The Factors of Quality Assurance in Agile Environment

Ayesha Saad[1], Manish Madhav Tripathi[2], Preetam Suman[3]

[2, 3]*Dept. of CSE, Integral University, Lucknow, India*

***Abstract*- Agile is one of the most popular methodologies to produce software these days. They aim to produce software wherein specifications are constantly altering and seek to make development easy whilst ensuring quality. The several agile techniques just like XP, Scrum and many others use best ways that help to improve Quality. Thus ensuring Software package Quality Assurance (SQA) inside product delivered. In this particular paper Quality techniques of Agile may be focused upon. The analysis from the different techniques may be carried out in basis of crucial features, quality variables achieved, timing, in addition to cost.**

Keywords: **Agile, Agile methodologies, Software quality, Quality Assurance**

I. INTRODUCTION

While there is a constant modifying environment inside the software business it is usually impacting the software program development course of action which calls for the procedures to face expected alterations through it's lifetime never-ending cycle [1]. Agile systems hence are available as a rescue mainly because it offers up development systems which have been adaptive and also encourages fast and also versatile response to alterations in requirements. Abbrahamson describes about how exactly to recognize which a development methodology can be an agile one [6]:

- **Incremental:** Small software releases with rapid cycles.
- **Cooperative:** Customer and developer working constantly together with close communication.
- **Straightforward:** The method is easy to learn, modify and document.
- **Adaptive:** It is easy to make last minute changes.

Good quality Assurance in agile is a matter of concern as well as a deciding factor of delivering an item that is acceptable towards the customer. A traditional Good quality Assurance technique relies upon heavy weight assessment methods whereas Agile Good quality Assurance techniques are built-in daily activities by teams. The actual paper collects high quality assurance practices in agile together along with analyzes them. Software quality is the measure or stage to which a head unit, or process meets what's needed that are specified because of the customer and fulfils the particular expectations of buyer. Quality Assurance is a collection of planned actions accomplished in a systematic way to provide confidence the software development process confirms with all the requirements. The main objective of the research is to accumulate the various high quality assurance practices connected with agile and analyze these phones gain a deeper comprehension of the level along with state of high quality assurance in agile along with how this methodology aims to attain good quality software. This paper analyses quite features of every single agile practice of which helps in reaching.

This particular paper is structured in the following manner. First, a short description of agile good quality and agile good quality assurance is talked about. In the next section the many quality assurance methods followed in Agile is briefed upon. Next, the practices as well as the existing researches are usually analysed. The last along with final section wraps up the paper along with the future work.

II. AGILE QUALITY AND AGILE QUALITY ASSURANCE

Pressman [2] defines quality as “conformance to explicitly stated functional requirements, explicitly defined development standards, and implicit characteristics that are expected of all professionally developed software”. Sommerville [3] defines software quality as a management process concerned with ensuring that software has lesser defects and that it reaches the required benchmark of maintainability, reliability, portability and so on.

A. Agile Quality

Ambler [4] considers agile quality to be a result of practices such as effective collaborative work, iterative and incremental development as implemented through techniques.

B. Quality assurance (QA)

It is a way of preventing mistakes or defects in products and avoiding problems when delivering solutions or services to customers; which ISO 9000 defines as "part of quality management focused on providing confidence that quality requirements will be fulfilled".

C. Agile Quality Assurance

It is the development of software that can respond to change, as the customer requires it to change [5]. Thus providing tested, working, and user approved software at the end of each iteration. The various aspects of agile quality have shifted the focus from

heavy documentation that was the requirement for quality in traditional processes.

III. QUALITY ASSURANCE PRACTICES IN AGILE

A. On-Site Customer

On-site customer is usually an exercise where you or his representative should be present and intended for the development team whole-time. The client has to know his requirements through the system and the actual developers should question you concerning requirements after they are uncertain as to what the system ought to be doing. A real customer may be the one who will really operate the system only when it's in production [1]. Effective communication and feedbacks are necessary. An increased reliance upon less informative communication channels end in higher defect charges [6]. User guidance is through preparing games, user testimonies, story cards along with acceptance tests. This technique helps ensure that users have the ability to carry their work on their own satisfaction in a effective, efficient along with economical manner. Tests confirmed that On-site purchaser practice has substantive optimistic influence on quality of transmission and speed involving software production [7].

