Research Journal of Computer and Information Technology Sciences ______________________ISSN 2320 – 6527
Vol. **2(5),** 1-15, December **(2014)** *Res. J. Computer and IT Sci.*

Review Paper

Requirement Engineering Process in Agile Software Development: Review

R. Mazhar Iqbal* and A. Waleed Abbasi
Department of Computer Sciences Bahria University E-8 Shangrilla Road Islamabad, 44000, PAKISTAN

Available online at: www.isca.in, www.isca.me
Received 19th August 2014, revised 15th November 2014, accepted 19th December 2014

Abstract

Agile has gained a lot of popularity nowadays and have become a major source of money for lots of organizations. We have compared traditional requirement engineering approaches and agile requirement engineering. In addition to this, discussion over here revolves around agile methodologies, requirement-engineering approaches, and how we perform traditional requirement engineering in agile software development. On basis of literature, requirement-engineering is discussed in detail in both traditional and agile software developments along with benefits and challenges and in addition to this, improvements/proposed ideas needed application of Agile in distributed development environments are proposed effectively.

Keywords: Traditional requirement engineering, agile software development, agile methodologies, RE and Agile, Agile review, literature review of RE and Agile.

Introduction

Before going into detail, let us have a quick discussion regarding requirement engineering. Requirement engineering is simply a process involving proper organization, documentation, and gathering of all requirements of a system[1]. In addition to this, it provides a well-documented agreement between projects' team members and clients, which states that any of the requirements can be changed depending on the nature of system. The term requirement engineering normally termed to be traditional requirement engineering.

Lot of organizations explicitly adopt traditional requirement engineering approaches. This proves to be the most challenging part nowadays. This is challenging because software organizations come across infinite software requirements, which arise unexpectedly and rapidly[2]. This creates problems for concerned requirement engineers because many of them are unable to handle such requirements i.e. users' issues, incompleteness or inconsistency of requirements, development tools, user expectations, time-to-market issues, conflict of views, communication gaps, unnecessary/unexpected changes, and lot more[2].

Agile, on the other hand, have become more popular and effective among IT experts and software organizations. Agile basically originated from Agile Manifesto [Beck etal., 2001], which was about individuals and interactions are preferred over tools, no or very less documentation, and embrace the change. This Agile Manifesto covers all development techniques i.e. eXtreme Programming, Scrum, and Rational Unified Process (RUP) [3, 4, 5].

The paper sections comprise of the following format: Section 2 discusses RE and Agile as a comparison. Moreover, all Agile and RE methodologies we have to target will be discussed in this section with a very brief discussion on each. Section 3 is concerned with a detailed agile requirements engineering process discussion on basis of literature study and it also comprises of a detailed survey report carried by Lan Cao and Ramesh[6], where comparison is made among all agile RE of sixteen software organizations along with benefits and challenges on basis of participants' comments and views. In addition to this, individual comparison of all those agile approaches is being made in general, which use traditional RE techniques in number of ways. Section 4 is about literature survey mentioning all existing knowledge regarding conduct of requirement engineering approaches (elicitation, validation, analysis etc.) in all agile software development approaches along with detailed discussion against each agile methodology. Section 5 will be composed of findings we gathered from complete literature survey. Section 6 comprises of all proposed ideas, which relate to gaps and present issues with respect to agile in developments scenario (on basis of literature) and last section i.e. Section 7, successfully concludes the paper.

RE and Agile: A Comparison

Let us briefly differentiate both of traditional requirement engineering and agile methodologies. i. Requirement engineering totally emphasizes at properly gathering organizing and documenting all requirements and excludes any live meetings/conferences. This is what Agile lacks. Agile emphasizes on minimum or no documentation and involves many face-to-face meetings. ii. In RE, developing requirements completely and consistently is due to communication problem because both users and development team members lack communication but agile lets developers develop requirements as user stories i.e. use cases. These use cases are written in natural language, provide functional requirements, focus on user intentions, and include "what" features instead of "how". iii. The worst thing of RE is" shall" argument i.e. system shall do it, and interface shall be manageable etc. On the contrary, Agile is real-time system, which involves statements as if I want a system, which can be easily manageable.

Many differences do exist but we have mentioned very few of them. All of these are of prime importance especially whenever there is a discussion regarding software development projects.

Agile Methodologies to Target: In early days, when there were very few companies the problems were very few regarding requirements and in addition to this, development was very easy too. This was because there were very few requirements and changes in requirements very easy to manipulate. With the passage of time, more companies evolved and field became vaster, this gave rise to more complex issues regarding requirements and development. These included requirements overload, no customer's engagements, and lack of effective communications, meeting deadlines, budget issues, and much more inconveniences[1]. To overcome these issues and difficulties, IT experts were introducing Agile Manifesto. This Manifesto focussed on delivery of working product to user and later on making necessary changes as per his needs and expectations. Agile Methodologies primarily emphasize on software development instead of documentation. They successfully accelerate project deadlines, response to customers' changes, satisfying users, rapid development, and much more[3]. Agile methods exist too much in number but few most common of them include: i. XP commonly known as Extreme Programming, ii. Methodology of Agile Modeling, iii. Agile SCRUM Methodology, iv. Agile FDD, iv. Agile ASE, v. Agile LSE Lean Software Development

Agile Extreme Programming (XP): Kent Beck gave idea of XP in 2000. Main focus of XP is on effective communication and rapid feedback[7]. It involves contribution from entire team along with their collaborations and feedbacks. XP discusses whole development process along with its future prospects. XP provides us with four major software development phases i.e. planning, coding, designing, and testing. The best advantage of having XP is that it welcomes changes to requirement anytime during development and timely delivery of project is always ensured by this methodology[8].

As far as XP values are concerned, XP focuses on four basic values, which include effective communications/discussions, simple designing, change of requirements whenever needed, and correct requirements' gathering[8].

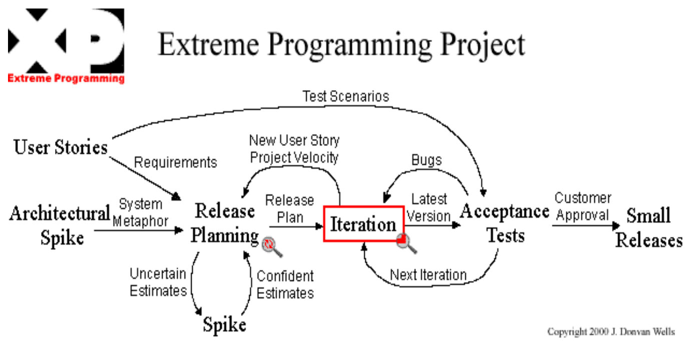

Figure-1
Project Flow in XP [8]

XP programming principles include simple and flexible behaviours. These two factors tend to cause a lot of reduction in software maintenance costs. Other principle is deep testing procedure or mechanism. This is also called intensive testing of software, which surely reduces bugs, which may occur after delivery of software to customer[8].

These are very few rules we have listed. Few other principles of XP include creating user stories, making small releases, iterative divisions of projects, pair programming, creating test cases, and much more[8, 9].

XP never suits to all projects or all companies surely. Limitations are always there. Many organizations having very large teams cannot deploy XP because XP needs only a team of maximum 12 persons. In addition to this, coordination and coordination among team members as well as client must exist.

