International Journal of Advanced Science and Technology
Vol.58, (2013), pp.75-86
http://dx.doi.org/10.14257/ijast.2013.58.07

An Appraisal of Agile Software Development Process

Muhammad Amir[1], Khalid Khan[2], Adnan Khan[3] and M.N.A. Khan[4]

[1,2,3,4] *Department of Computing, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology*
[1]*amir_amirkhan94@yahoo.com*, [2]*khalid.sarhadi@gmail.com*,
[3]*adnan_online47@yahoo.com*, [4]*mnak2010@gmail.com*

Abstract

Software development, as a whole, is a complex process and on top of it, the requirements keep changing during the development phase. Software configuration management happens to be the most critical part as it necessitates doing considerable modification in the software design and code. Agile software development process provides a solution to such a changing environment. Agile methods use an incremental approach to develop high quality software within time, cost and other associated constraints through several iterations. In this paper we provide a critical assessment of the agile software development process in a systematic manner. This study is based on the survey of previous research reported in the contemporary literature and the practices being followed in this area.

Keywords: *Agile Software Development, Global Software Development, Extreme Programming, TSDM, Requirement Engineering, Component Based Software Engineering*

1. Introduction

The software development process provides specific guidelines to the developers. Agile software development (ASD) is an innovative software development process. Agile methods provide ways to develop quality software quickly and allow accommodating change requests at any stage of the software development process. Structured programming started in 1954 and the object oriented approach was introduced in 1960 [1]. Later on, software process models were introduced in 1970 [1]. Eventually, software engineers, programmers and practitioners felt the need for proper guidelines for software development. The first process model was the waterfall process model which was adopted from other engineering disciplines. After the waterfall process model, several other traditional development models were introduced. Different software development approaches were introduced because of the practice of varied software development cultures in the software industry. One of the distinctive characteristics of agile methods is that they do not rely on meticulous documentation. Traditional software development methods such as waterfall and spiral ordinarily involve heavy documentation and large design of the system; and because of this they are called heavy weight development methods. Because of the structured nature, stability and high assurance, the traditional software methods are still exercised in the industry [2]. During the last two decades, various agile methods have been introduced and adopted by the software industry and practitioners due to the specific limitations in the traditional approaches [1]. However, some practitioners prefer using a mixture of agile methods.

Despite all this, still the knowledge about agile software development process and its connection with software engineering remains obscure. That is why, project managers face difficulty in implementing agile methods within their organizations. In this paper, we make an

ISSN: 2005-4238 IJAST

attempt to describe the overall agile software development process in a methodical manner. Agile methods were developed and introduced in 1990's. These methods have been widely used in the business sector where the requirements are unstable [2]. Agile methods have short but fast development life cycle and focus on iterative and incremental development, customer collaboration and delivery of lightweight working code; that is why, these methods are generally known as lightweight development methods [2].

Agile manifesto was developed in 2001, and over the period of time, many agile activist got involved into this filed which ultimately led to the formation of an agile alliance [3]. Agile alliance is a non-profit group that carries out search for new agile approaches as well as sponsors the annual agile conferences. Many leading software companies such as AOL, CNBC, Yahoo, Google, Microsoft, Siemens, *etc*., adopted agile methods in their projects [3].The main purpose of agile process is to support the changes occurred during the development and quick production of working code and other related artifacts that provide value to the customer and ultimately to the project.

Agile software development methods increase communication and interaction between all stakeholders and focus on the development of quality software through various iterations, but still there is a gap between the best practices and the actual market environment [4].

Agile in itself is not a method, it is a philosophy based on agile manifesto. The manifesto focuses on the following four values and 12 principles [5]:

- ***Individuals and interactions*** over processes and tools.
- ***Working software*** over comprehensive documentation.
- ***Customer collaboration*** over contract negotiation.
- ***Responding to change*** through following a plan.

There is an associated value for each item shown on the right, and the items on the left are valued more.

The twelve principles behind the agile manifesto include:

- The highest priority is assigned to satisfy the customer through early and continuous delivery of software releases.
- Welcome the changing in the requirements, even late in development phase. Agile processes harness change for the customer's competitive advantage.
- Deliver functioning software frequently, i.e., in the time span ranging from couple of weeks to couple of months, with a preference to the shorter time-scale.
- Business people and developers should work together on daily basis throughout the project.
- Build projects team comprising the motivated individuals. Provide them a conducive environment and necessary support they need besides entrust them enough authority and empowerment to get the job done.
- The most efficient and effective method of conveying information 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 user should be able to maintain a constant pace for an indefinite period.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity 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 behavior accordingly.

