. It is very difficult to resolve the issue of software requirements dependency at the beginning of iteration. Mostly these types of issues arise when these software requirements are implemented. Ericsson China also has problems in understanding software requirements because they are defined at a high level and not broken down into enough detail level.

According to project manager of Shinetechchina in China their site have problems in understanding software requirements due to unclear requirements document, project schedule and slow response from the onshore site. Although their site have long distance and face to face meetings with other sites more than once every iteration but they do not get a quick response from the onshore team. Thus they have to wait which leads to increase in frustration and overall software development time for the offshore time. In accordance with agile development [5], the onshore team does not define detail software requirements. But it becomes problematic when the onshore site cannot be able to provide satisfying answers to the offshore team leading to misunderstandings about software requirements.

According to Software developer at Thoughtworks India the biggest problem faced at their site is related to lack of communication between on and offshore site. Although they have long distance meetings with onshore team more than once every project but it is not proving to be enough. Another reason is that either team does not respond quickly to each other for

example if nothing is happening on onshore site then the onshore team does not inform the offshore team and vice versa creating a big communication gap leading to mistrust and lack of interest in work. Team work the basic principle of agile [5] is destroyed because either team will blame other for failure.

According to system analyst at Thoughtworks India it is important for the offshore team to have domain knowledge about the software product for example if the software product is related to banking then the offshore team should have domain knowledge. According to (s)him it is difficult to transfer tacit knowledge from business context which is onshore to offshore team. . According to [32] tacit knowledge is "non-codified, disembodied know-how that is acquired via the informal take-up of learned behavior and procedures. Learning in an unstructured or semi-structured way is a key process within tacit knowledge acquisition and transfer". This tacit knowledge is difficult to codify and is responsible for producing insight, intuition and decisions based on gut feeling [27]. The knowledge that can be codified into software requirements specification is explicit knowledge but the knowledge that cannot be codified is tacit knowledge. It is difficult for onsite customers to codify tacit knowledge attached to their processes which is very important for the development team to understand. If both teams had been on the same location then it would have been much easier to communicate and explain what they want. But when both teams are not co-located it makes it difficult to share this knowledge. Another problem that (s)he identified is the inability of offshore team to validate requirements that change due to business priorities. In agile methodologies there is no hard coded detail software requirement specification but rather more abstract level software requirements that needs to be interpreted and refined by the offshore team to understand the detail business requirements. Since Business priorities change with passage of time and subsequently software requirements change therefore the offshore team has to make sure that the new software requirements are validated from the onshore team and customer.

According to CTO of Aginity there is a general belief at their site that agile development means little or no software requirements. It makes it difficult for the software development team to perform their tasks as not enough effort is put into specifying software requirements. Although in agile methodologies you do not specify detail software requirements specification as in the traditional waterfall model but still you do specify software requirements. We can take the example of writing compelling stories at onshore site [11] that can be discussed thoroughly between on and offshore site to get the exact software requirements from them. (S)his site at Aginity also beliefs that Quality Assurance team at their site should handle the major testing which is not a problem. But it becomes a problem when there are little or no software requirements defined because testing is performed on the basis of software requirements. Thus there is a general lack of understanding within the software development team about the agile development process. There is also the problem of involving clients to review early builds of the software product for validation of software requirements. This could happen because clients are normally non-IT personal who believe that software development is not difficult to do. Once software requirements are specified then it is just developing it without much of intelligence required. They don't realize the intangible nature of software development and how complicated it is. Another reason could be that they cannot find enough time to be part of the software development team as they have to perform their own duties.

7 DISCUSSION OF DIFFERENT GROUPS

Different groups were made of the data collected to identify any particular reasons to better understand the challenges identified in section 5. These groups are given below:

i. Role group
ii. Iteration-duration group
iii. Test specification group
iv. Long distance meetings with other sites
v. Long distance meetings with other sites at the start of iteration group
vi. On/offshore group

For making the data easily readable it is presented in two tables for each sub-group within all groups.

7.1 Role Group

The data was observed by grouping it on the basis of role performed by each respondent in their respective companies. In table 8 and 9 below data is given for project managers, CTO, system analyst and business analyst whereas software developers are given in table 10 and 11. The project managers, CTO, system analyst and business analyst were grouped together because they are at a higher level in the project team. For ease of discussion this sub-group was called "analyst".

Respondents	**Questions**												
	3	**4**	**5**	**6**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1(Shinetechchina)	PM	No	RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	4-8 weeks	TC, IM, TP, EM	In some cases	In some cases	More than once every iteration	More than once every iteration	More than once every iteration	More than once every iteration
2(Thoughtworks)	BA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	Once every iteration	More than once every project	Once very iteration
3(Thoughtworks)	SA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PPM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	More than once every iteration	More than once every project	More than once every project
4(Aginity)	CTO	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, Project Monitoring, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM, SK, WI, WC	In some cases	In most cases	More than once every iteration	More than once every iteration	Once every project	More than once every iteration
5(Ericsson)	PM	Yes	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	12-more weeks	TC, TP, EM	In some cases	In some cases	Never	Never	Once every project	Once every iteration
6(Ericsson)	PM	No	PI, RS, RCM,	Yes	No	4-8 weeks	TC, TP,	In some	In some	Never	Never	More than once	More than once

			RCR, HLD, DD, IT, TA, PM, RI				EM	cases	cases			every iteration	every iteration

Table 8: Non-Likert Responses of analyst

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	4	3	3	2	3	4	2	4
2(Thoughtworks)	3	3	3	3	3	3	3	2	3
3(Thoughtworks)	3	3	3	4	3	4	4	2	3
4(Aginity)	3	3	3	3	3	3	3	2	3
5(Ericsson)	3	3	3	4	4	4	4	3	3
6(Ericsson)	3	3	4	3	3	4	3	3	3
Mean	3	3.16	3.166	3.33	3	3.5	3.5	2.33	3.166
Standard deviation	0	0.408	0.408	0.516	0.632	0.547	0.547	0.516	0.408

Table 9. Likert Responses of PM, SA, BA, CTO

Respondents	**Questions**											
	4	**5**	**6**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1(Shinetechchina)	Yes	DD, IT,PM, RI	Yes	No	4-8 weeks	TC, IM, EM	In some cases	In most cases	Never	Never	Once every iteration	More than once every project
2(Shinetechchina)	No	DD, RI	No	No	4-8 weeks	IM, EM	In some cases	In some cases	Once every project	Never	Once every iteration	More than once every project
3(Thoughtworks)	No	PI, RS, HLD, DD, IT, TA, PM, RI	No	Yes	2-4 weeks	TC, IM, TP, EM , WI, WB	Never	In most cases	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration

Table 10: Non-Likert Responses of Software developers

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Shinetechchina)	2	4	3	3	3	4	3	2	4
3(Thoughtworks)	3	3	4	4	4	3	1	1	2
Mean	2.67	3.33	3.33	2.67	3	3.67	2	1.67	2.33
Standard deviation	0.577	0.577	0.577	1.527	1	0.577	1	0.577	1.52

Table 11. Non-Likert Responses of software developers

Most of the respondents within both sub-groups in role group are satisfied with the current software requirements specification at their site. Only one respondent within the software developer sub-group from Shinetechchina is not satisfied with it because (s)he does not get any kind of software requirements specification. At (s)his site detail design and implementation are performed. Software requirements are finalized after mutual agreement with the onshore team therefore they need to be documented for implementation by the offshore team. But the absence of any kind of software requirements specification makes it very difficult for (s)his team to perform these activities. Although agile strongly supports less software documentation [11] but it does not mean that there should not be any of it at all. It also puts more pressure on the development team to remember every software requirement which is almost impossible. Even though the development team have a detail understanding of the software product but still it becomes very difficult to remember every software requirement which in most cases are quite a few of them.