Agile Modeling

Whenever you sit for developing any software, the most primary step, which must never be forgotten, is modelling. It is considered the most important activity because it opens eyes of IT professionals by providing them with a lot of complex issues and problems before proper implementations of softwares.

This is performed naturally before programming because programmers perfectly think of those critical situations, which may be risky in future. Scott Ambler in 2002 [7] gave a wide concept of Agile Modeling by providing professionals with detailed modelling software comprising of all those values, rules, experiences, practices, and procedures that are effectively responsible for successful implementation of any project related to software development [7].

Basic aim of Agile Modelling was to develop software, which would successfully meet customer's needs their maximum and developers were using it by combining with XP and RUP (Rational Unified Process).

Many IT experts call Agile Modelling an extension of XP because it extends moral values as well as principals of XP, which include communication and, courage factors primarily. Do not get confused with humility issue because here it means that you do not know anything but others know a lot more than you do. Therefore, they make valuable contributions to your project[10]. In same way, principles of Agile Modeling are also same as those of XP, which include rapid responses, feedbacks, quick changes, faster results, infinite modeling models, authentic reasons for modelling, effective and beneficial meetings among development teams, and much more[10].

SCRUM: Ken Swaber did initial start of SCRUM. He initiated this superb methodology in 1995. Right before Agile Manifesto, SCRUM methodology was thoroughly tested and checked. Due to similarity in rules, regulations, and principles, SCRUM joined the list of Agile Methodologies very soon. It manages and supervises software projects so cleanly and perfectly that application of simple processes is assured completely and easy documentations are preferred over high-value or much exhausted documentation scenarios[11].

Project management is considered a basic practice of SCRUM, which effectively lets developers to perform required tasks at various development stages. SCRUM successfully provides professionals with a process known as backlog. Backlog is defined as an area or simply a place, which is comprised of all those requirements, which remain in pending or unapproved state for specific projects. Simply there lies a single statement against each requirement, which is taken under supervision and consideration of development team for better discussion about needed details about implementation procedures[11].

The three persons i.e. Product owner, Members of SRUM i.e. team members, and SCRUM master are primarily responsible in this regard. Product owner is the major personality in this scenario and he is the only one who speaks his own business. SCRUM team is a well-dedicated and matured team comprising of developers, testers, designers, professionals, and much more other roles. This SCRUM team is responsible for direct communication with clients against each product regarding detailed requirements. Last but not the least; SCRUM master keeps all of team members thoroughly united and focussed on their goals along with better solutions of their problems or development issues. SCRUM is concerned with phases of development, which means that all development, testing, and finishing would be done in each phase. Team cannot proceed to new phase until or unless they do not complete one phase thoroughly. Literature has verified that SCRUM fails to define usability. This is because product owners emphasize on business values and they mostly do not care about usability issues. Due to much business hunger, they totally ignore usability factors, users' experiences, designing issues, motivations, and skills[12].

Agile FDD (Feature-Driven Development): Developing any system is made easy using Feature-Driven Development i.e. FDD. FDD has its major focus on development, designing, and building processes. The most amazing fact about FDD is that it does not need any software model for development.
As far as processes of FDD are concerned, FDD is composed of five (5) major processes, which are naturally followed in a sequence. During these five processes, effective design and development of project is made[13].

Palmer[13] discusses these five models as: i. Over All Model Development, ii. Building list of features, iii. Features for planning, iv. Features for designing, v. Features for perfect software building.

Knowing all requirements, scope, and context, development teams construct a detailed model for the system by a continuous presentation of a "walkthrough". This walkthrough illustrates detailed descriptions of a system to all team members and chief architects. Necessary object models are discussed and finally a model is proposed[13].

Model, which is being built previously provides professionals with detailed features' list having all necessary and mandatory features if a system, which is to be developed. All clients' needs, functions, and requirements are provided in this list. These features give rise to a feature set, which is successfully tested by users for proper validation and completeness.

This planning through feature step includes a complete and thorough high-level planning. This planning involves prioritization of features, which are later assigned to professional programmers.

Design and Build by Features include proper selection of very small features from feature sets and those teams who make features selections are called feature teams. Class owners develop these teams. Few selected features are produced during these iterative procedures of design and build. Multiple teams work simultaneously for effective designing and building of features, which include coding, unit testing, code or design inspections, and much more. After successful results, finished features are being put into main build and iteration cycle re-continues within a new set of features[13].

Talking about FDD, one must never ignore roles of individuals. Major and prime roles of Feature Driven Development are role of project managers in some organization, Role of chief architects, Role of development managers, role of chief programmers, Role of class owners, and finally the role of domain experts. In the same way, supporting roles are total five in number, which include release manager, lawyer or language lawyer, system administrator, build engineer, and toolsmith. Few other roles include testers, deployers, and technical writers[13].

Finally, practices are necessary in FDD. These include Domain object modelling or DOM, Coding, Feature Teams, Regular Inspections, Builds, Configurations, and Progress reporting.

Adaptive Software Development: This development methodology provides a platform or roadmap for iterative development of large systems. In this methodology, simultaneous prototyping, iterations, and incremental developments are performed[14]. This was an idea of ASD that waterfall approach would only work for properly-defined environments. We all know that changes are always welcome in software scenarios so there must be a system, which should effectively manage such changes. Each cycle of ASD finishes with focus group review of clients. During these review meetings, developers and experts find a finalized application. The possible outcomes of these meetings are changed or modified requests. ASD never refer any schedules for JAD sessions.

ASD is carried in three successive phases. These phases are speculation, collaboration, and learning. Speculation is the second name of planning about uncertainties or weaknesses. Collaboration surely involves teams' contributions in project along with their importance and learning always refers to mistakes or non-repetition of mistakes along with proper changes in requirements [14,] page 18.

For detailed description regarding these three phases, we must keep in mind that speculation is a planning for future results. If you have any uncertainty in your project or product or you feel anything, which may prove it poor, you must follow this speculation feature of ASD. Similarly, communication and sharing ideas among others is something good. Collaboration over here refers to regular and effective communication among team members. This may include anything like discussions, reviews, focussed groups, personas, strategic plannings, and much more. Collaboration has been the most important factor in software industry always. Training of employees or users is equally important to that of tools and equipments. This means that if you are well trained, you can produce fruitful results. However, if you are not well trained or badly trained or not enough trained, you would not only suffer alone but also your whole project along with your organization would suffer more than 50 percent of before. Therefore, learning is mentioned here in ASD. Learning would teach developers in so effective ways that they would not ever make repetitions in future; these reputations may include anything like coding, designing, or any issue. Therefore, learning is important in each aspect of life.

ASD mainly focuses on results and outcomes instead of tasks and activities being performed. ASD does not actually specify general roles or responsibilities explicitly because it is all teamwork. One-person i.e. executive sponsor is completely responsible for perfect development and deployment of a product. Yes! JAD sessions may define some roles like project manager, team lead, facilitators etc.

ASD is composed of highly professional and talented teams because it focuses on large systems' development. Therefore, for large or complex system developments, IT professionals are hiring special trained teams so that they do not let company suffer in future.

If you need to divide these phases, you must then make divisions, which include project, initiation and adaptive cycle planning in first phase i.e. speculation. Concurrent component engineering is being added as the part of collaboration phase [14] page 84.

Every professional first trains his or her staff, which is mandatory for proper system use and knowhow of a system. This is because if someone is a newbie, he or she may come across that system soon. Quality review, final questions/answers and successful release of a system is included in learning phase respectively[14] page 84.

