ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 4, October 2015

Extreme Programming: Aiming towards Quality Assurance

Ayesha Saad Khan, Mohammad Suaib
M.tech CSE (2nd Year), Integral University, Lucknow, India
Assistant Professor, Integral University, Lucknow, India

Abstract- Agile methodologies are among the most popular methodologies to develop software in recent times. They aim to build up software wherein requirements are constantly changing and seek to make development easy while ensuring quality. The different agile techniques like XP, Scrum etc. use best ways that help to improve Quality. Thus ensuring Software Quality Assurance (SQA) in the product delivered. The agile process extreme programming is targeted towards smaller development teams and requires relatively few details. In this paper Quality Assurance techniques of extreme programming has been focused upon. The analysis of the different techniques has been carried out on basis of important features, quality factors achieved, timing, and cost.

***Keywords*-Agile methodologies, Extreme Programming, Software quality, Quality Assurance.**

I. INTRODUCTION

Agile promotes planning that is adaptive, evolutionary development, and encourages quick and flexible response to changes in requirements. The methodologies contain a subset of iterative methods focusing on constantly changing requirements and early product delivery while maintaining quality. Extreme programming is one of the most widely used agile methodologies. It is a lightweight methodology for small to medium sized teams developing software based on vague or rapidly changing requirements [1]. The paper collects quality assurance practices in XP together and analyzes them. Software quality is the measure or degree to which a system, or process meets the requirements that are specified by the customer and fulfils the expectations of customer. Quality Assurance is a set of planned actions carried out in a systematic way to provide confidence that the software development process confirms with the requirements. The main objective of this research is to collect the various quality assurance practices of extreme programming and analyze them to gain a deeper understanding of the level and state of quality assurance in extreme programming and how this methodology aims to achieve good quality software. The analysis of quality assurance practices focuses on important features within each technique, quality factors achieved, timing of carrying out the particular technique, cost of quality assurance. The aim is to show that if all the quality assurance techniques when well combined will yield a better quality software that fulfils the quality factors of correctness, maintainability, verifiability, integrity, usability, reliability, testability etc. The paper is structured as follows. First, a short description of software quality, quality parameters, quality assurance and agile quality is mentioned. Second, extreme programming is described. In the third section of the paper the various quality assurance practices followed in Extreme programming is briefed upon. Next, the practices are analysed. The last and final section concludes the paper along with the future work.

II. QUALITY, QUALITY PARAMETERS AND QUALITY ASSURANCE

Many software engineers have defined quality as Meyer [2] defines software quality according to an adapted number of parameters that aim to achieve quality as defined by McCall [3], which are; correctness, extendibility, robustness, reusability, compatibility, efficiency, portability, integrity, verifiability, and ease of use etc. Pressman [4] 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 [5] 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.
Agile Quality Ambler [6] considers agile quality to be a result of practices such as effective collaborative work, iterative and incremental development as implemented through techniques.

Table I. Software Quality Parameters in Extreme Programming

Quality Parameters	Description
Correctness	The extent to which software meets its intended functions without failure.
Efficiency	The amount of computing resources and code required by software to perform a function.
Portability	The effort required to transfer a program from one platform to another platform.
Usability	The extent of effort required to learn, operate, and understand the functions of the software.
Integrity	The extent to which access to software or data by the unauthorized persons can be controlled.
Verifiability	How easy it is to test the system.
Extensibility	It is the measure of the ability to extend a system and the level of effort required in implementing the extension.
Maintainability	The effort required to locate and fix an error during maintenance phase.
Flexibility	The effort required to modify an operational program.
Reliability	The extent or degree to which the software performs its proposed functions without failure.
Testability	The effort required to test software to ensure that it performs its intended functions.

A. 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".

B. Agile Quality Assurance

It is the development of software that can respond to change, as the customer requires it to change [7]. Thus providing tested, working, and customer 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. EXTREME PROGRAMMING

Extreme Programming is a combination of engineering practices that support each other when used together [8]. Beck et al (2004) defined Extreme Programming (XP) as:

- XP is lightweight that is you only do what you need to create value for the customer.
- XP adapts to unclear and rapidly changing requirements.
- XP addresses software development constraints.
- XP can work with teams of any size.