The analyst group agrees more than the software developer group about misunderstanding about software requirements due to lack of communication. A particular reason for the

analyst group to have such a level of agreement could be that they are in between on and offshore site and performs role of a bridge between them. They more or less have to communicate with the software development team about software requirements and find it difficult to explain it due to lack of communication. The software developer group also agrees about this lack of communication apart from one who strongly disagrees. (S)his site already have face to face meetings with other sites once every iteration and long distance meetings at the beginning of iteration. It shows that lot of effort and time is put into these meetings. In general all of the respondents agree on the importance of communication between on and offshore site which is in accordance with agile methodologies i.e. strong interaction with all the project team.

The analyst subgroup agrees that domain and product knowledge helps in understanding software requirements since there is less focus on detail software requirements specification. Another reason is that analyst sub-group finds it easier to discuss software requirements with team members who already have domain or product knowledge. On other hand, most of the software developers disagree with it because they have to write software code and they may always want concrete and detail software requirements as it is easier to implement. Thus requiring greater effort to be put into making software requirements more documented which is what software developers mostly complain about. The response of one of the software developers who strongly agrees makes an interesting distinction in the software developer group because development teams at (s)his site does not get any kind of software requirement specification. It makes software development more dependent on the team which is not a disadvantage but if any one of them leaves the team due to any reason then the replacement should be able to know all about the software product.

7.2 Iteration-duration group

In agile, duration of iteration depends upon how much of the functionality is going to be implemented [5]. Smaller iterations keep the effect of change as minimum as possible. The responses are discussed by grouping them on basis of iteration duration.

Respondents	**Questions**											
	3	4	5	6	8	10	11	12	13	14	15	16
1(Thoughtworks)	SD	No	PI,RS, HLD, DD,IT, TA	No	Yes	TC,IM ,TP,E M,WI, WB	Never	In most cases	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration
2(Thoughtworks)	BA	No	PI,RS, RCM, RCR, HLD, DD,IT, TA,P M,RI	Yes	No	TC,IM ,TP,E M	In most cases	In most case	More than once every project	Once every iteration	More than once every project	Once every iteration
3(Thoughtworks)	SA	No	PI,RS, RCM, RCR, HLD, DD	Yes	Yes	TC,IM ,TP,E M	In most cases	In most case	More than once every project	More than once iteration	More than once every project	More than once every project
4(Aginity)	CTO	No	PI,RS, RCM, RCR, HLD, DD,IT, TA,P M, RI	Yes	Yes	TC,IM ,TP,E M.SK, WI, WC	In some cases	In most case	More than once every iteration	More than once every iteration	Once every project	More than once every iteration

Table 12. Non-Likert Responses of Iteration duration: 2-4 weeks.

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Thoughtworks)	3	3	4	4	4	1	1	1	2
2(Thoughtworks)	3	3	3	3	3	3	3	2	3
3(Thoughtworks)	3	3	3	4	3	4	4	2	3
4(Aginity)	3	3	3	3	3	3	3	2	3
Mean	3	3	3.25	3.5	3.25	2.75	2.75	1.75	2.75
Standard	0	0	0.5	0.58	0.5	1.25		0.5	0.5

deviation									

Table 13. Likert Responses of Iteration duration: 2-4 weeks.

In table 12 and 13 we present data for 2-4 weeks iteration duration. This sub-group is satisfied with the software requirements specification. They strongly agree with lack of communication due to geographical distance between on and offshore team as a reason for misunderstanding about software requirements at their respective sites. When the team is physically co-located then all sorts of informal and social communication is going on within the team which helps in sharing knowledge and having better understanding of software requirements. This is not the case in offshore development [11, 14]. Having a small duration of 2-4 weeks requires more regular communication between on and offshore team. This subgroup does not have any language problem when communicating with the onshore team. Thus having no language problems eases the communication between sites which otherwise creates lots of difficulties in communication.

Respondents	**Questions**											
	3	**4**	**5**	**6**	**8**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1(Shinetechchina)	SD	Yes	DD,IT,PM,RI	Yes	No	TC, IM, EM	In some cases	In most cases	Never	Never	Once every iteration	More than once every project
2(Shinetechchina)	PM	No	RS, RCM,RCR,HLD, DD, IT, TA, PM, RI	Yes	Yes	TC, IM, TP, EM	In some cases	In some cases	More than once every iteration	More than once every iteration	More than once every iteration	More than once every iteration
3(Shinetechchina)	SD	No	DD, RI	No	No	IM, EM	In some cases	In some cases	Once every project	Never	Once every iteration	More than once every project
4 (Ericsson)	PM	No	PI,RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	TC, TP, EM	In some cases	In some cases	Never	Never	More than once every iteration	More than once every iteration

Table 14. Non-likert responses of Iteration duration: 4-8 weeks

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Shinetechchina)	3	4	3	3	2	3	4	2	4
3(Shinetechchina)	2	4	3	3	3	4	3	2	4
4 (Ericsson)	3	3	4	3	3	4	3	3	3
Mean	2.75	3.5	3.25	2.5	2.5	3.75	3	2.25	3
Standard deviation	0.5	0.57	0.5	1	0.57	0.5	0.81	0.5	1.41

Table 15. Likert responses of Iteration duration: 4-8 weeks

The sub-group given in table 14 and 15 has an iteration duration of 4-8 weeks which is larger than normal iterations proposed by agile methodologies. However larger duration for iteration means that more software requirements are included in it and allowing the extra time to be consumed by communication overhead between on and offshore team. Three of the respondents within this subgroup also agree that lack of communication creates misunderstandings about software requirements whereas one strongly disagrees with it. Two of the respondents disagree with physical distance amongst on and offshore team as a hurdle in sharing knowledge about software requirements. [11, 14] have identified physical distance as one of the key issue in sharing knowledge between on and offshore team. This attitude is due to frequent communication with onshore team by their respective site. It is interesting to note that one of the respondents i.e. project manager from Ericsson has agreed with English language as a hurdle in communicating with other sites about software requirements. This could mean that (s)his site will also have communication problems with other site due to language barrier leading to increase in overall software development time. [35] Had

identified language barrier as one of the common issues in agile based offshore development especially when collaborating with onshore team.