B. System Metaphor

The system Metaphor is a method of explaining the logical architecture of a system, it could be the means of communicating about the project in terminology that both designers and customers can understand, and which does not require pre-existing familiarity with the problem site [8, 9]. System metaphor helps the consumer to communicate while using developers using the shared vocabulary in terms recognized by both developers and also customer. System metaphor raises the interaction between buyers and developers, which is a key point in XP, an agile practice for that success of version. According to [1] through metaphor you'll be able to get an architecture that may be easy to connect and elaborate. Garzaniti R., Haungs J., Hendrickson [10] in their case study regarding payroll project identified activities that teams can use to develop metaphors with regards to systems, and processes for evaluating system metaphors. They provided an effective, structural model regarding system metaphors, based upon Peircean semiotics, giving a simple account of just how metaphors can promote a software process. They said that the team had a major benefit of a very loaded domain model developed by members of the team within the project's first version. It gave the members with the project an advantage in understanding an extremely complex domain.

C. Joint Application Development

Combined application development (JAD) group meetings are planned in addition to controlled sessions that assemble cross functional people to be able to bring out high-quality deliverables in much very less time of time (High smith, 2000). JAD sessions are an aid to produce many deliverables as well as requirements and prototypes. These sessions last typically for any day or less and might be repeated prior to the goals have also been achieved. JAD sessions really are a cost efficient and rapid technique to develop requirements (Carmel et 's., 1992). JAD sessions reduce the defects induced while collecting requirements and as well design defects. Joint application development (JAD) is a facilitated group technique which they can use in systems demands determination (SRD). JAD can be utilized with other ways to increase their effectiveness. Evan V. Duggan in addition to Cherian S. Thachenkary in their study integrated JAD in addition to nominal group process (NGT), a well accepted technique that was used to reduce the effects of adverse group dynamics on task-oriented objectives. They examined this specific integrated structure within a lab experiment to ascertain if it could ease the down sides that JAD features faced during SRD. The outcomes suggested that this integrated approach outperformed JAD in their test environment; it was as efficient as JAD alone and yes it appeared to raise the decrease from the need for wonderful facilitation skills throughout group decision-making.

D. Refactoring

Refactoring as defined by Fowler[11] is a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behaviour. There are significant advantages that refactoring provide[11]:

- Refactoring helps developers to program faster
- Refactoring improves the design of the software
- Refactoring makes software easier to understand
- Refactoring helps developers to find bugs

The first advantage aims towards productivity. The last three advantages of refactoring relates to software quality attributes. In their case study R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, G. Succi [12] concluded that refactoring had effects on quality of the code, particularly on software maintainability, and development productivity. Their case study provided evidence that refactoring increases development productivity and improves quality factors, reduces code complexity and coupling and increases cohesion. T. Mens, S. Demeyer, B.D. Bois, H. Stenten, P. van Gorp [13] focused on different types of refactorings. They point out that some refactorings eliminate code redundancy, some increase the level of abstraction, some improve the reusability of the product, and some have a negative effect on the performance.

E. Pair Programming

Pair programming is an exercise where the code is written by two individuals at a single system. Every person has his individual role to participate in. Where one head concentrates on current method and its implementation while creating the code the other person has a additional strategic work regarding examining that perhaps the current approach will work or not as well as finding other methods of the problem. Cockburn The., Williams L. [14] conducted an experiment to determine the efficiency of pair programming in comparison with programming by just one programmer it has been observed that 15% more hours on the program was spent as compared to with individual. Costs tend not to increase and ensuing code has with regards to 15% fewer disorders. It was also concluded that the increase throughout development costs with regards to 15% with pair programming