Jim Highsmith and Sam Bayer worked at rapid application development and proposed idea of adaptive systems development. This methodology is concerned with basic adaption of any process during software development[15].

Lean Software Development: As far as lean development is concerned, it is composed of two major parts. First, one is fastest process to attain best quality flow and the other one is a process, which captures knowledge for better repetition, quality increase, and reusability in iteration cycles. Few steps need to be taken in consideration for this purpose (These are provided on the basis of product development model of Toyota)[16]. i. Strategies of development must be kept separate or different from execution stages. ii. All processes must be managed and maintained differently with regular checks. iii. Meetings need to be held for discussing scopes of each iteration phase. iv. Organization must collaborate in all departments and learning process must be effectively increased. v. Interrupts need to be handled with extreme care.

Traditional product development and lean development are two different approaches[17]. Lean development is usually concerned with prototyping and testing phases. On the contrary, traditional development involves design freezing in very early stages. This means that design gets ready as soon as possible and is delivered to concerned persons early in morning [17, 18]. Quality and usability factors played a vital role in breakdown of Lean in manufacturing industries. Traditional business practices also failed and had no comparison with Lean. Lean is no doubt unique and different from traditional approaches because traditional approaches focus on linear development but lean focuses at planning and development of one complete feature before next feature[18].

The word Lean has come from manufacturing industry[18]. Toyota first introduced Lean when there was a dire need of Lean. The basic drawback of Lean was that it would cause business loss more than 90% if a very small single mistake were made. The major problem, which occurred in manufacturing industries, was that customers would demand more than before.

In addition to this, they would expect too much and as a result, their satisfaction would never be there. This refers to a fact that customers' values were discouraged and they were facing many problems. In the same way, manufacturing companies were also facing many problems i.e. management issues, manufacturing issues, requirement issues, and much more[18].

Let us have few basic principles of lean development[19]. i. Value specifications must be from customer's point of view. This is because customer satisfaction is of prime importance in all cases. ii. All steps in value set must be identified by explicit elimination of each step, action, and practice. iii. Product flow must be made smooth and fast. iv. Greater transparency is required and perfection is needed instead of wastage[19].

Approaches for Requirements Engineering: Process of requirements engineering includes a detailed, well-managed, organized, and well-documented requirements, which one needs for a system. These requirements do let us know about what actions are to be taken but how things will work is never discussed by requirement engineering process[20]. Requirement engineering is also called as traditional requirement engineering process and it usually occurs in early stages of some software project. The most critical and widely used technique of requirement engineering process is requirements management. Infinite definitions and descriptions regarding requirement engineering process do exist but Sommerville and Sawyer has defined requirement engineering very uniquely[21].

"A requirements engineering process is a structured set of activities which are followed to derive, validate and maintain system requirements document."

The three best techniques or generic activities of requirement engineering are extracting requirements or requirements elicitation, analyzing requirements deeply and carrying out perfect analysis, and validating all needed requirements properly. Later on, these activities are supported by requirements management[1, 21].

Let us have a quick review regarding these three basic activities of requirement engineering process along with few issues[1].

Requirements Elicitation: This phase i.e. requirements elicitation phase successfully gathers all desired requirements and provide engineers with necessary system domains or scopes by constant collaboration with clients or customers or stakeholders. These domains can be anything like discussions, system constraints, values, application domains, and much more. Elicitation phase gathers or elicits all requirements using techniques i.e. Interviews, Use Cases, Observations, Focus Groups, Brainstorming, and Prototyping.

Requirements Analysis and Negotiation: Requirements Analysis phase analyzes all of the requirements in detail. This includes their completeness, traceability, mature deign, testability, feasibility, necessity, consistency, and many other factors. If there are any conflicts in requirements, they are successfully solved by negotiation process. This process involves both customers and developers sit on a table and discuss all issues regarding requirements. Main techniques being used by professionals are JAD sessions, and Modeling.

Requirements Validation: Requirements validation phase validates all of system's requirements. Are these requirements valid for system, which we are going to develop? Are they acceptable? All those necessary works or tasks, which we need to perform during validation phase, are proper documentation of requirements, complete standards of organization, and detailed organizational knowledge. These are said to be inputs actually for validation phase. All those results or outcomes, which we must expect include all reported problems and necessary steps, which we have to take to overcome these issues or problems. Proper testing of requirements and necessary requirements' reviews are considered the tools of requirements validation phase.

Requirements Engineering Issues: Requirement engineering is considered to be the most critical and primary phase for any software project. If requirements are not complete, consistent, accurate, precise, or accurate, whole project suffers, which effectively create infinite issues and problems for projects' team members. Therefore, traditional requirement engineering needs to be performed accurately. Let us have a quick overview regarding those issues, which normally arise during requirements elicitation, specification, and validation/ verification phases[22].

Requirements Elicitation Issues: Requirements elicitation provides professionals with issues, which are sub-divided into three main categories. The very first category is concerned about scope i.e. requirements may either lie outside of boundary or projects' scope or they may define too much or too less information. Second category is concerned about communication or understanding factors, which includes lack of knowledge or understanding or access issues regarding developers and customers. The last category refers to requirements' changing or their constant varying behaviours. This also leaves bad impact on developers or technical professionals. IT experts try to solve these issues by constant interviewing, discussions, round-table meetings, use cases, and prototyping[23].

Requirements Specification Issues: This phase i.e. specification is said to be the centre of requirements engineering process because it lies in centre like elicitation, specification and verification/validation. The major issue, which arises in requirements specification, is that their proper documentation or better understanding is not done. How will system work and how system will behave etc are some common issues. For example, if you have to develop a system, which rings the bell when door opens. Now, is this information complete? No. This is because one must know what would be tone of bell, which will open the door, what would be system's domain, and many other related factors. The main objective is to provide unambiguous, complete, consistent, and testable requirements. Such ambiguities were solved by concept of use cases by Booch in 1999. In addition to this, many IT professionals propose that putting all requirements in natural language is the best solution of such requirements' specification issues. Still if you are unable to record all requirements due to their complex natural language behaviours, you can put them down using latest methodologies i.e. Pseudcodes, Finite State Machines, Decision Trees, and much more[22, 24].

Requirements Validation/Verification Issues: Requirements verification occurs before validation phase and not vice versa. Validation is concerned with evaluation of product itself but on the other hand, verification includes planning, coding, and proper documentations. Issues, which arise during these phases, include traceability issues, inactive management of requirements, testing issues, and much more. This can be overcome easily by tracing techniques, to assure that development is on right track, and changes are well dealt.

Agile Requirements Engineering: Lan Cao and Ramesh analyzed sixteen software development organizations and finally provided us with seven major Requirement Engineering practices, which as follows.

Agile RE in Various Organizations: For better understanding of concepts, i.e. how requirements engineering of Agile differentiates itself from traditional requirement approaches, numerous organizations apply Agile RE in their own ways. For this purpose, Lan Cao and Ramesh[6] visited more than sixteen software development organizations and only those, which apply agile approaches. Out of total sixteen organizations, the first ten of them were considered the highest participants in Agile. By highest participants, it was meant that these 10 organizations did not follow some certain "brand" of agile but they followed some of those practices, which agile methods suggested. Agile methods like Scrum or XP suggested those relevant and close practices. The other six organizations were those, which exactly followed agile methods like XP, Scrum or both of them[6].