Respondents	Questions											
	3	4	5	6	8	10	11	12	13	14	15	16
1(Ericsson)	PM	Yes	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	TC, TP, EM	In some cases	In some cases	Never	Never	Once every project	Once every iteration

Table 16. Non-Likert responses of Iteration duration: 12-more weeks

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Ericsson)	3	3	3	4	4	4	4	3	3
Mean	3	3	3	4	4	4	4	3	3
Standard deviation	0	0	0	0	0	0	0	0	0

Table 17. Likert responses of Iteration duration: 12-more weeks

Only one respondent shown in table 16 and 17 had a 12-more weeks iteration. It is due to the nature of its software product which is mainly market driven and regular changes in software requirements requiring more time to implement. This respondent strongly agrees that lack of communication and physical distance amongst sites creates misunderstandings about software requirements. (S)he also agrees that English language is a barrier in understanding software requirements.

The language problem in understanding software requirements is only identified by respondents at Ericsson. It shows that when both on and offshore sites are non-English speakers and they communicate in English, it is not easy to understand each other. Shorter duration iteration puts more importance on communication between on and offshore site than longer duration iteration due to much less time.

7.3 Test specification group

Now we group the data on the basis of test specifications. Below in table 18 and 19 data is presented for those respondents who get test specification. In table 20 and 21 data is given for those respondents who do not get test specifications.

Respondents	Questions											
	3	4	5	6	9	10	11	12	13	14	15	16
1(Shinetechchina)	PM	No	RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	4-8 weeks	TC, IM, TP, EM	In some case	In some cases	More than once every iteration	More than once every iteration	More than once every iteration	More than once every iteration
2(Thoughtworks)	SD	No	PI, RS, HLD, DD, IT, TA, PM, RI	No	2-4 weeks	TC, IM, TP, EM ,WI ,WB	Never	In most cases	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration
3(Thoughtworks)	SA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	More than once every iteration	More than once every project	More than once every project
4(Aginity)	CTO	No	PI, RS, RCM, RCR, HLD,	Yes	2-4 weeks	TC, IM, TP, EM	In some cases	In most cases	More than once every iteration	More than once every iteration	Once every project	More than once every iteration

			DD, IT, TA, PM, RI			, SK, WI, WC						
5(Ericsson)	PM	Yes	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	12-more weeks	TC, TP, EM	In some cases	In some cases	Never	Never	Once every project	Once every iteration

Table 18. Non-likert responses: test specification

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Shinetechchina)	3	4	3	3	2	3	4	2	4
2(Thoughtworks)	3	3	4	4	4	3	1	1	2
3(Thoughtworks)	3	3	3	4	3	4	4	2	3
4(Aginity)	3	3	3	3	3	3	3	2	3
5(Ericsson)	3	3	3	4	4	4	4	3	3
Mean	3	3.2	3.2	3.6	3.2	3.4	3.2	2	3
Standard deviation	0	0.44	0.444	0.54	0.83	0.54	1.30	0.7	0.7

Table 19. Likert responses: Test specification

Respondents	Questions											
	3	4	5	6	9	10	11	12	13	14	15	16
1(Shinetechchina)	SD	Yes	DD, IT,PM, RI	Yes	4-8 weeks	TC, IM, EM	In some cases	In most cases	Never	Never	Once every iteration	More than once every project
2(Shinetechchina)	SD	No	DD,RI	No	4-8 weeks	IM, EM	In some cases	In some casess	Once every project	Never	Once every iteration	More than once every project
3(Thoughtworks)	BA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	Once every iteration	More than once every project	Once every iteration
4(Ericsson)	PM	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	4-8 weeks	TC, TP, EM	In some cases	In some cases	Never	Never	More than once every iteration	More than once every iteration

Table 20: Non-likert responses: No-test specification

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Shinetechchina)	2	4	3	3	3	4	3	2	4
3(Thoughtworks)	3	3	3	3	3	3	3	2	3
4(Ericsson)	3	3	4	3	3	4	3	3	3
Mean	2.75	3.25	3.25	2.5	2.75	3.75	2.75	2.25	2.75
Standard deviation	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	1.25

Table 21. Likert responses: No-test specification.

Within the test specification sub-group all of the respondents are satisfied with their current software requirements specification. In no-test specification only one respondent is dissatisfied as (s)he does not get any kind of software requirements specification. Since this respondent does not get any kind of software requirements specification or test specifications it makes it difficult for him to understand and remember every software requirements that is going to be implemented.

The test specification subgroup strongly agrees that lack of communication and physical distance leads to misunderstandings about software requirements. Most of the respondents in

no test specification subgroup also agree with it except one from Shinetechchina who strongly disagrees with it. (s)he is not satisfied with the software requirements at (s)his site. It means that more detail and concrete software requirements specification will help in better understanding of software requirements by (s)him. In general both test and no-test specification sub groups are satisfied with the lack of communication as a reason for misunderstandings about software requirements.
Both test specification and no-test specification subgroup agrees that team members should have domain and product knowledge for better understanding of software requirements which will also help in communication.
Both of the subgroups agree with lack of communication as a hurdle in software requirements. Since regular communication is a primary principle of agile [5] therefore both on and offshore agile teams require regular communication with each other. This communication will be more effective if both sites have domain and product knowledge.
Strong emphasis has been put on test specifications for communicating software requirements between different sites. More mature XP teams use acceptance tests for communicating software requirements. They get test scripts before the start of an iteration to allow development team to aim at a concrete target [11].

7.4 Long distance meeting with other sites

Now the data is grouped by how often a site has long distance meetings with other sites. In agile based offshore development communication is very important for achieving better understanding of software requirements. Great geographical distances amongst sites makes it difficult to have face to face meetings therefore long distance meetings are used instead [11]. Three sub-groups are obtained of long distance meetings with other sites which are as follows: more than once every iteration, once every iteration and more than once every project.

Respondents	Questions											
	3	4	5	6	8	9	10	11	12	13	14	15
1(Shinetechchina)	PM	No	RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	4-8 weeks	TC, IM, TP, EM	In some cases	In some cases	More than once every iteration	More than one every iteration	More than once every iteration
2(Thoughtworks)	SD	No	PI, RS, HLD, DD, IT, TA, PM, RI	No	Yes	2-4 weeks	TC, IM, TP, EM, WI, WB	Never	In most cases	More than once very project	More than one every iteration	More than once every iteration
3(Aginity)	CTO	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM, SK, WI, WC	In some cases	In most cases	More than once iteration	More than one every iteration	Once every project
4(Ericsson)	PM	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	4-8 weeks	TC, TP, EM	In some cases	In some cases	Never	Never	More than once every iteration

Table 22. Likert responses of Long distance meetings with other sites: more than once every iteration

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Shinetechchina)	3	4	3	3	2	3	4	2	4
2(Thoughtworks)	3	3	4	4	4	3	1	1	2

3(Aginity)	3	3	3	3	3	3	3	2	3
4(Ericsson)	3	3	4	3	3	4	3	3	3
Mean	3	3.25	3.5	3.25	3	3.25	2.75	2	3
Standard deviation	0	0.5	0.577	0.5	0.816	0.5	1.25	0.816	0.816

Table 23. Likert responses of Long distance meetings with other sites: more than once every iteration

In the subgroup given in table 22 and 23 all of the respondents have long distance meetings with other sites more than once every iteration. This subgroup is satisfied with their current software requirements specification. They agree that lack of communication between different sites creates misunderstandings about software requirements. For this purpose they have long distance meetings with other sites more than once every iteration and strongly support it. This frequents communication with other sites allows them to share knowledge that otherwise is not available in the software requirements specification. They agree that team members should have previous domain and product knowledge to better understand software requirements. This frequent communication allows on and offshore team to develop a level of understanding and trust which facilitates in sharing knowledge. All of the respondents except from Ericsson within this subgroup disagree that they have language problem during communication between on and offshore team.