Practices	Important Features	Empirical Evidence
On-Site Customer	Real, full-time user to answer queries	Experiments were carried out which confirm that On-site customer practice has substantial optimistic influence on quality of communication and speed of software production [7].
System Metaphor	Means of communication between developer and user.	Garzaniti R., Haungs J., Hendrickson [10] carried out experiments and said that the team had the benefit of a very rich domain model developed by members of the team in the project's first iteration. It gave the members of the project an edge in understanding an extremely complex domain
Joint Application Development	JAD sessions are structured, facilitated workshops that bring together cross functional people in order to produce high-quality deliverables in a short period of time [28].	Studies have shown that JAD sessions are a cost effective and fast technique to develop requirements [20].
Refactoring	Modifying the source code without changing its external behaviour.	Case study by R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, G. Succi [12] concludes that refactoring increases development productivity and improves quality factors, reduces code complexity and coupling and increases cohesion.
Pair Programming	Code written by two people at a single system	Cockburn A., Williams L. [14] conducted an experiment to conclude that with pair programming costs do not increase and resulting code has about 15% fewer defects
Test Driven Development	Unit tests written before writing the source code and executing after its implementation	Maximilien and Williams[18] accessed Test driven development at IBM and found that the application of Test Driven Development reduced the defect rate by about 50% compared to a similar system that was built using an adhoc unit testing.
Automated Acceptance Testing	Automated tests defined by user to verify system's functionality.	Any program feature without automated tests simply doesn't exist [1].

was recovered from the reduction of disorders. Lui and Chan [15] deducted of 5% moment savings gained by way of pair programming. Müller [16] discovered that pair programming reduced some time spent on the standard assurance phase of the project nearly simply by half.

F. Test Driven Development

Most of these tests are published before writing the source code and working after its rendering, thus they are executed during the entire development of the program. Developer performs this sort of unit tests to construct their confidence in code when compared with acceptance testing performed because of the customer for his satisfaction inside the system. S. Yenduri, L. A. Perkins [17] performed an experience two groups associated with students, one developing computer software and testing it inside the conventional way after implementation and the other group through Test Driven Improvement. It was figured the total volume of faults in device, integration and acceptance testing was smaller in TDD when compared with the traditional strategy. The number associated with faults detected because of the Quality Assurance party in TDD was not even half of that from the traditional approach. TDD brings better results while quality and output are of issue, which can be due to the way test cases are designed as the computer software is developed. Maximilien and Williams[18] used Test driven improvement at IBM with a project of 71KLOC associated with non-test code and also found that use of Test Driven Improvement reduced the defect rate by about 50% in comparison to a similar system that's built using an adhoc unit examining.

G. Automated Acceptance Testing

Acceptance Jeff Canna[19] describes acceptance testing by comparing the coverage it provides to the one provided by unit tests as "Perfectly written unit tests may give you all the code coverage you need, but they don't give you (necessarily) all the system coverage you need. The functional tests will expose problems that your unit tests are missing". Acceptance tests are written by the customers to verify that the system's functionality is in accordance to the requirements and expectations from the system. Automated testing is preferred is the process of executing automated acceptance tests rather than executing them manually. Any program feature without automated tests simply doesn't exist [1]. Automated acceptance tests must be ready by the middle of iterations and should run daily.

IV. ASSESSMENT OF QUALITY ASSURANCE TECHNIQUES IN AGILE

Given in Table at the top of the page

V .CONCLUSION

Agile follows the most beneficial practices to achieve quality inside the software. It is designed on achieving item quality with techniques, like Test Influenced Development, Automated Popularity tests, Continuous Integration. Quality is assured since it rapidly responds to changes in prerequisites and assures that will client's needs are met as the client is present full time at the improvement site. As the complete development proceeds with

small iterations having regular unit assessments and feedback by client, Agile fully addresses the needs of user. Test Driven Development makes the software program more acceptable to changes. JAD offers cross functional people to produce high-quality deliverables in the lesser span of their time. This paper aims to pay up the quality factors which might be achieved of software package built using Agile. It can be quite evident on the data reviewed that will Agile achieves requirement and design High quality Assurance using Technique Metaphor, JAD as well as On-Site customer as well as development quality warranty using pair coding, refactoring, acceptance assessments and unit assessments. These techniques although achieving sufficient quality benefits also make an effort to improve cost and amount of time in finding bugs as well as removing them.