How Data Was Collected?: On basis of proper reviewing of all documentations, semi structured interviews, and participants' activities, data was collected, extracted, and synthesized. In all sixteen organizations, there were interviews of top-level managements, quality assurance teams, developers, designers, product managers, team leads, and project managers. Research domain was limited to qualitative methodology with respect to data collection.

How Data Was Analyzed?: Data analysis totally based on grounded theory methodology[25]. This methodology is a well-structured qualitative research method. Data analysis was performed in three different ways i.e. open, axial, and selective coding [6, 25].

Open coding method allowed us to find out major data groups and label them as per their frequency. These frequencies include agile RE practices, agile RE positive impacts, and agile RE issues. The other method i.e. axial analysis involved successful building of relationships among those found practices, benefits, and challenges of agile RE. Final coding analysis method compared agile RE practices and provided professionals with larger patterns. In this process, two coders performed their individual jobs and results were compared later, which effectively resolved all differences by detailed discussions. Finally, data was recorded mentioning all common RE practices across those sixteen organizations.

One should be keep in mind that Agile methodologies totally depend on requirement engineering generic activities for gathering and managing requirements. Many variations for requirement engineering approaches do exist in agile environments. Whole teams in agile environment collaborate among one another and gather necessary requirements with constant communications with clients[26]. Also, it should be noted that documentation in both Agile RE and Traditional RE is dependent a size of team an organization has. If size is small, it is ok and you can manage with small documentations but if size is explicitly large, it would be almost impossible for you to make one single thing understand multiple times to multiple users or people[27].

Agile has its main objective of protection against wastage of requirements, which is commonly known as Lean Software Development as discussed in later section in thorough detail [28].

Identification of Agile RE Practices: On basis of research and constant participations, Lan Cao and Ramesh[6] successfully identified seven agile RE practices common in those organizations. i. Front end face-to-face Communications, ii. Iterative requirements Engineering, iii. Priority based requirements engineering, iv. Requirements Change Management, iv. Constant prototypes and communications, v. Test based software development, vi. Review Meetings and Acceptance Tests.

Agile RE Practices, Benefits, and Challenges: This will discuss all of the Agile requirements engineering practices in detail along with their pros and cons. This would all base on participants' views and comments.

Face-to-face Communications

According to participants, agile RE discourages documentation and encourages exchange if ideas among customers and developers. Therefore, preference is always given to face-to-face discussions for effective requirements engineering instead of documenting anything.

Positive Aspects: All sixteen organizations had no documentation procedures implemented and would depend on face-to-face discussions. By participants' comments, this is beneficial because requirements can be changed at any stage and customers can accomplish their projects in unique ways. Moreover, these communications prove to be more beneficial than documentations because they save time of both customers and development teams.

Negative Aspects: The worst thing reported by participants was constant interaction. This meant that one has to interact with one another and this high-quality interaction is not always possible. This can give rise to serious issues regarding requirements and requirements gathering may suffer from this. One more thing needs to be mentioned over here that face-to-face communication is always based on customer's access, customer's attitudes, and trust factor among customers and developers. Lacking a single out of these can be prove to be a great challenge in face-face domains of agile requirement engineering.

Iterative Requirements Engineering

In fourteen out of sixteen organizations, requirements are not well-defined. All of the requirements evolve during development stages. Iterative requirements engineering is concerned with requirements gathering in each development cycle. Most of the organizations follow this iterative strategy. Their requirements engineering processes start at each development cycle[29]. On the start of each successive cycle, customer accesses development teams and provides all necessary implementation strategies they should follow. He provides them with all those features, which are to be added during implementations or necessary coding strategies. All requirements are being discussed in thorough detail among clients and developers and in the end, both of them come up with some implementation plans, fine-grained requirements, or preliminary designs.

Positive Aspects: As far as reported benefits are concerned [6, 25], this iterative RE methodology builds a strong relationship among customers and developers. Moreover, all of the requirements in this phase are crystal-clear, complete, finely grained, precise, accurate, and consistent because of easy and early access of customers along with their effective participations.

Negative Aspects

Participants were discussing three major issues. i. One of them is cost and schedule estimation issues. Requirements evolve during developments and constant interaction is going on, which results in improper cost/size and schedule estimations. This proves to be the biggest challenge in agile RE. ii. The other reported challenge is of less or no documentation. This is because when there is an unexpected lack in communication anytime like any personal issue, no access, or any issue, having no or very less documentations can be problematic. This causes problems in software metrics, proceeding of process until one get access to customer or development team, or anything, which needs to be reviewed from documentation but not discussion. iii. Third most important reported challenge is ignoring all non-functional requirements. Usually customers ignore non-functional requirements and rely on functional ones only. Few of them include maintainability, performance, scalability, portability, and much more.

Requirements Prioritization

All organizations successfully prioritize their all requirements during development with continuous evolvement of requirements. Their prioritizations are done after when they are added or modified. In agile requirements engineering, requirements prioritization process works quite differently than traditional requirements engineering prioritization. In traditional RE, requirements are prioritized only once but on basis of sixteen software development organizations, it is proved that agile RE prioritizes requirements in each development phase. In addition to this, requirements prioritization in agile involves bug fixing, changes to existing functionalities, and refactoring techniques. One more reported difference between traditional RE and agile RE is there i.e. traditional RE prioritizes requirements on basis of many factors like risk, performance cost, implementation etc. However, in agile RE prioritization only depends on one and only factors i.e. only those business values, which are defined explicitly by customer.

Positive Aspects: Development teams better know what customers want. Therefore, prioritization is made on basis of customers' needs and satisfactions. Moreover, interaction between the two parties also helps in better prioritization of these all requirements in agile RE. Moreover, agile RE allows developers to reprioritize all requirements, which traditional RE never provides.

Negative Aspects: Business values are surely important but keeping them as primary aspect of prioritization is quite challenging in agile RE. Many issues may arise like scalability issues and requirements' accommodation issues. In addition to this, continuous reprioritization may also lead to instability issues.

Change of Requirements

Requirements change is concerned with development as per users' needs or satisfactions. Participants reported two types of requirement change mechanisms. These included adding/dropping necessary features and also making a small or big change for already implemented features as per needs. At the end, customers can provide developments team with a feedback and can request any change if it s not met.

Positive Aspects: Constant and early validation of requirements never involves many changes. Maximum change can be of some spellings issues, colour issues, and much more. Therefore, cost of change requirements decrease significantly.

Negative Aspects: Change in requirements is very less in agile RE but the major issues being faced by both agile members and customers is of refactoring. This is because refactoring always claims that software's original behaviour would never change but if there is a need of refactoring in agile RE, one has to re-write whole code from scratch. This is because participants claimed that refactoring never provides issues regarding inappropriate architecture or inadequate structure. Therefore, developers have to throw away codes and re-write from scratch.

Prototyping

Many organizations try settling requirements specification very quickly. This helps them in eradicating all errors related to software. Software organizations intend to develop those softwares, which are their self's operational prototypes and completely refined codes having all necessary features.

Positive Aspects: Instead of complex documentations, many software development organizations rely on prototyping technique to better involve their customers/clients in validation and verification scenarios. Eleven out of sixteen organizations successfully used prototyping methodologies.