Respondents	**Questions**											
	3	**4**	**5**	**6**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1(Ericsson)	PM	Yes	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	12-more weeks	TC, TP, EM	In some cases	In some cases	Never	Never	Once every project
2(Thoughtworks)	BA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	2-4 weeks	TC, IM, TP, EM	In most cases	In more cases	More than once every project	Once very iteration	More than once every project

Table 24. Non-Likert responses of long distance meetings with other sites: once every iteration

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Ericsson)	3	3	3	4	4	4	4	3	3
2(Thoughtworks)	3	3	3	3	3	3	3	2	3
Mean	3	3	3	3.5	3.5	3.5	3.5	2.5	3
Standard deviation	0	0	0	0.707	0.707	0.707	0.707	0.707	0

Table 25. Likert responses of long distance meetings with other sites: once every iteration

The subgroup given in table 24 and 25 has long distance meetings with other sites once every iteration. This subgroup is satisfied with the software requirements specification and strongly agrees that lack of communication leads to misunderstandings about software requirements. However they have long distance meetings with other sites only once every iteration which is not enough and requires more communication with other sites, maybe more than once every iteration. One respondent i.e. Business analyst from Thoughtworks disagrees and the other project manager from Ericsson agrees that language is hurdle in communication between different sites. Since project manager's site at Ericsson has only one long distance meeting every iteration therefore communication level cannot be improved with other sites. It requires frequent long distance meetings with other sites to develop strong

communication between them. Both respondents agree that domain and product knowledge helps in better understanding of software requirements. In case of Ericsson it is important that team members should have knowledge about the development of software products because of working on different versions of the same product.

Respondents	Questions											
	3	4	5	6	8	9	10	11	12	13	14	15
1(Shinetechchina)	SD	Yes	DD,IT, PM,RI	Yes	No	4-8 weeks	TC,IM ,EM	In some cases	In most cases	Never	Never	Once every iteration
2(Shinetechchina)	SD	No	DD,RI	No	No	4-8 weeks	IM,E M	In some cases	In some cases	Once every project	Never	Once every iteration
3(Thoughtworks)	SA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC,IM ,TP,E M	In most cases	In most cases	More than once every project	More than once every iteration	More than once every project

Table 26. Non-Likert responses of long distance meetings with other sites: more than once every project

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Shinetechchina)	2	4	3	3	3	4	3	2	4
3(Thoughtworks)	3	3	3	4	3	4	4	2	3
Mean	2.67	3.33	3	2.66	2.66	4	3	2	2.66
Standard deviation	0.577	0.57	0	1.527	0.577	0	1	0	1.52

Table 27. Likert responses of long distance meetings with other sites: more than once every project

The subgroup given in table 26 and 27 has long distance meetings with other sites more than once every project. It is important to note that long distance meetings “more than once every project” are less than “more than once every iteration”. Two of the respondents within this subgroup agree that lack of communication between different sites creates misunderstandings about software requirements however one respondent i.e. software developer from Shinetechchina working onshore strongly disagrees with it and is satisfied with the current communication between different sites. (S)he and system analyst from Thoughtworks are satisfied with their current software requirements specification. However the other software developer from Shinetechchina who is working offshore is not satisfied with the current software requirements specification. This dissatisfaction could be due to lack of communication between (s)his site and the onshore team. As mentioned frequently above that in agile based development software requirements are not defined in detail which requires more communication with the onshore team to clarify misunderstandings about these software requirements.

Regular long distance meetings between on and offshore sites will allow to better understand software requirements. Long distance meetings once or twice every week between on and offshore site would strengthen communication between them and will also be able to resolve misunderstanding about software requirements much quicker [11].

7.5 Long distance meeting at the start of an iteration

In this group the collected data is discussed with respect to long distance meetings at the start of iteration. Two subgroups are available: first is “in most cases” and second “in some cases”.

Respondents	Questions											
	3	4	5	6	8	9	10	11	13	14	15	16

1(Shinetechchina)	SD	Yes	DD, IT,PM, RI	Yes	No	4-8 weeks	TC, IM, EM	In some cases	Never	Never	Once every iteration	More than once every project
2(Thoughtworks)	SD	No	PI, RS, HLD, DD, IT, TA, PM, RI	No	Yes	2-4 weeks	TC, IM, TP, EM, WI, WB	Never	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration
3(Thoughtworks)	BA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	2-4 weeks	TC, IM, TP, EM	In most cases	More than once every project	Once every iteration	More than once every project	Once every iteration
4(Thoughtworks)	SA	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM	In most cases	More than once every project	More than once every iteration	More than once every project	More than once every project
5(Aginity)	CTO	No	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM, SK, WI, WC	In some cases	More than once every iteration	More than once every iteration	Once every project	More than once every iteration

Table 28. Non-Likert base responses: In most cases Long distance meetings with other sites.

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Thoughtworks)	3	3	4	4	4	3	1	1	2
3(Thoughtworks)	3	3	3	3	3	3	3	2	3
4(Thoughtworks)	3	3	3	4	3	4	4	2	3
5(Aginity)	3	3	3	3	3	3	3	2	3
Mean	3	3	3.2	3	3	3.4	2.6	1.8	2.4
Standard deviation	0	0	0.44	1.22	0.70	0.54	1.14	0.44	0.89

Table 29. Likert base responses: In most cases Long distance meetings with other sites.

Respondents	**Questions**											
	3	**4**	**5**	**6**	**8**	**9**	**10**	**11**	**13**	**14**	**15**	**16**
1(Shinetechchina)	PM	No	RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	4-8 weeks	TC,IM,TP,EM	In some cases	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration
2(Shinetechchina)	SD	No	DD, RI	No	No	4-8 weeks	IM,EM	In some cases	Once every project	Never	Once every iteration	More than once every project
3(Ericsson)	PM	Yes	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	12-more weeks	TC,TP,EM	In some cases	Never	Never	Once every project	Once every iteration
4(Ericsson)	PM	No	PI, RS, RCM, RCR, HLD, DD,	Yes	Yes	4-8 weeks	TC,TP,EM	In some cases	Never	Never	More than once iteration	More than once every iteration

			IT, TA, PM, RI									

Table 30. Non-Likert responses: In some cases Long distance meetings with other sites.

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	4	3	3	2	3	4	2	4
2(Shinetechchina)	2	4	3	3	3	4	3	2	4
3(Ericsson)	3	3	3	4	4	4	4	3	3
4(Ericsson)	3	3	4	3	3	4	3	3	3
Mean	2.75	3.5	3.25	3.25	3	3.75	3.5	2.5	3.5
Standard deviation	0.5	0.577	0.5	0.5	0.816	0.5	0.577	0.577	0.577

Table 31. Likert responses: In some cases Long distance meetings with other sites.