1.1. Requirement Engineering in ASD

In ASD projects, customer plays an important role and the overall progress and success of project depends on the customers. This results in a number of challenges for the project managers and the development team. Agile processes employ face to face communication with customer and confirm the customer change request through several iterations [6].

Customers or their representatives describe requirements and decide which requirement should be implemented in the next iteration. This is the responsibility of customer that the requirements must be unambiguous, correct and consistent in order to save development cost. Functional requirements can be easily added, modified and deleted for an existing system. But, the non-functional requirements like performance, reliability, *etc.*, cannot be easily added; therefore, non-functional requirements should be elicited and implemented properly at early stages of the software development life cycle. Medium and large size projects require high quality requirement engineering skills and technical knowledge. Since an ordinary customer does not possess these skills; therefore, the requirement engineer should play the role of liaison officer [6]. The requirement engineer is required to work with the customer to prevent the project from failure. In TSDM process, the role of requirement engineer is involved only at the early phases, while in agile projects, the requirement engineer should be available throughout the development process. Requirement engineer should work as a facilitator mediating communication between the customer and the development team [6]. The main activities generally carried out by the agile requirement engineer include [6]:

- He/she should be able to develop the cost estimation and schedule for each and every iteration.
- He/she should support the customer in order to identify stakeholders and representatives.
- He/she should help and guide the customer to elicit the right requirements at right time.
- He/she support customer in expressing requirements in consistent manner.
- He/she also support customer in identifying test cases and also in testing the increments.

In short, identification of customers and communication with them is an important activity for the elicitation of valid requirements. Since a requirement engineer himself or herself works as a liaison officer, it helps improve communication between customer and developer.

1.2. Software Architecture and ASD

Architecture is the basic building block or blue print of a system. Architecture is the action or process of building a system and agile is the ability to think quickly [7]. However, these two words do not fit together to coin the term *Agile Architecture*. Further, there is a difference between software design and architecture. Every system must have architecture but it is not necessary that architecture models describe that architecture [7]. Every project must have an architecture, but in agile paradigm, there is a glaring difference between architecture and agile software development.

A study conducted by Philippe Kruchten describes these conflicts and provides the following interconnected solutions [8].

Semantics: Architecture and design are not the same thing and every design is not an architecture. Agile team must understand and agree with the definition of architecture. This can provide a good starting point.

Scope: Scope defines the boundaries of every project. Agile team must have the knowledge of the scope of the project, and on the basis of this, they should have knowledge about the effort needed for a robust architecture.

Lifecycle: Architecture is basically a set of design decisions. These decisions will be hardest to change. Therefore, software architecture should be carried out at the early stages of the software development life cycle.

Role: Who will be the architect? It is the most critical issue. The identification of the right people and assigning them right roles and responsibilities is a key point to the success of every project.

Documentation: How much description is needed for the architecture? Architecture document is required for communication between development team and other stakeholders, therefore sufficient documentation is needed.

Methods: Decide on what methods can be used to resolve architectural issues. The given method is useful. Concentrate on the above points.

Value and Cost: All agile methods provide early business value. The critical task is the cost that is not visible. Incremental Funding Method can be used to resolve this issue. The above technique will provide help for software architects to develop the architecture of agile projects.

This paper is organized into seven sections. This section being the introduction about agile software development is followed by literature review in the second section. Third section discusses agile methods, along with their related issues, that are widely used in the industry. The benefits of agile methodology are described in the fourth section followed by a brief account of the issues and challenges related to the overall agile process in the fifth section. The prospective research directions are outlined in the sixth section and finally, we conclude in the last section.

2. Literature Review

Malik *et al.*, [1] discuss three agile methods extreme programming, scrum and agile modeling as well as describe differences among them. The authors also provide some suggestions regarding when and where these methods should be followed by the practitioners.

Juyun [2] highlights differences between agile and traditional software development methods and describes scrum framework - a well known agile method. The author also discovers issues and challenges in scrum through an in-depth case study. The identified issues include documentation, user involvement, working environment and scrum ceremonies. The issues and challenges discovered by authors on the basis of one case study and interview of only nine employees is not sufficient to establish an assumption about a method.