Negative Aspects: Many organizations faced prototyping issues regarding scalability, security, and robustness. Many customers stopped to accept longer software development life cycles, which were mandatory for proper and robust implementations.

Test-Driven Development

In this mode of development, developers usually write test cases before implementations or functional coding. These test cases provide you with a brief idea about the system you are going to develop. Writing explicit specifications also come under its domain in agile RE.

Positive Aspects: In agile environments, many software companies, by making use of test cases, try to attain all requirements related to codes or necessary implementations. This makes requirements changes easy and you can come up with new and fresh ideas.

Negative Aspects: TDD is surely important and beneficial but most of organizations are unable to follow this practice. This is because developers do not feel comfortable with it. They claim that it needs a lot of discipline, constant flow, and regularity and most of them are not habitual of writing test cases before coding. One more challenge, which is reported by participants i.e. TDD needs in-depth understanding of all requirements because it refines all low-level specifications in iterative ways.

Review Meetings and Acceptance Tests

Review meetings are held by many software organizations for the purpose of requirements validation. These meetings usually occur when software development ends. Professionals involved in these meetings are developers, designers, quality assurance experts, testers, management's personnels, and many other stakeholders. Developers provide a quick demonstration of products along with their core features and relevant questions are being asked by quality assurance experts and customers.

Acceptance tests are also considered tools for requirements validation and verification. Few software organizations consider acceptance test to be the part of requirements specifications phase.

Positive Aspects: The main purpose of these meetings is to get feedback against developed products. At the end of meeting, detailed report is provided to stakeholders and Quality Assurance departments, which is comprised of status if project, project main goals and objectives, factors that help in attaining customer trust, and properly identifying all development issues.

Negative Aspects: Participants claimed that this agile RE practice mainly focuses a lot on requirements validation, which was never the case in traditional requirements engineering approaches. No formal verification methods are clearly addresses and no consistency checking occurs during this phase. QA personnels develop acceptance tests because implementing these acceptance tests in most of organizations is difficult.

Comparing Agile RE Practices: All sixteen organizations were clearly examined and interviewed. Table below mentioned details all practices and number of organizations flowing them.

Table1
Agile requirements-engineering practices in 16 organziations[6]

Adoption Level	Face-Face Communication	Iterative RE	Extreme Prioritization	Constant Planning	Prototyping	Test-Driven Development	Reviews and Tests
High	8	9	10	8	8	5	11
Medium	8	5	6	6	3	1	4
Low	0	2	0	2	0	0	1
None	0	0	0	0	5	10	0

Table above provides us with a detailed overview regarding each practice being applied in all software companies. This shows that which practice is adopted by how many organizations. For example, reviews and tests are performed in eleven organizations at higher levels but four at medium levels and in the same way, ten organizations follow extreme prioritization techniques at higher levels and so on. All comparisons are made at high, medium, and low levels, which is basically the degree of following of each organization for each practice. Levels are decided as per organizations needs and resources and also it is decided in the basis of challenges being faced by each organization. If any challenge is faced by any organization against each practice, it would be placed in suitable level depending in its implementation strategies.

Is Agile RE Preferable Over Traditional RE?: By surveying these software development organizations, we have come to know that basic feature, which distinguishes agile RE from traditional RE is iterative development. It never means that agile RE is strongly preferred over traditional one because one can never be sure about agile's unexpected outcomes. Agile provides quick responses and rapid productions but its challenges must never be ignored. This is because agile never suits to all organizations. Therefore, software organization has to decide itself depending on cost, schedules, efforts, and resources that which methodology or practice they have to adopt for betters accomplishment of their software projects as well as success of the company.

Comparable RE Techniques used by Agile Methodologies (Individual Comparison): All generic activities of traditional requirement engineering i.e. requirement elicitation, analysis, and validation do take place during agile methodologies. Discussion is further carried out regarding such scenarios.

Agile applies these approaches if size of their teams is very small and in addition to this, agile suits to small projects only. If bigger projects are over there, teams would be making use of some other suitable techniques for requirements gathering along with requirements management. One must keep in mind that discussion in this section is not done about requirements engineering techniques in agile methodologies but individual agile methodologies are discussed along with their major tools or techniques they use for requirements engineering[30].

Scrum: Before discussing requirements engineering process, we must give a close look at Scrum. It is an agile methodology, which effectively maintains and manages complete development of a system. All factors, which affect this agile methodology are flexibility, maintainability, reliability, and adaptability. The techniques being adopted by major Scrum are mainly product backlogs, sprints, and daily scrums. Scrum emphasizes on team's work. Collaboration, and hard work to provide excellent results to software development industry. Whenever there is a discussion regarding requirements engineering process, product backlog is of primary importance. All of the main characteristics of a product, further extensions, more advancements or improvements, bugs, and core functionalities are mainly are contained by this product backlog in prioritized manner. Sprint is defined as n iteration of 30 days. Product backlog is also known as requirements changing container, which comprises of changed requirements. After each 30 days iteration, all those tasks, which are highly prioritized, are

transferred and technically moved from one phase to another i.e. product backlog to sprint backlog. During development phases or sprints, nobody is allowed to make any change in requirements but users can change or modify requirements as per their needs by again prioritizing them or reprioritization process for next coming sprints. At the end, a grand round table meeting takes place, which is known as sprint review meeting. In this meeting, customer is being presented a demonstration by senior members about product's functionality and features and customer provides them with effective and usable feedbacks[1].

Requirements engineering in Scrum is done using various techniques mentioned below.

Time Boxing: Time boxing provides professionals with lo of meanings. A time box is basically a standard time period for set of various activities. In scrum, time boxing is used for minimizing work-loads by work break down procedures i.e. sprints.

Face-to-face: Better and more effective communication takes place in Scrum, which involves both Scrum members and product owner. This is one of the prime factors for handling requirements engineering in SCRUM.

Late Decisions: Having late and deferred decisions seriously affects requirements gathering and management in Scrum. In these cases, requirements are modified and developed very late. As a result of this, development is affected providing poor impressions to clients.

Changes in Requirements: Product backlog changes during SCRUM requirements engineering.

Extreme Programming (XP): Extreme programming always emphasizes overall software development process and all those tasks and activities, which are mandatory during software development project. Many requirement engineering techniques are widely adopted by XP in comparison to traditional one.

XP makes best use of interviews and prioritization. The major requirements engineering technique being adopted by XP is story cards. Story cards' concept can be compared easily to requirements elicitation phase. We all know that XP relies only and only on business values for their customers. Experts say that story cards must never be compared to use cases. This is because use cases are just user interactions with system but story cards comprise of almost everything, which users need.

What system should do, how system should behave, how much effort is needed, how much time will it take for completion, and much more similar scenarios are being written in thorough details, which give users a brief overview of a system by complete brainstorming allowing them to make prioritization of stories for coming iterations.

One more technique, which XP adopts is of acceptance tests. These tests are defined and verified by customer himself for successful story accomplishment. XP relies on very short releases, which distinguish it with prototyping concept. The major difference between the two is that XP provides users with a tested and completely clean codes but prototyping is a scattered format of requirements and can be hacked. Users can effectively have detailed discussions among them regarding all issues they usually face for next releases[1].

Agile Modeling (AM): Basic purpose of agile modelling is to develop necessary models relevant to software development projects. No specific requirements engineering tools are used by this methodology but very few of its practices make best use of tests and brainstorming. Face to face communications are made possible by using certain models being developed by this AM methodology. Moreover, all those standard models, which are followed for documentation purposes are also included in this domain[1].