Both of the sub-groups are satisfied with their software requirements specification. They also agree that lack of communication between different sites creates misunderstandings about software requirements except one of the respondent who is software developer from Shinetechchina and working offshore. (S)his response differs from his own subgroup and the other sub-group. Table 30 and 31 contains data for respondents who have long distance meetings with other sites in some cases.

Both of the subgroups agree that physical distance between different sites slows down knowledge sharing except one of the respondents already mentioned above disagrees with the rest of them. (S)his different attitude could be possible that (s)he is not involved in any kind of activities related to specifying software requirements since in most cases system analyst and testers working onshore are more in contact with the offshore team.

The first subgroup disagrees with English language as a hurdle in understanding software requirements since they have quite frequent communication with other sites therefore language issues are resolved. The second subgroup has mixed level of agreements. Only Ericsson is having the language problem because at their sites English is not spoken as a first language.

The first subgroup has mixed level of agreements when asked about "Does knowledge about the development of software product by your site makes it easier to understand software requirements". Both of the software developers disagree with it whereas the other respondents agree with it. A particular reason for agreement of three respondents could be that they are more involved in communication between different sites. They realize that it is much easier to explain something if both onshore and offshore team have previous product and domain knowledge. The second sub-group strongly agrees that domain and product knowledge is helpful in understanding software requirements.

Agile supports frequent communication between on and offshore team to resolve any misunderstandings about software requirements. Since software requirements are not detailed enough therefore long distance meetings at the start of iteration between on and offshore team will considerably facilitate in better understanding of software requirements. The offshore site can ask a lot of questions from the onshore site and in particular the onshore client about the software requirements that would lead to a much better implementation of these software requirements [11].

7.6 On/offshore group

In this group there are two sub-groups: on and offshore. It is important to take on and offshore perspectives to see that are there any differences or similarities in their responses. The offshore sub-group contains more responses than onshore group. It could affect the mean and standard values of their responses. As mentioned previously utmost efforts were made to get more onshore responses but it was not possible due to lack of time available for responding to the questionnaire from these software companies. Below in table 32 and 33

data is given for the offshore respondents whereas table 34 and 35 contains data of onshore respondents.

Respondents	Questions											
	3	5	6	8	9	10	11	12	13	14	15	16
1(Shinetechchina)	PM	RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	4-8 weeks	TC, IM, TP, EM	In some cases	In some cases	More than once every iteration	More than once every iteration	More than once every iteration	More than once every iteration
2(Shinetechchina)	SD	DD, RI	No	No	4-8 weeks	IM, EM	In some cases	In some cases	Once every project	Never	Once every iteration	More than once every project
3(Thoughtworks)	SD	PI, RS, HLD, DD, IT, TA, PM, RI	No	Yes	2-4 weeks	TC, IM, TP, EM , WI, WB	Never	In most cases	More than once every project	More than once every iteration	More than once every iteration	More than once every iteration
4(Thoughtworks)	BA	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM	Yes	No	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	Once every iteration	More than once every project	Once every iteration
5(Thoughtworks)	SA	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM	In most cases	In most cases	More than once every project	More than once every iteration	More than once every project	More than once every project
6(Aginity)	CTO	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	Yes	2-4 weeks	TC, IM, TP, EM , SK, WI, WC	In some cases	In most cases	More than once every iteration	More than once every iteration	Once every project	More than once every iteration
7(Ericsson)	PM	PI, RS, RCM, RCR, HLD, DD, IT, TA, PM, RI	Yes	No	4-8 weeks	TC, TP, EM	In some cases	In some cases	Never	Never	More than once every iteration	More than once every iteration

Table 32: Non-Likert responses of offshore respondents

Respondents	Questions								
	7	17	18	19	20	22	23	24	25
1(Shinetechchina)	3	4	3	3	2	3	4	2	4
2(Shinetechchina)	2	4	3	3	3	4	3	2	4
3(Thoughtworks)	3	3	4	4	4	3	1	1	2
4(Thoughtworks)	3	3	3	3	3	3	3	2	3
5(Thoughtworks)	3	3	3	4	3	4	4	2	3
6(Aginity)	3	3	3	3	3	3	3	2	3
7(Ericsson)	3	3	4	3	3	4	3	3	3
Mean	2.89	3.285	3.285	3.285	3	3.42	3	2	3.14
Standard deviation	0.377	0.48	0.48	0.48	0.577	0.534	1	0.577	0.69

Table 33: Likert responses of offshore respondents

Respondents	Questions											
	3	5	6	8	9	10	11	12	13	14	15	16
1(Shinetechchina)	SD	DD, IT,PM, RI	Yes	No	4-8 weeks	TC, IM, EM	In most cases	In most cases	Never	Never	Once every iteration	More than once every project
2(Ericsson)	PM	PI, RS,	Yes	Yes	12-more weeks	TC,	In some	In some	Never	Never	Once	Once

		RCM, RCR, HLD, DD, IT, TA, PM, RI				TP, EM	cases	cases			every project	every iteration

Table 34: Non-Likert responses of onshore respondents

Respondents	**Questions**								
	7	**17**	**18**	**19**	**20**	**22**	**23**	**24**	**25**
1(Shinetechchina)	3	3	3	1	2	4	2	2	1
2(Ericsson)	3	3	3	4	4	4	4	3	3
Mean	3	0	0	2.5	3	4	3	2.5	2
Standard deviation	0	0	0	2.12	1.4	0	1.41	0.70	1.414

Table 35: Likert responses of onshore respondents

Both of the sub-groups are satisfied with their software requirements specification except one respondent from the onshore subgroup who does not get any software requirements specification. The offshore sub group agrees that lack of communication between different sites leads to misunderstandings about software requirements. On the other hand onshore sub-group has more dispersed response. One of the onshore respond strongly disagrees with it whereas the other one strongly agrees with it. The one who agrees has regular long distance meetings and face to face meetings once very iteration with other sites. The other respondent within onshore sub-group strongly agrees with it and that is understandable because (s)his site have long distance meeting only once every iteration and face to face meeting once very project. This respondent had identified a problem that is faced at (s)his site i.e. software requirements are not defined with enough detail. However that is one of the main principles of agile [5] to focus more on developing software code and less on software requirements specification or just having enough software requirements specification. The responses of both sub-groups are more uniform when asked about physical distance between sites as a hurdle in knowledge sharing about software requirements. The offshore subgroup except one respondent from Ericsson disagrees that language makes it difficult to understand software requirements. Similarly within the onshore sub-group only project manager from Ericsson agrees with having the language problem during communication whereas the other disagrees. As mentioned frequently before, language problems are faced only at Ericsson. The offshore sub-group agrees that previous domain and product knowledge makes it easier to understand software requirements whereas the onshore sub-group has more dispersed where one strongly disagrees and the other agrees.