Hassan *et al.*, [3] present a case study on scrum in a government organization and discover issues and challenges related to the adoption of this method. They further compare these issues and challenges with previous research carried out in this area. The issues and challenges in scrum are: missing the agile master role, the overzealous teams, the absent of a pilot project, scrum implementation, current work pressure, upper management concerns and

governmental bureaucratic hurdles. Some of these issues are also identified by other researchers; however, the new challenges which were not discovered before are current work pressure, governmental bureaucratic system and documentation.

Frank *et al.*, [4] give an overview of agile methods along with their strengths and limitations in different domains. The authors discuss the nitty-gritty that is needed for agile methods to cross the chasm. They author further discuss strengths, weakness and issues in agile methods and introduce a diffusion model to overcome these issues. However, all these assumption are based on only interviewing few participants that is not a sufficient to make a conclusion.

Trenton [5] introduces an approach for using extreme programming (XP) in large-scale projects. The author also proposes a hybrid approach that combines both extreme programming and waterfall models to overcome the issue of adoption of agile method in large-scale project. But still there are some issues related to this method like scaling and better documentation that demand further work. Despite the solution proposed by the author raises an important question as how both the models could be integrated since they have different framework.

Elke [6] discuss the role of a requirement engineer in agile software development. In agile software development, the customer or customer representative is responsible for requirements description. Sometime, customers or their representative do not have the necessary skills to describe their requirements. To overcome this issue, the requirement engineer should play a role as liaison officer mediating the customer requirements to the developers. Author also highlights some benefits that can be achieved if requirement engineer works shoulder by shoulder with the customer. This approach is helpful to improve customer-developer communication during the agile software development process. For the sake of exploiting true benefits of such an approach, the requirement engineer should be an expert in other areas of software engineering i.e., cost estimation, requirements verification and validation.

Christine [7] describes the experience of doing agile architecture in an agile and a non-agile way, and argues that the agile projects should have architecture and the agile architecture must follow the agile principles.

Philippe [8] discusses disagreement between agile methods and architecture and propose a model called zipper model to overcome such conflicts.

Bhalerao *et al.*, [9] discuss agile software development process in detail. The authors describe that agile methods do not follow traditional software development life cycle which is the core reason that many practitioners do not adopt these methods. Authors stress that there is a strong need for a generalized agile software development life cycle. ASDLC provide guidelines for managers to understand and implement agile methods. ASDLC represent all phases of SDLC in iterative manner. ASDLC phases improve both internal and external quality of software product but this technique requires strong validation.

Dan *et al.*, [13] identifies limitations of agile methods (*i.e.*, extreme programming, scrum, agile modeling and unified process). The limitations include inadequate support for distributed development environment, subcontracting, building reusable artifacts, development involving large teams, developing safety critical software and developing large software.

Jorg *et al.*, [14] state that if conflict occurs in agile projects, then such failures can lead towards the worst-case scenarios. The authors describe that the choice of method and tools is an expression of the value systems of an organization. They further discuss this value conflict. They do not provide any suggestion and guideline for such conflict and the paper is not totally focus on the agile methods.

3. Insight into Agile Methods

Various agile methods have been introduced in the last two decades. Many IT industries are using these methods and producing quality software products. Agile manifesto provides solution to changing requirements; therefore, agility is becoming more popular than the traditional software development methods. Every agile method has its own life cycle and all agile methods are based on agile manifesto [9].The main philosophy of agile methods is to deliver early working code of software in short iterations, update the next version according to the feedback of customer, satisfy customer requirement and develop the whole system through several increments [1]. Agile methods include Extreme Programming (XP), Scrum, Agile Modeling, Test Driven Development, *etc.*, [2].

3.1. Extreme Programming

Extreme programming is the most popular agile method widely used in the industry. This methodology suggests that customer must be the part of the team. It does not entirely eliminate the requirements, design and testing phase, rather it suggest combining all these phases and performing work in iterations [5]. Each iteration includes little amount of planning, a small bit of analysis and design followed by minimum testing. XP is a more development oriented method than the management oriented method. XP produce the final product through several iterations. At the end of each iteration, user acceptance test is performed and on the feedback of customer, the agile development team decides whether to terminate the development process and release the software product or continue with any change in the current system [10]. User describes their requirements in stories. On the basis of these stories, development team develops the software development plan. The team also performs estimation on these stories and breaks up the development tasks into a series of iterations [10]. There are four following main concepts of XP:

Continuous Integration: The development team should integrate changes to the system at least once a day.