Extreme Programming is based on iterations that are usually short and incremental developments with constant feedback from customers. The working stories are delivered rapidly mostly within two weeks. The customer decides further two weeks work. Thus system grows in functionality, step by step.

According to [1], Extreme programming has following phases:

1) Exploration Phase: Story cards are written by the customers for the features they want to be included in the first release. This phase takes about few weeks to a few months.

2) Planning phase: Priority order for stories is decided and the contents of the first small release are made. The programmers estimate how much effort each story requires and the schedule is agreed upon. This phase takes a couple of days.

3) Iterations to release: The outcome of the planning stage that is the schedule is broken down to a number of iterations that will each take one to four weeks to implement. The first iteration creates a system with the architecture of the whole system. The customer decides the stories to be selected for each iteration. The functional tests created by the customer are run at the end of every iteration.

4) Product ionizing phase: Requires extra testing and checking of the performance of the system before releasing it to the customer.

5) Maintenance phase: New iterations are produced. Development activities are slowed down. New People are added in the team and team structure is changed.

6) Death Phase: This phase occurs when no more changes to the architecture and design have to be made and final documentation of the system is written. There are no more stories to be implemented.

The "extreme" in the name of Extreme Programming comes from the various principles and practices that XP takes to extreme levels [1].

- If code reviews are good, code will be reviewed all the time (pair programming).
- If testing is good, frequent tests will be conducted (unit testing), even the users will test the system before accepting it (functional testing).
- If design is good, it will be made part of everyday development practices (refactoring).
- If simplicity is good, the system will be designed with simplicity supporting its current functionality.

IV. QUALITY ASSURANCE IN EXTREME PROGRAMMING

A. On-Site Customer

On-site customer is a practice where a customer has to be present and available full time for the team. The customer must know what the system should do and the developers should interrogate the customer concerning requirements when they are not sure that what the system should be doing. A real customer must sit with the team, available to answer questions, settle disputes, and set small-scale priorities. A real customer is the one who will really use the system when it is in production [1]. Effective communication and feedbacks are crucial. An increased dependence on less informative communication channels result in higher defect rates [9]. User involvement is through planning games, user stories, story cards and acceptance tests. This technique guarantees that users are able to carry their work to their own satisfaction in an effective, efficient and economical manner. Thus increasing quality and productivity. This process is basically focused on frequent and intensive involvement of customer. As no one knows customer’s business and his specific needs better than the customer himself. Experiments confirmed that On-site customer practice has substantial optimistic influence on quality of communication and speed of software production [10].Cost can be high in small projects. But an On-site customer practice improves communication and informal validation and increases usability and correctness.

B. System Metaphor

The system metaphor is a means of communicating about the project in terms that both developers and customers will understand, and which does not require pre-existing familiarity with the problem domain [11]. System metaphor helps the customer to communicate with the developers using a shared vocabulary in terms understood by both developers and customer that helps in defining the framework of the system. The customer should be comfortable to talk in terms of metaphor. According to [1] through metaphor we are likely to get architecture that is easy to communicate and elaborate. We need to emphasize on the goal of architecture which is to give everyone a coherent story within which to work. System metaphor increases the interaction between customers and developers, which is an important factor in XP for the success of iteration. It helps increasing maintainability, efficiency, reliability.

C. Pair Programming

Pair programming is a practice where the production code is written by two people at a single system. Each of the participants has his own role to play. One person's responsibility is to focus on current method and its implementation while writing the code and the other person has a more strategic work of examining whether the current approach will work or not and thinking about other ways to deal with current problem. XP suggests using pair programming for all production code all the time. An experiment was conducted to determine the efficiency of pair programming as compared to programming by a single head it was observed that 15% more time on the program was spent than with individual. Costs do not increase and resulting code has about 15% fewer defects [12]. It was also concluded the increase in development costs about 15% with pair programming was recovered in the reduction of defects. Pair programming is carried out almost all the time during the development of system. Pair programming can improve the design quality factors such as correctness, verifiability, testability while reducing defects. Rotation of pairs is also carried out to achieve knowledge transfer within pairs.