7.7 Validity threats

According to [33] "There are several threats to validity that will raise potential issues about an experimenter's ability to conclude that the intervention affects on outcome". This thesis work was of quantitative nature. There are several aspects of validating a survey that must be taken into consideration when verifying the measurements of intended characteristics [25]. They are given below:

7.7.1 Content validity

According to [25] "Content validity is a subjective assessment of how appropriate the instrument seems to a group of reviewers (i.e. a focus group) with knowledge of the subject matter". We reviewed the questionnaire repetitively with our advisor to ensure that it includes everything it should regarding our research area.

7.7.2 Face Validity

According to [25] “Face validity is a cursory review of items by untrained judges”. We distributed the online questionnaire survey amongst different students to identify any particular problems with formulation and positioning of questions in the online survey before sending it to the software companies.

7.7.3 Other validity threats

Although it was tried to get some knowledge about details of the software process at each company through some of the questions in online survey, it was very difficult as number of questions were kept as minimum as possible. What we do know that each company has its own process and conditions which are different from the others. It was impossible to physically observe each respondent while answering the online questionnaire due to physical distance. Personal biasness of the respondents can also be a threat to our research work.

8 CONCLUSION

In this section a brief conclusion of the research work is provided. There has been research work on agile based offshore development that provides us a general insight into its costs and benefits. But it has been focused on generic challenges such as lack of communication, language problem, cultural differences and difficulties in knowledge sharing due to physical distance.
This thesis work was motivated by the fact that agile methodologies put strong emphasis on team members be physically co-located to better understand software requirements. But this is not the case in on/offshore setting. However acknowledging existence of these challenges would allow advantages of agile methodologies to be achieved in this setting.
The contribution of this research work was to analyze the challenges mentioned above from different perspectives for example comparing the responses of on and offshore respondents. This analysis would then allow us to look deeper into these challenges and dealt with them more effectively.

8.1 Answers to the research questions

The research work in this thesis was aimed to answer some research questions. This section answers those research questions and what is its contribution to agile based on/offshore research.

Q1. What are the challenges in understanding software requirements in agile based offshore development?

i. Communication and knowledge sharing

It was confirmed that lack of communication between on and offshore teams is a reason for having misunderstandings about software requirements. It leads to frustration among team members at both sites. Therefore a lot of questions in the online questionnaire were related to communication between on and offshore team. In agile development methodologies software requirements are defined at an abstract level rather than in the traditional detail manner. A lot of knowledge about software requirements is not present in these specifications. Software requirements change rapidly especially due to change in business context particularly for market driven software products. Respondents who have 2-4 weeks iteration agree that physical distance slows down knowledge sharing about software requirements whereas respondents who have 4-8 weeks iteration disagree with it. Thus in shorter iterations physical distance between on and offshore teams become a hurdle as there is less time to implement software requirements.
For Ericsson in China the software requirements are not defined detailed enough which is in accordance with the spirit of agile. However there is not enough communication with the onshore site leading to misunderstanding about software requirements.
There are some problems being faced in validating changing software requirements by the offshore team at Aginity. The client (s)himself is not available to validate them. In agile methodologies the client is an important member of the team and plays a crucial role in validating software requirements particularly during the first iterations. However it is difficult to fully involve them in the agile team which makes it difficult to perform software requirements validation by them.
It was found that only at Ericsson English language is a hurdle in communicating with other sites about software requirements. They do not have very frequent long distance meetings between different sites which could increase this language problem. All of the other software companies do not have this language problem. The onshore sites of each of those companies

were located in an English speaking country. Therefore the language problem seems to occur only when both on and offshore site are non-English speaking and communicate in English.

ii. Domain and product knowledge

It was also confirmed that previous domain and product knowledge by both on and offshore teams facilitates in understanding software requirements. Since software requirements are not very detailed therefore previous domain and product knowledge provides that missing knowledge which is not present in the software requirements specification. The people working at higher and intermediate level between on and offshore sites such as system analyst and project manager agree that previous domain and product knowledge is helpful in understanding software requirements. On the other hand software developers disagree with it. In our opinion the software developers want more communication with the other sites about software requirements and not being dependent on previous domain and product knowledge.

iii. Less software requirements documentation

8 out of 9 respondents agreed that the current software requirements specification is enough for their site to understand it. High abstraction level of software requirements specification becomes a problem when there is lack of communication between on and offshore site.

iv. New challenges

New challenges were identified by asking an open ended question from the respondents about mentioning any challenges that they face at their site. These challenges fall under the same categories mentioned above but are given under the new challenges category because they provide a new insight into the existing challenges.
A new challenge identified was that the less detail of software requirements makes it difficult for the quality assurance team to perform major software testing at the offshore site.
Another challenge is not getting clients to review early builds. Thus offshore team has to wait for a long time to get a response from the client.
Transfer of tacit knowledge from the business (which is onshore) to the offshore team is a challenge as details of this tacit knowledge is not present in the software requirements specification.
Validation of software requirements by the offshore team due to change in business priorities is a challenge. It could be possible that not knowing the full business context leads to difficulties in software requirements validation by the offshore team.

Q2. How can these challenges be resolved to improve understanding software requirements in agile base offshore development?

The solutions below are not mentioned separately for the challenges above because these solutions are very interrelated to the challenges.
Regular communication needs to be established between on and offshore site to make transfer of knowledge easier. This regular communication is useful to resolve any type of conflicts and misunderstandings about software requirements. In shorter iterations lack of communication is bigger problem than in longer iterations due to physical distance between on and offshore teams and lack of time to implement software requirements. Thus either iteration duration could be increased or software requirements could be decreased for iteration to deal with lack of communication and difficulties in knowledge sharing.
According to all nine of the respondents' regular long distance meetings between on and offshore site will help in better understanding of software requirements. In agile

methodologies software requirements are not defined in a detail manner [7]. Thus regular long distance meetings between on and offshore sites should take place most preferably once every week. It will also solve the language problem as both of them will get acquainted with each other and developed a level of understanding between them. Long distance meetings at the start of iteration would resolve the initial misunderstandings about the software requirements. Once the offshore site gets software requirements specification they can analyze it and then have long distance meeting with the onshore site. All of the team members from on and offshore site should be present in these long distance meetings to discuss as many issues about software requirements as possible and to let all of the team members know about the overall status of the project. These regular long distance meetings would also facilitate in transferring tacit knowledge from the onshore site to offshore site.

According to all 9 of the respondents' face to face meetings between on and offshore site will help in better understanding of software requirements. Face to face meetings are much more effective than long distance meetings as a lot of issues can be discussed. However regular face to face meetings between on and offshore site may not be possible due to great geographical distance between on and offshore site. However a few of the team members may be sent from offshore to onshore site to have these face to face meetings and vice versa.

According to 7 out of 9 respondents, regular exchange of team members between on and offshore site will help in better understanding of software requirements by effectively transferring knowledge between on and offshore site. This regular exchange of team members particularly between on and offshore site would also help in resolving the difficulties of offshore team to validate software requirements due to change in business priorities of the client.

7 out of 9 respondents agreed that previous product and domain knowledge helps in better understanding of software requirements. Thus when creating teams, the domain and product experience of team members both at on and offshore site should be taken into consideration. In case of market driven software products, the team members should work with the same feature in the next release also.