Agile Crystal Methodologies: Agile crystal methodologies are mixed with many methodologies out of which any suitable methodology is chosen for development process. Requirements engineering techniques being adopted by these crystal methodologies include prototyping, testing, product workshops, overall product reviews, and reviews per each product's release.

Feature Driven Development (FDD): One must keep in mind that Feature Driven Development methodology i.e. FDD is never concerned with overall system's development but it is only focused at software's design and implementation phases[26]. FDD is basically concerned with domain model building where domain experts build domain models, which comprise of major attributes, needs, functionalities, classes, and relationships are mentioned. Feature is a terminology used n FDD for client's functionalities or core features. Weekly presentations of 30 minutes are given to clients for demonstrating current status of features and detailed report writing. This report writing is a requirements engineering technique being used in FDD[1].

Dynamic Systems Development Method (DSDM): The two major requirement engineering techniques adopted by Dynamic Systems Development Method or DSDM are feasible and business studies. Both of these techniques allow professionals to perform requirements elicitation of base requirements. All other requirements are gathered during software development process. You must know that DSDSM has no specific guidelines or limitations for making use of requirements engineering techniques because any RE techniques can be induced within software development process suitably. Testing techniques and JAD sessions are widely applied in DSDM[1].

Agile Adaptive Software Development (ASD): Adaptive Software Development of Agile i.e. Agile ASD is all about iterative development projects. It is usually a structured framework for perfect development of very large and complex systems so that project should never lead to failures. The three major techniques being adopted are speculation, collaboration, and learning. Speculation is all about planning or requirements management, which allows professionals to plan for future prospects. Collaboration is similar to customer involvement because customers and team members collaborate among themselves to find out necessary conflicts and issues along with necessary agreements. Learning is concerned with training or change of requirements during development process, which means that developers or customers learn from their pas discussions and discuss more for better results[1].

Lean Software Development: This is much difficult to do because identifying and reducing wastage is much time-consuming. In lean software development, requirements once started wasting lead to further fluent wastage in later stages. It can be understood by simple example. If an industry produces more than it is needed, it would be complete wastage along with loss of resources and energy usage. In perspective of requirements engineering, in early stages i.e. gathering or elicitation, Agile allows customer interactions or face to face interviews, which cause basic requirements evolution sometimes. This increases costs and more resources are uses. As a result, results being obtained are more than required. To overcome this, waste is handled in early stages. In agile environments, for lean software development or management of wastage of requirements is done by making best use of techniques i.e. requirements prioritization and incremental releases [18, 28].

Traditional Requirements Engineering in Agile (Overall Comparison)

All generic requirements engineering activities i.e. elicitation, analysis, and validation are used in Agile but in very different context. Overall comparison is made here on basis of a literature survey.

Customers' Availability and Involvements: The basic and most important factor, which is primarily responsible for performing requirements engineering in agile is customers' availability or involvement. This is concerned with constant feedbacks and up-to-date discussions. Moreover, both parties i.e. customers and development teams may attain a more better view of requirements, and this step of customer involvement is considered to be the most primary goal for RE. General example can be taken by keeping this concept in mind that a single person i.e. customer can answer all issues, questions, and problems being faced by developers so precisely and accurately that they can provide him with best results. In addition to this, it would be beneficial for them to take right decisions and act smartly.

Same customer involvement is also done in traditional requirement engineering phase i.e. elicitation phase, which allows you to gather data from interviews, questionnaires, brainstorming, focus groups, observations, use cases, and much more.

The major difference, which distinguishes agile from traditional RE is that customer involvement in traditional RE is only done during elicitation phase but in agile development, it is included in whole software development phase. This is what professionals lack in actual software development phase in traditional RE.

Constant Interactions/Interviews: Agile mainly focuses on constant interactions between professionals and customers. This involves solving necessary problems, having maximum communications, answering questions, asking questions, and many other relevant factors. All misunderstandings, misinterpretations, and misleading information are eradicated by this constant interaction in Agile. Alternatively, constant interaction is said to be a form of interviews One more advantage of having agile interviews is that many customers come to know skills of development teams by interacting with them through interviews so that they may check their development skills for better performance in software projects.

Requirements Prioritization: Similar to traditional RE, agile also applies prioritization among its methodologies. The main goal of developers or development teams is to perfectly deliver necessary software to clients having high-priority features. Change in requirements take places during development phases. Therefore, all of the requirements are being managed and sorted as per their priorities and are updated regularly in each phase. This helps clients to have a more clear view of all system requirements including their impacts on system. Lean software development makes best use of this technique more effectively.

JAD Sessions: Both ASD and DSDM make bets use JAD sessions in various ways. Customer involvement increases a lot by these JAD sessions. In ASD, these sessions enable customers and developers to sit on a round table and discuss necessary features of a product. These sessions provide useful results provided that head manager r top level manager leading these sessions act smartly. The most amazing fact about these sessions is that in ASD these short sessions never force all customers to have face-to-face session physically or they do not necessarily need to be On-Site like XP. These sessions are held in almost each phase of software development for best results.

In DSDM, these sessions allow professionals and customers to better understand whole system in starting phases i.e., if system is new for them. This increases promotion, communication, discussion, cooperation, collaboration, speculation, understanding, and through team work.

Requirements Modeling: Do not get confused with modelling you came across in Agile Modeling technique. It is surely true that modelling is only preferred and performed in Agile Modling methodology i.e.AM but its use with respect to requirements engineering varies significantly. In AM, modelling is concerned with better understanding of a very small and minor part of system, which is to be developed. Models in AM are drawn on white paper and are useless after project is done or their purpose is met. Many of them never come under domains of proper and complete documentation procedures.

But when we talk about requirements engineering perspectives, modelling provides us with some different meanings. Here in RE, variety of models comes under through discussion. These models in RE provide a quick and brief overview of a complete system to be developed by making use of all other models. Moreover, all of these models in RE become permanent attributes of complete documentations and all of them need to remain properly updated and effectively maintained.

Modelling concept of RE is identical to that of FDD i.e. Feature Driven Development. The finally developed model defines whole of the system along with its detailed features and all of the development bases on this developed model. We know that deigning and implementation phases in FDD manipulate iteration, therefore, requirements changing can be done later on. One should always keep in mind that this modelling approach never guides you about final system but allows you to initiate development more effectively and accurately.

Requirements Documentation: Less or no documentation in agile has been the problem for developers always. It' a great challenge being faced by many professionals for software development process. Though documentation cerates long-term problems yet it's beneficial to speed up development process. Merits and demerits always lie in between. For example, new comer comes in organization and needs to know about software or its attributes, having no documentation may let him suffer too much. Moreover, if he asks any other team member, it would be too complex to train him about software development fundamentals. Therefore, one more purpose of documentation is knowledge sharing.

One more factor, which is concerned with documentation is change of requirements. This means that developers need to make changes or customers do that regularly are not documented. This may result in serious future issues, which may arise if the same team member or developer leaves the organization tomorrow or comes across any mishap unexpectedly, new person joining the team would never know how system was changed or which system implementations are made and so on. Mostly in agile environments, customers request team members to effectively document design attributes when system is migrated or there is any chance of expecting any change in the system.