Project Velocity: Project velocity means progress on the project. Project velocity basically provides measure to check how much work is done.

Pair Programming: Pair programming is the most important aspect of XP. Pair programming is a technique which is used to improve quality of the product. XP claims that if two programmers work together on a single system, then they can understand well the user requirements from user stories and this strategy can help produce high quality software product.

User Story: User story basically defines the actual problem. Users are responsible to write these stories. These stories are approximately three sentences long and are written in simple English. Since these stories are short, therefore, there is a need of customer representative who can work with the development team to further describe and explain these stories.

Paul *et al.*, [11] explain that the problematic feature of XP is the amount of on-site customer involvement that it requires. The on-site customer involvement has a lot of benefits, but research in this area is still immature to measure these benefits.

Trenton [5] reported some advantages and disadvantages of large-scale projects using XP (an agile method). Software development is a complex task and developers' aim is the successful completion of this task within time, cost and other related constraints. However, agile methods provide solution for such complex tasks for small and medium-scale projects.

However, there are still some issues involved to incorporate agile methods in large-scale project. Distributed projects are one of the types of large-scale projects which comprises different teams located at different places. In such a global and distributed environment, the complexity of project grows and requires effective coordination. The main difficulties in such environments are spatial, temporal and cultural [5].

Spatial Issues: One of the most important parts of XP is that it requires open communication between development team and customers. The task is easy when the teams are working in the same building but become more complex when teams are located at different cities and countries.

Temporal Issues: When teams are located at different locations having varied time zones, the working hours and routines becomes different due to which sufficient and effective communication among the teams is generally not possible.

Cultural Issues: The cultural issues do not directly relate to the distributed environment, but due to the lack of face-to-face communication, there may be a chance of misunderstanding.

3.2. Scrum

Scrum is an agile method which had been practiced before the announcement of agile manifesto [1]. Scrum can be used to manage and control complex software development using iterative process [2]. It was developed by Schwaber and Sutherland in 1990's. Schwaber define three main characteristics of scrum: transparency, inspection and adaption. These three characteristics are defined as below. Software code should be visible to everyone involved in the process (Transparency). The code can be reviewed by the experienced developers (Inspection). If there is need to improve the code, the reviewers suggestion should be adopted (Adoption). The scrum framework consists of three components: Roles, Ceremonies and Artifacts.

Roles: Basically there are three kinds of roles in the scrum process: Product Owner, Team and Scrum Master. Product owner is responsible for developing overall project's requirements, return on investment objective, project release plan and initiating funding for project. The responsibility of the team is the implementation of required functionality. Scrum Master represents management and team. Scrum master is responsible for enforcement of scrum values and practice.

Ceremonies: Ceremonies includes in the scrum process are: Daily Scrum Meeting, Daily Scrum of Scrum Meeting, Sprint Review Meeting, Sprint Planning Meeting. The daily scrum meeting is a short meeting approximately 15 minutes long. The discussion is carried out on what part of the project has been accomplished since the last meeting and what must be done before the next meeting. Daily scrum of scrum meeting is also a short meeting and purpose of this meeting is to synchronize the work between different scrum teams. Sprint planning meeting is a monthly meeting in which team and product owner discuss what needs to be done in the next sprint. Scrum review meeting is a four hour monthly meeting in which scrum team presents what had been developed during the scrum.

Artifacts: In scrum process there are three types of artifacts: Product Backlog, Sprint Backlog and Burn-down chart. The product backlog contains the functional and non-functional requirements in prioritized form. The priority is assigned according to the importance of the requirements to the business. In sprint backlog, the team selects the first high priority requirement from the product backlog and breaks down it into a set of smaller

tasks. The burn-down chart provides a graphical representation of work and time. The burn-down chart is accessible to all the members involved in the project. Malik *et al.*, [3] identify issues and challenges in scrum as:

- Scrum Master plays an important role in the project, but that position may not be filled due to financial constraints.
- Absence of pilot project is also a critical issue.
- In the absence of scrum master, the implementation of scrum is not an easy task.
- Work pressure is also an important issue.
- Convincing upper management for investing in a new method is also a critical task.
- Minimum documentation is also greatest big challenge.