Client has to be informed about their involvement in the process and be convinced that their active participation would lead to much quicker completion of their software product. Client should review every build of the software so that they can identify immediately any features missing or incorrectly implemented.

Regular test scripts should be created by the quality assurance team to communicate and understand software requirements. After getting the software requirements specification tests scripts should be written. It would allow the system analyst and quality assurance team on the offshore site to know what really is needed and can then ask the client relevant questions as they arise, from the onshore client.

Although this thesis shows that these challenges exist in the industry but their existence depends upon the place and process of the software company. This research work was of exploratory type therefore a much more detailed and comprehensive answer could be given by conducting interviews of team members at all of the four companies. However it does provide a starting point for software development companies to know that what type of challenges will be faced when adopting an agile based offshore process. The solutions for the challenges are also provided through responses from the software companies.

8.2 Future work

Below future research direction is given.

8.2.1 Validation of the research work

As discussed in section 8.1 this research work was of exploratory type therefore a much more detailed and comprehensive answer could be given by conducting interviews of team

members at all of the four companies. This validation was not possible due to the unavailability of respondents from these software companies.

9 REFERENCES

1. T.M. Rajkumar, R.V.S. Mani, "Offshore Software Development: The View from Indian Suppliers", Information Systems Management, vol. 18, Iss. 2, pp: 1-11, dated: 2001.
2. Muhammad Faisal Nisar, Tahir Hameed, "Agile development handling offshore Software Development issues", Multitopic conference proceedings of INMIC 2004. 8th International, dated: 2004-Dec-24-26.
3. Tony Gorschek "Requirements Abstraction Model", Requirements Engineering, 2006.
4. Tony Gorschek, Claus Wohlin, "Requirements Abstraction Model", Springer-Verlag London, dated:
26-11-2005.
5. Webpage: http://www.agilemanifesto.org/principles.html, last visited: 2007-01-19.
6. Alistair Cockburn, Jim Highsmith, "Agile Software Development: The People Factor", Software
Management, dated: 2004.
7. Frauke Paetsch, Dr. Armin Eberlein, Dr. Frank Maurer, "Requirements Engineering and Agile Software Development", Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), dated: 2003.
8. Ernest Ferguson, Clifton Kussmaul, Daniel D. McCracken, Mary Ann Robbert, "Offshore Outsourcing: Current Conditions & Diagnosis", Technical Symposium on Computer Science Education, Proceedings of the 35th SIGCSE Technical symposium on Computer Science Education, Norfolk, Virginia, USA, pp: 330-331, ACM Press New York, Ny, USA, dated: 2004.
9. Anandasviam Gopal, Tridas Mukhopadhyay, Mayuram S. Krishnan, "The role of software processes and communication in offshore software development", Communications of the ACM, Vol. 45, Iss. 4, pp: 193-200, ACM Press New York, NY, USA, dated: Apr-2002.
10. Avron Barr, Shirley Tessler, "The Globalization of Software R&D: The Search for Talent" Council on Foreign Relations' Study Group on the Globalization of Industrial R&D, dated: 12-Dec-1996.
11. Webpage: http://www.it.uu.se/edu/course/homepage/acsd/ht03/Fowler.pdf, last visited: 17-Jul-2008.
12. Robert C. Martin, "*Agile Software Development, Principles, Patterns, and Practices*", dated: Prentice Hall PTR, Upper Saddle River, NJ, USA, dated: 2003.
13. Ajay Danait, "Agile Offshore Techniques – A Case Study", Proceedings of Agile Development Conference, pp: 214-217, IEEE, dated: 24-29-Jul-2005.
14. Phillip g. Armour, "Agile…and Offshore", Communications of the ACM, Vol. 50, Iss. 1, ACM Press New York, NY, USA, dated: Jan-2007.
15. Gerald Kotonya, Ian Sommerville, *"Requirements Engineering Processes and Techniques"*, John Wiley & Sons Ltd, England, dated: 1998.
16. Tony Gorschek, "Requirements Engineering supporting technical product management", PHD thesis, Department of Systems and Software Engineering, Blekinge Institute of Technology, Ronneby, dated: 2006.
17. Joseph A. Goguen, Marina Jirotka, "*Requirements Engineering: Social and technical Issues*", Academic Press, London, dated: 1994.
18. Ann M. Hickey, Alan M. Davis, "Elicitation Technique Selection: How do Experts Do It?", Proceedings of the 11th IEEE International Requirements Engineering Conference, pp: 169-178, dated: 8-12-Sep-2003.
19. Bashar Nuseibeh, Steve Easterbrook, "Requirements engineering: a roadmap", Proceedings of the Conference on The Future of Software Engineering, pp: 35-46, Limerick, Ireland, dated: 2000.

20. Ann M. Hickey, Alan M. Davis, "" Requirements Elicitation and Elicitation Technique Selection: A Model for Two Knowledge-Intensive Software Development Process", Proceedings of the 36th Hawaii International Conference on System Sciences, dated: 2002.
21. Marielle den Hengst, Elisabeth van de Kar, Jaco Appelman, "Designing Mobile Information Services: User Requirements Elicitation with GSS Design and Application of a Repeatable Process", pp: 10-, Proceedings of the 37th Hawaii International Conference on System Sciences, IEEE, dated: 2004.
22. J. M. Coakes, E.W. Coakes, "Specifications in Context: Stakeholders, Systems and Modelling of Conflict", Requirements Engineering, Vol.5, No.2, pp: 103-113, Springer London, dated: 19-Feb-2004.
23. Colin Potts, Kenji Takahashi, Annie I. Anton, "Inquiry-Based Requirements Analysis", Vol. 11, Iss. 2, pp: 21-32, IEEE Software, dated: March-1994.
24. Bashar Nuseibeh, Steve Easterbrook, "Requirements Engineering: A Roadmap", Proceedings of the Conference on the Future of Software Engineering, pp: 35-46, Limerick, Ireland, dated: 2000.
25. Barbara Kitchenham, Shari Lawrence Pfleeger, "Principles of Survey Research Part 4: Questionnaire Evaluation", Vol.27, Iss. 3, pp:20-23, ACM SIGSOFT Software Engineering Notes, dated: May-2002.
26. Webpage: http://en.wikipedia.org/wiki/Likert_scale#cite_note-2, last visited: 28-Jun-2008.
27. Ikujiro Nonaka, "*Knowledge Management: Critical Perspectives on Business and Management*", Taylor & Francis, dated: 2005.
28. Jim Highsmith, Alistair Cockburn, "Agile Software Development: The Business of Innovation", Computer, Vol. 34, Iss. 9, pp: 190-127, IEEE, dated: Sep-2001.
29. Kent Beck, "Embracing change with extreme programming", Computer Journal, Vol. 32, Iss. 10, pp: 70-77, IEEE, dated: 1999.
30. Laurie Williams, Alistair Cockburn, "Agile software development: it's about feedback and change", computer Journal, Vol. 36, Iss. 6, pp: 39-43, IEEE, dated: 2003.
31. Graeme Martin, *Managing People and organizations in changing contexts*, Elsevier Ltd., dated: 2006.
32. Jeremy Howells, "Tacit knowledge, Innovation and Technology Transfer", Technology Analysis & Strategic Management, Vol. 8, No. 2, dated: 1996.
33. John W. Creswell, "Research Design Qualitative, Quantitative, and Mixed Methods Approaches", 2nd edition, Sage Publications: dated: 2003.
34. Keith Braithwaite, Tim Joyce, "XP Expanded: Distributed Extreme Programming", Springer-Verlag, pp: 180-188, Berlin, dated: 2005.
35. Joachim Sauer, "Agile practices in offshore outsourcing – an analysis of published experiences", Proceedings of the 29th Information Systems Research Seminar in Scandinavia, Helsingborg, Denmark, dated: Aug-12-15-2006.
36. Ken Schwaber, Mike Beedle, "*Agile Software Development with Scrum*", Prentice Hall PTR, Upper Saddle River, NJ, USA, dated: 2001.
37. Jawed Siddiqi, M. Chandara Shekaran, "Requirements Engineering: The Emerging Wisdom", Vol. 13, Iss. 2, IEEE Software, dated: Mar-1996.
38. Scott Amber, "*Agile modeling: effective practices for extreme programming and the unified process*", Wiley cop., New York, dated: 2002.
39. Pnina Soffer, Leah Goldin, Tsvi Kuflik, "A Unified RE Approach for Software Product Evolution: Challenges and research agenda", Proceedings of Situational Requirements Engineering Processes, dated: 2005.
40. Steve R. Palmer, Mac Felsing, "*A Practical Guide to Feature-Driven Development, 1st edition*", Pearson Education, dated: 2001.
41. Renee McCauley, "Agile Development Methods Poised to Upset Status Quo", Vol. 3, Iss. 4, ACM SIGCSE Bulletin, dated: Dec-2001.
42. Ashish Arora, Suma Athreye, "The software industry and India's economic development", Vol. 14, Iss.2, pp: 253-273, Information Economics and Policy, dated: Jun-2002.