To talk more about agile methodologies, they are more successful than traditional ones because they take very less time, they offer very les documentation, and they provide rapid development results.

Traditional RE in the other hand focuses too much on documentation, which answers almost each and every question of the customer. In simple words, agile methods lack documentation and traditional methods are over documented.

Requirements Validation: Validating requirements is a major aspect with respect to requirements engineering perspective including both traditional and agile developments. In agile development methods, software validation is done by professional and review meetings and technically proven acceptance tests. This gives a significant rise to customer's motivations towards software development process and development team also because regular demonstrations are given to customers by team leaders or developers. Project' schedules, management scenarios, costs, effort-estimations, cost-estimations, risk management issues, quality issues, and everything relevant come under these domains. All agile approaches have their own ways to validate requirements.

Many of them make best use of various meetings. These meetings are basically review meetings in which customers collaborate with developers in many ways. They come to know in detail that what does system do, what is current status, how system behaves, and much more. If any question arises in their minds, they feel free to ask to developers, who are present in all review meetings actively. Moreover, any suggestions, functionalities' discussions, change in requirements, feedbacks, strengths, weaknesses, advantages/pros, disadvantages/cons, limitations, and many other things are widely discussed between customers and developments teams. In addition to this, customers validate that all requirements of the system are well understood and no ambiguities lie in between them. If system works as it was expected customers finally run acceptance tests to check it. It it's not working according to expected results, they demand clarifications from development teams. Later on, after successful review meetings, software product reaches final production phase, which eventually leads to a deployment phase of first version. This very first deployment phase provides development teams with awesome returns on their investments.

XP is a special case for developers always because it is mainly concerned with face to face interaction of customer with developers. This creates great rise in factors i.e. trust, responsibility, and performance. Moreover, chances of asking and answering questions become quite easy in this domain.

We all know that agile methodologies are mostly dependent on small releases, therefore, validating process of requirements is no doubt primary and most important. This is because a constant feedback is at least received against each release form the customer's side. Agile mainly follows the concept of evolutionary prototyping as far as requirements validation is concerned because it also needs in-time delivery of product with customer feedback. A small difference lies in testing phase of agile because agile focuses more on testing than evolutionary prototyping[31].

Requirements Management: Requirement management effectively needs requirement changes, tracking, traceability, and proper documentation. Tradition development approaches strictly follow these requirement management issues in a lot of detail but Agile methodologies lack in some of these issues. It does not mean that agile has no mechanism for saving or managing requirements but lack of documentation or very les documentation creates lots of limitations for agile methods.

If there is a signed contract between two software companies, documentation means a lot. This is because one can change his views any time and unexpected moments never inform before coming. This is the major reason why agile methodology relies more on cost, time, and expenses rather than wasting its time on other useless activities. Moreover, agile methodologies widely emphasize at fixed price contracts. Instead of documentation, the major objective of Agile is to build a strong relationship and trust among its customers. Delivering all projects rapidly and on time and focusing only on software development phases instead of jotting requirements down gives a strong positive impression to clients.

Many agile approaches perform requirements management in various formats.

Index cards or user stories are comprised of all major requirements with necessary details. Maybe level of detail is low, but all of these requirements are managed by the use of cards or product backlogs. Though it is done at minor level, which means that agile eradicates many if necessary details but it uses all of those details during software development process. This slow management process is said to be lazy requirements elicitation process.

DSDM also allows requirements management by constant track of all changes. When business study has necessary changes to apply or repeat, record is separately kept in some other documented.

Scrum makes best use of product backlog for effective requirements management. The best feature of product backlog of scrum is that it ever deleted old requirements but refers to all new requirements by explaining or mentioning that why those requirements were changed, modified, or deleted.

Observations and Brainstorming: Both of these techniques are not properly defined by any of agile approaches but they do exist in them within certain extent.

Observation surely comes under requirements elicitation phase, which gathers users' requirements by looking at the tasks of others or by observing others. The same thing happens in agile when some of the users or developers forget to mention any requirement or any details regarding work. Later on, observing system's behaviour, new requirements can be suggested by developers or proposed by the customers for effective results. That's how observation approach is applied.
Brainstorming on the other hand is bringing or introducing creative ideas for better development. Agile approaches widely make use of this approach indirectly. For example, face to face communications, review meetings, change in requirements, feedbacks, and many other procedures involve this approach because creative and new/fresh ideas are always welcome by experts in any development scenario.

Non-Functional Requirements: Non functional requirements give rise to serious issues in agile development. This is because customers want best and fast results and they always ignore performance, workloads, efficiency, and safety issues. Requirements are elicited in agile approaches without keeping an eye on portability, maintainability, and many other non-functional issues. Mostly non-functional requirements or NFRs are closely related and equally important to that of functional requirements and are almost not expressible in many cases. For example, customer in agile development environment wants you to develop a gaming system, which must be a third person shooting game. Though all requirements will be provided by him like gaming modes, strategies, rules, processors, and graphics etc yet he will always ignore minor requirements that may be any ethical or organizational or any requirements like it should not be a copy of any idea of other game, which standard should it follow for gaming, its usability, flexibility, serviceability, quality, and many other factors. Therefore, NFRs in both traditional developments as well as agile developments must never be ignored in any case. We do not mean to consider all of NFRs but at least all those should be included, which affect system's performance in any way especially when final product is released.

Waste of Requirements (Lean): Requirements in lean development are always saved from wastage. This refers to any ambiguity, which takes place gathering of requirements. Waste is eradicated successfully during lean manufacturing.

Proper identification of waste becomes necessary in Agile modelling. This avoids further creation of waste in later stages of development. It is obvious that if you develop a system for which you gather many requirements, which are more than needed. These would be totally useless requirements. You have to get rid of them. Lean made best use of these by providing reduction and prioritization mechanism to professionals.

These all are requirements gathering and minimizing requirements procedures. Triage process can be related to them in some scenarios but details would vary surely. Moreover, useless requirements include writing more code with higher costs, increased complexity of code, delayed deliveries, using more resources than needed, and many other factors are added.

Findings (On Basis of Literature Survey)

On basis of literature, we found many techniques for problems resolving, which allow professionals to manage requirements efficiently.

Sources We Have Used: We have compared both Traditional RE approaches and Agile RE approaches. Following table illustrates this.

Table-2
Sources We Used

Source	Count
IEEE	5
Google Scholar	3

Agile RE Limitations and Improvements: i. Most of customer interaction is done in Agile by prototyping. It is not possible to gather all requirements from just one person. Therefore, elicitation must occur from various stakeholders. ii. Open questions, meta-questions, and JAD sessions can help removing a lot of conflicts. iii. Agile relies on validation or testing but not on quality assurance. iv. N verification is there in Agile but this would be possible if both validation and verification occur at the same time. v. Agile does not address requirement engineering in distributed agile development. This can be done effectively by making use of certain tool or technique. vi. Agile can make use of UML, and negotiation tools for better development scenarios. vii. Along with configuration management, Agile should also rely on requirements management practices for better tracking of customer needs/requirements. viii. Project plans must be constant in all releases

Proposed Idea (On Basis of Issues/Gaps)

Agile has never discussed requirements engineering in distributed environments. Distribution of software saves both time and cost. Moreover, distributed development relies on software development in various geographical surroundings. Cultural differences, large distances, and various time zones lead t weak communications, which adversely affects the projects. Many issues arise in it like communication, documentation, training, work distributions, and much more. Improvements in communication, proper visits, team divisions, team coaching, improving documentations, and effective use of best tools can help removing these issues.