3.3. Agile Modeling

Agile modeling is a collection of values, principles and practices for modeling software systems [1]. Agile modeling was introduced by Scott Ambler in 2002. This technique can be adopted and used with other existing agile methodologies aiming to develop software in a quick and effective manner. The values of agile modeling include: communication, simplicity, feedback, courage and humility. Agile Modeling principles are similar to XP. However, some additional principles are included for: modeling purpose, multiple effective models and to focus on the quality. Agile modeling includes practices to create successful model for the system. Agile modeling practices highlight active stakeholder participation, focus on group work, verify the model for correctness, implement the model and show the result to user through interface as well as creating several models in parallel.

Agile modeling is basically a modeling method which provides modeling facilities to designers and other agile practitioners. Like other agile methods the main issue in agile modeling is communication with the stakeholders and understanding their requirements.

3.4. Test-Driven Development

Test driven development is an agile software development method in which the automated tests are written before the functional code is developed in iterative manner. TDD is not a complete development methodology; rather it is used with the integration of other agile methods such as XP. It is used to write automated tests for individual units of system [12]. An individual unit may be a smallest unit known as component. In traditional development environment, unit testing is performed after the code is written by the programmer and the unit testing is carried out by the owner of the code. While in TDD environment, the programmer first writes the automated test for the component prior the coding.

TDD is also called test-first programming, test-driven design and test-first design. Agile methods got popularity during the last decade because of its quick and iterative development process with small documentation and design process. The most popular agile method is XP which is used with scrum and TDD is an integral part of XP.

4. Benefits

This study finds the following benefits in agile software development methodology:

- The first and foremost benefit of agile methodology is that it reduces the production time because of shorter development life cycle and early delivery of incremental versions.

- The product progress is visible, so risk management and other managerial tasks can be easily handled.
- Flexibility to accommodate change at any stage of the development.
- Due to lightweight development process, it eliminates additional costs and time.

The software is developed under the time, cost and other associated constraints.

5. Issues and Challenges

Dan *et al.*, [13] present the following limitations of agile process by performing analysis on the assumptions underlying XP, Scrum, Agile unified process and agile modeling:

- Limited support for distributed developments.
- Limited support for subcontractors as contract is not in a well-defined form in agile process.
- Limited support for building reusable artifacts, because these processes focus on building software products that solve specific problem.
- Limited support for involving large teams during development. With a large team, the number of communication lines increase which decrease the effectiveness of such process. It does not mean that agile methods are not applicable in these domains rather the degree of agility will be affected.
- Agile process provides limited support in the development of safety critical systems and complex software. Large and complex software have a critical and complex architecture which is difficult to change and also, the cost of change in such systems is high. In complex systems, the functionality is tightly coupled and it is not possible to develop software for such systems incrementally.

Elke *et al.*, [15] reported challenges in agile software development through some myths and explained why agile methods were not completely adopted in the industry.

- Different projects have different size, application domain, constraints and risk. Agile methods are beneficial only for interactive applications; therefore, they cannot be adopted in the technical systems.
- Transition from traditional environment to agile is a critical task and requires proper planning.
- Since the agile methods concentrate on coding and not on documentation, therefore, it is difficult to validate a project.
- The quality of product cannot be measured due to the absence of sufficient quality model.
- Cost estimation and control is also a critical task.
- Requirements change is one of the main reasons of increased cost and maintenance in agile process.
- Agility minimizes the need for design. It is only beneficial for small systems and for those systems where the focus is on user interface.
- Agile practices try to minimize the need of documentation. But documentation plays an important role in reverse engineering and helps support staff.

From the above-mentioned issues and challenges, we do not mean to criticize the agile software development process. But, the intention of in this study is to discover some issues and challenges in this area to make this discipline more effective.

6. Future Work

The agile software development is a new discipline of research and requires further research to attain maturity. In this study, we have identified the following prospective areas of research in ASD.

- Need of standard SDLC for ASD.
- Quality is the most important aspect of every system, but it has not been amicably addressed in ASD.
- There is a strong need for defining standardization about the agile methods.
- How agile methods can be made workable for large-scale projects?

In addition, empirical research in agile software development has a very small presence in the literature, and thus it is need of the time to conduct empirical research in ASD.