43. Barry Boehm, Richard Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations", Vol. 22, Iss. 5, pp: 30-39, IEEE Software, dated: Sep-Oct-2005.
44. Lowell Lindstrom, Ron Jeffries, "Extreme Programming and Agile Software Development Methodologies", Vol. 21, Iss. 3, pp: 41-52, Information Systems Management, dated: 2004.
45. Daniel Turk, Robert France, Bernhard Rumpe, "Assumptions Underlying Agile Software-Development Processes", Vol. 16, Iss. 4, pp: 62-87, Journal of Database Management, dated: Oct-Dec-2005.
46. Clifton Kussmaul, "Agile Product Development: Lessons from Industry", The NCIIA 8th Annual Meeting, dated: 18-20-Mar-2004.
47. Kent Beck, "*Extreme Programming Explained: Embrace Change, Second Edition*", Addison Wesley Professional, dated: 16-Nov-2004.
48. Ron Jeffries, Ann Anderson, Chet Hendrikson, "*Extreme programming installed*", Addison-Wesley, Mass, Boston, dated: 2001.
49. Ken Auer, Roy Miller, "*Extreme programming applied: playing to win*", Addison-Wesley, Mass, London, dated: 2001.
50. Ming Huo, June Verner, Liming Zhu, Muhammad Ali Babar, "Software Quality and Agile Methods", Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Vol. 1, pp: 520-525, IEEE, dated: 28-30-Sep-2004.

Appendix 1:

Quantitative results from participant responses

The following table shows participant's response in percentage for each question based on Likert scale.

Question No.	Scale (Participant's response in percentage)			
	1	**2**	**3**	**4**
7	0	11	89	0
17	0	0	78	22
18	0	0	78	22
19	11	0	56	33
20	0	22	56	22
22	0	0	44	56
23	11	11	45	33
24	11	67	22	0
25	11	11	56	22
27	0	11	67	22

Appendix 2:

Responses to Question 27 of the online questionnaire

This appendix contains each respondent's response to question 27 of the online questionnaire. The question and responses are presented in its original wording.

Question 27 "Could you briefly describe three most common problems about software requirements that are faced at your site keeping in mind the on/off shore setting".

Respondent 1 (Project Manager Ericsson, China): Requirement dependency, due to poor anatomy planning. Solution requirement from solution level not clear enough. Sometimes too high level, not broken to detailed enough level.

Respondent 2 (Project Manager Ericsson, Sweden): Requirements are unclear and difficult to understand for design. Requirements Specification is unstable (preliminary version). Late changes in Requirements.

Respondent 3 (Software developer Shinetechchina, China): No answer.

Respondent 4 (Project Manager Shinetechchina, China): Unclear requirements document, unclear project schedule, slow response.

Respondent 5 (Software developer Shinetechchina, China): No answer.

Respondent 6 (Software developer, India): Communication. We need to communicate, communicate, communicate. If nothing is happening on either, its help to communicate saying nothing is happening.

Respondent 7 (Business Analyst Thoughtworks, India): No answer.

Respondent 8 (CTO Aginity, USA): the belief that agile means little or no requirements. Not getting clients to review early builds to validate requirements. The belief that QA people will handle the major testing (with little or no requirements).

Respondent 9 (System Analyst Thoughtworks, India): Ability of the offshore team to understand the full business context relating to software requirements. Transfer of tacit knowledge between the business and offshore team. Ability of the offshore team to validate requirements that changes due to business priorities.

Appendix 3

Abbreviations

Activity based abbreviations
Project Inception (PI)
Requirements Specification (RS)
Requirements Implementation (RI)
High Level Design (HLD)
Detail Design (DD)
Integration Testing (IT)
Testing (All kinds except integration testing) (TA)
Requirements change request (RCR)
Requirements change management (RCM)
Role based abbreviations
Project Manager(PM)
System analyst (SA)
Business analyst (BA)
Chief technical officer (CTO)
Software developer (SD)
Communication based abbreviations
Teleconferencing (TC)
Instant messengers (IM)
Telephone (TP)
Emails (EM)
Wikis (WI)
Webex (WB)
Web conference (WC)
Skype (SK)

Appendix 4

A Checklist of Questions for Designing a Survey Method

- Is the purpose of a survey design stated?
- Are the reasons for choosing the design mentioned?
- Is the nature of the survey (cross-sectional vs. longitudinal) identified?
- Are the population and size of the population mentioned?
- Will the population be stratified? If so, how?
- How many people will be in the sample? On what basis was the size chosen?
- What will be the procedure for sampling these individuals (e.g., random, nonrandom)?
- What instrument will be used in the survey? Who developed the instrument?
- What are the content areas addressed in the survey? The scales?
- What procedure will be used to pilot or field test the survey?
- What is the time line for administering the survey?
- What are the variables in the study?
- How do these variables cross-reference with the research questions and items on the survey?

  What specific steps will be taken in data analysis to

  (a) – analyze returns?

  (b) – check for response bias?

  (c) – conduct a descriptive analysis?

  (d) – collapse items into scales?

  (e) - check for reliability of scales?

  (f) – run inferential statistics to answer the research questions?