V. FUTURE WORK

The core of Quality Assurance is that if in an organization the process to develop the products are good and are followed strictly and carefully, then the products are bound to be of good quality. The contemporary quality assurance concept includes direction for recognizing, defining, analyzing, and improving the production process. Agile includes best practices that helps improving the process of developing a product but how closely these practices are followed and to which extent is the point that needs to be investigated. The drawbacks and the assets of each practice must be acknowledged and how these practices support each other to achieve process assurance must be focused upon by the participants.

REFERENCES

[1] Beck K., "Extreme Programming Explained: Embrace Change",Addison-Wesley, 1999.

[2] Pressman, R.S., Software Engineering a Practitioner's Approach, Mcgraw-Hill, 2001

[3] Sommerville,I., Software Engineering. Addison-Wesley, 2004

[4] Ambler, Quality in an agile world. AmbySoft, Inc., 2005

[5] Pete McBreen. Mcbreen, Quality Assurance and Testing in Agile Projects, Consulting 2003.

[6] Abrahamsson, P., Ronkainen, J., Siponen, M., Warsta, J., "New Directions on Agile Methods: A Comparative Analysis", 25th International Conference on Software Engineering, 2003.

[7] Adam W., Maciej W., Wojciech C., Experimental Evaluation of 'On-Site Customer' XP Practice on Quality of Software and Team Effectiveness, OTM'10 Proceedings of the 2010 international conference on "On the move to meaningful internet systems", Volume 6428, pp 269-278

[8] Beck, K.: The Metaphor Metaphor. Invited presentation at OOPSLA (2002)

[9] Beck, K., Cockburn, A., Bossavit, L.: System Metaphor. http://c2.com/cgi/wiki?SystemMetaphor (2003)

[10] Garzaniti, R., Haungs, J., Hendrickson, "Everything I Need to Know I Learned from the Chrysler Payroll Project", SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, ACM Press,1997

[11] M. Fowler, Refactoring Improving the Design of Existing Code, Addison-Wesley, 2000

[12] R. Moser, P. Abrahamsson, W. Pedrycz, A. Sillitti, G. Succi "A case study on the impact of refactoring on quality and productivity", Balancing Agility and Formalism in Software Engineering, Volume 5082, pp 252-266, 2008

[13] T. Mens, S. Demeyer, B.D. Bois, H. Stenten, P. van Gorp,"Refactoring: Current Research and Future Trends.", Electronic Notes in Theoretical Computer Science, **82**(3), 2003.

[14] Cockburn A., Williams L., "Costs and Benefits of Pair Programming", in Extreme Programming Examined. Addison-Wesley, 2001

[15] K. M. Lui and K. C. C. Chan, "When Does a Pair Outperform Two Individuals?" XP2003, Italy, 2003

[16] M. M. Müller, "Are Reviews an Alternative to Pair Programming?" 7th International Conference on Empirical Assessment in Software Engineering, UK, 2003.

[17] S.Yenduri, L.A. Perkins, "Impact of Using Test-Driven Development", International Conference on Software Engineering Research and Practice & Conference on Programming Languages and Compilers, SERP 2006.

[18] Maximilien E.M., WilliamsL., "Assessing test-driven development at IBM", 25th International Conference on Software Engineering, 2003.

[19] Canna, J. Testing Fun? Really?, IBM developerWorks Java Zone (2001).

[20] Carmel, E., George, J.F., Nunamaker, J.F., Jr., "Supporting joint application development (JAD) and electronic meeting systems: moving the CASE concept into new areas of software development", Proceedings of the Twenty- Fifth Hawaii International Conference on System Sciences, 1992, Volume: iii,7-10 Jan. 1992, pp. 331 -342 vol. 3.