7. Conclusion

In agile software development, the software system is developed through several increments. All agile methodologies welcome the change request at any stage of development. In spite of these methodologies there is still a strong need of a generalized SDLC for agile process. Agile approaches have a number of features and benefits, but there are several issues associated with it and this is the sole reason why this approach is not fully adopted in the industry. In this paper, we have focused on the overall agile process including its methods as well as the, issues and challenges associated with it.

References

[1] M. Hneif and S. H. Ow, "Review of Agile Methodologies in Software Development", International Journal of Research and Reviews in Applied Sciences, vol. 1, no. 1, **(2009)**.
[2] J. Cho, "Issues and Challenges of Agile Software Development with Scrum", Issues in Information Systems, vol. IX, no. 2, **(2008)**.
[3] H. Hajjdiab and A. S. Taleb, "Adopting Agile Software Development: Issues and Challenges", International Journal of Managing Value and Supply Chain, vol. 2, no. 3, **(2011)**.
[4] F. Maurer and G. Melnik. "Agile Methods: Crossing the Chasm", 29th International Conference on Software Engineering, IEEE, **(2010)**.
[5] T. Hafterson, "Incorporating Agile Methods into the Development of Large-Scale Systems", UMM CSci Senior Seminar Conference, Morris, **(2011)**.
[6] E. Hochmuller, "The Requirement Engineer as a Liaison Officer in Agile Software Development", in Proceeding of AREW, **(2011)**.
[7] C. Miyachi, "Agile Software Architecture", ACM SIGNOSOFT Software Engineering Notes.[On-line], vol. 36, no. 2, **(2011)** March, pp. 1-3, Available: www.doi.acm.org/10.1145/1943371.1943388 [November 2012] .
[8] P. Kruchten, "Software Architecture and Agile Software Development-A Clash of Two Cultures?", proceeding of 32nd ACM/IEEE International Conference on Software Engineering, **(2010)**.
[9] S. Bhalerao, D. Puntambekar and M. Ingle, "Generalizing Agile Software Development Life Cycle", International Journal on Computer Science and Engineering, vol. 1, no. 3, **(2009)**.
[10] Sareena Software Inc, "An Introduction to Agile Software Development", Internet: www.sareena.com, **(2007)** June-**(2012)** November.

[11] D. Janzen and H. Saiedain, "Test-Driven Development: Concepts, Taxonomy and Future Direction", IEEE Computer Society, **(2005)**.
[12] P. S. Grisham and D. E. Perry, "Customer Relationship and Extreme Programming", Proceeding HSSE'05, **(2005)**.
[13] D. Turk, R. France and B. Rumpe, "Limitation of Agile Software Processes", Agile Alliance, [On-line], Available:cf.agilealliance.org/articles/system/article/file/1096/file.pdf, **(2012)** November.
[14] J. Pechau and P. Becker-Pecahu, "Challenges: Agile Values Meet Different Value Systems", proceeding companion to 23rd ACM SIGNOPLAN Conference on Object-Oriented Programming System Language and Application, **(2008)**.
[15] E. Houchuller and R. T. Mittemeir, "Agile Process Myths", Proceeding APOS'08, **(2008)**.
[16] T. Dingsøyr, T. Dyba and P. Abrahamsson, "A Preliminary roadmap for Empirical Research on Agile Software Development", Agile'08, **(2008)**.

Authors

Muhammad Aamir completed MS degree in Software Engineer from Shaheed Zulfikar Ali Bhutto Institute of Sciences and Technology, Islamabad Pakistan. He received a BS degree in Software Engineering from City University of Sciences and Information Technology Peshawar, Pakistan. He has more than two years' experience in software quality assurance, system analysis, manual testing, automated testing and system modeling. Currently he is working as Sr. QA Analyst in a medium level software company in Islamabad, Pakistan. His research area of interest is Agile Software Development, Software Quality Assurance, Software Requirement Engineering, and Automated Testing.

Khalid Khan is an MS student at Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad Pakistan. He received a BS degree in Computer Science from Malakand University, Pakistan. His research areas of interest are in Software Quality and Agile software Development.

Adnan Khan is an MS student at Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad Pakistan. He received a BS degree in Computer Science from Hazara University Abbotabad, Pakistan. His research area of interest is Software Engineering.

M.N.A. Khan received his PhD degree in Computer System Engineering from University of Sussex, Brighton, UK. His research interests are in the areas of Artificial Intelligence, Computer Forensics. Cloud Computing and Software Engineering.