Major Differences

Table-3
Differences between Agile RE and Traditional RE[33]

Agile RE	Traditional RE
Suitable for projects where development is done along with requirements gathering for quick releases and updated customer feedbacks	Traditional RE does not address whole development life cycle and suits to those projects, which are enhanced versions of some applications.
In Agile development, organizations do not know about their initial needs and all needs and requirements evolve during development depending on customers' feedbacks	Traditional RE defines all goals scopes, definitions, effort estimations, cost estimations, team sizes, and everything related to development before proceeding.
Change in requirements is applicable in Agile	Requirements once defined can never be changed later on
Number of stakeholders are limited in agile along with project sizes	Traditional RE involves many stakeholders and a big elicitation process
Procedures for waste of requirements are discussed and applied in Agile	Traditional RE doe not concern with any wastage. Yes. It allows triage a bit similar technique to lean

Social Networking Tools: Social networking tools, which allow video conferences, chats, and many other reliable tools can help this way for distributed developments. i. Bug-fixing tools can be effective. ii. Knowledge centres and collaborative environments are also recommended[29].

Hire Skilled People: You must hire skilled people by yourself. Do regular visits and check status of developments off and on. Moreover, you may not all skilled people in all regions therefore, be careful in all aspects.

Documentation: Agile offers no or very less documentation. This needs to be improved because without documentation or sufficient manuals or information, new trainers or professionals cannot provide better outputs.

Meetings: Agile though offers regular meetings within each release but in distributed environments, geographical meetings must be adopted by making use of many tools like Skype or any conferencing tool.

Final Result: It is impossible for Agile to carry out requirement engineering in these environments because Agile needs customers to interact physically face to face, which never is possible in these environments. Moreover, agile needs less documentation. It is never possible that you hire bets professionals from various corners of the world or specific areas. Therefore, applying Agile RE in distributed environments causes great RE issues.

Research Questions: How Agile practices can be applied to distributed development environments and which of them can be best?

Sub Questions: How RE under the chosen Agile practice would work?

Conclusion

All discussion over here revolves around traditional RE, Agile RE, and all Agile approaches. Moreover, RE in Agile and Agile in RE is also discussed along with all challenges and benefits. Finally, need of improvements, comparisons, findings, and application of Agile in distributed environments are provided to fill out gaps and provide necessary and useful ideas.

References

1. Frauke Paetsch: Requirements Engineering and Agile Software Development, *IEEE,* **(2003)**

2. B. Boehm, "Requirements That Handle Ikiwisi, COTS, and Rapid Change," *Computer,* July 2000, 99–102 **(2000)**

3. Beck K., et al., *The Agile Manifesto.* 2001: p. http://www.agileAlliance.org

4. Cockburn A., Selecting a project's methodology, *IEEE Software,* 17(4) **(2000)**

5. Cohn M., Ford D., Introducing an Agile Process to an Organization, available at: http://www.mountaingoatsoftware.com/articles/Introducing AnAgileProcess.pdf) **(2002)**

6. Ramesh B., Cao L. and Baskerville R., (5 AUG 2010), Agile requirements engineering practices and challenges: an empirical study. Online Published at: 13 NOV 2007. Information Systems Journal, **20,** 449–480. doi: 10.1111/j.1365-2575.2007.00259.x) **(2010)**

7. Kent Beck Extreme Programming explained, Addison-Wesley, **(1999)**

8. Extreme Programming. What is Extreme Programming? [Online] Retrieved 18th March 2009. Available at: www.extremeprogramming.org) **(2009)**

9. M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global Software Engineering, 2008. ICGSE 2008, *IEEE International Conference on,* 222-226 **(2008)**

10. Agile Modeling Home Page. Effictive Practices for Modeling and Documentation. [Online] Retrieved 17th March 2009. Available at: www.agilemodeling.com **(2009)**

11. M. Cristal, D. Wildt and R. Prikladnicki, Usage of SCRUM Practices within a Global Company. Global Software Engineering, 2008. ICGSE 2008, *IEEE International Conference on,* 222-226 **(2008)**

12. Duncan R., "The Quality of Requirements in Extreme Programming", *The Journal of Defence Software Engineering, June* **(2001)**

13. Palmer S.R. and Felsing J.M., A Practical Guide to Feature-Driven Development. Upper Saddle River, NJ, Prentice-Hall **(2002)**

14. Highsmith J.A., Adaptive Software Developmet: A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing **(2000)**

15. Bayer S. and Highsmith J., RA Dical software development. American Programmer **7(6),** 35-42 **(1994)**

16. Yasuhiro Monden, *Toyota Production System, An Integrated Approach to Just-In-Time*, Third edition, Norcross, GA: *Engineering and Management Press,* 0-412-83930-X **(1998)**

17. NCMS study, Product Development Process –Methodology and Performance Measures Final Report, January 31, 2000 **(2000)**

18. Er. Kirtesh Jailia, Mrs.Sujata, Mrs.Manisha Jailia, Mrs.Manisha Agarwal, Lean Software Developmen, International Journal of Software Engineering and Its Applications **5(3), (2011)**

19. http://shapingsoftware.com/2009/06/15/introduction-to-lean-software-development/ **(2013)**

20. Alan M. Davis: *Software Requirements Revision Objects*, Functions, and States, Prentice Hall PTR, **(1994)**

21. Sommerville I., Sawyer P., Requirements Engineering: A good practice guide, John Wiley and Sons, 1997, ISBN: 0-47-97444-7 **(1997)**

22. Romi Satria Wahono, ANALYZING REQUIREMENTS ENGINEERING PROBLEMS, Proceedings of the IECI Japan Workshop 2003 Chofu Bunka Kaikan Tazukuri, Japan

23. [Christel-91] Michael G. Christel and Kyo C. Kang, Issues in Requirements Elicitation, Technical Report CMU/SEI-92-TR-12, ESC-TR-92-012, September 1992 **(1992)**

24. [Rumbaugh-99] James Rumbaugh, Ivar Jacobson, and Grady Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, **(1999)**

25. A. Strauss and J. Corbin, *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, Sage Publications, (**1990)**

26. Pekka Abrahamsson, Outi Salo, Jussi Rankainen and Juhani Warsta: Agile software development methods - Review and analysis, VTT Electronics, **(2002)**

27. Cockburn A., Agile Software Development, Addison-Wesley **(2002)**

28. Poppendieck T., Poppendieck M., Lean Software Development: An Agile Toolkit for Software Development Managers, Addison-Wesley **(2003)**

29. M. Fowler, Using an Agile Software Process with Offshore development, http://martinfowler.com/articles/agileOffshore.html, July 2006 (on March 10, 2010)

30. Pekka Abrahamsson, Outi Salo, Jussi Rankainen and Juhani Warsta : Agile software development methods - Review and analysis, VTT Electronics, **(2002)**

31. Poppendieck T., Poppendieck M., Lean Software Development: An Agile Toolkit for Software Development Managers, Addison-Wesley **(2003)**

32. http://en.wikipedia.org/wiki/Lean_manufacturing#Types_of_waste

33. M. Fowler, Using an Agile Software Process with Offshore development, http://martinfowler.com/articles/agileOffshore.html, July 2006 (Retrieved on March 10, 2010)