D. Refactoring

Refactoring is the technique of modifying the existing source code without changing its external behaviour. According to [1] Refactoring is a method of changing the existing program to make adding a feature simple and to look up ways to make a program simpler while running all the tests. Before doing refactoring of a section of code, a set of unit tests are performed, so as to confirm that the behaviour of the module is correct before refactoring. After refactoring of the code, the unit tests are run again to verify that refactoring doesn't break the tests. Refactoring is performed when the need arises. Refactoring is an iterative process of making small changes in program, testing and then making further changes. The cost of doing refactoring is approximately same as the cost of not doing it as it helps in providing easy to maintain software thus improving the cost and time spent in maintainability. It improves reliability, providing an internal architecture that is more expressive thus improving extensibility. Refactoring may also resolve hidden, latent or undiscovered bugs.

E. Test Driven Development

Create a test, Run the test, Make changes until the test passes [6]. Unit tests are written by the programmers to gain confidence in the operation of program. Testing makes the program more acceptable to changes, while building the confidence in the system gradually with time. These tests are written before writing the source code and running after its implementation, thus they are executed throughout the development of the software. For developer's confirmation of code such unit tests are performed by developer contrary to acceptance testing performed by the customer for his satisfaction in the system. Test-Driven development increases testability and reduces defects in the code. It helps in obtaining a high coverage as tests are written for every production methods that could break. The costs incurred in performing these tests are relatively similar to writing tests afterwards [13].

F. Continuous Integration

Continuous Integration is an accepted practice among agile methods in which the developers of a team integrate their work regularly. Automated builds and test processes are developed that allows a team to build and test their software many times a day. Many teams find that this approach leads to considerably reduced integration problems. The essence of it lies in the simple practice of everyone on the team integrating frequently, usually daily. The key point is that continuous integration catches enough bugs to be worth the cost [14]. It helps improving maintainability, testability, integrity while reducing the time spent in finding bugs.

G. Automated Acceptance Testing

Acceptance tests are written by the customers to verify that the system's functionality is not in conflict to the requirements and expectations from the system. Acceptance tests in XP are performed much earlier and more often than in traditional approaches to software building. Automated testing is preferred in XP and is the process of executing automated acceptance tests rather than executing them manually. Any program feature without automated tests simply doesn't exist [1]. Tests should ensure proper coverage and should work like normal acceptance testing. Extreme Programming uses scripted tests, although this makes updation of tests costly but running them is cheaper compared to manual testing [15]. Automated acceptance tests must be ready by the middle of iterations and should run daily. It improves reliability and flexibility. Automation of tests also saves time and money.

V. ASSESSMENT OF THE QUALITY ASSURANCE TECHNIQUES IN XP

There are various techniques in Extreme Programming that when followed closely will yield good quality software and promises quality assurance in the processes used to develop the product using Extreme Programming. An assessment of the various techniques followed in Extreme Programming has been carried out to figure out the important features of each techniques that focus on the issues they deal with that is their area of work in Extreme Programming, the quality parameters each one of them achieves, the costs incurred in each technique and the

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 4, October 2015

timing of carrying out each one of them in the process of development.

TABLE II. ASSESSMENT OF THE QUALITY ASSURANCE TECHNIQUES IN EXTREME PROGRAMMING

Practice	Important features	Quality parameters achieved	Cost	Timing
On-Site Customer	Real, full-time user to answer queries	Usability, Correctness	Customer's presence at development site	All the time during development
System Metaphor	Means of communication between developer and user	Maintainability, Efficiency, Reliability	Customer's presence at development site	As needed during each iteration
Pair Programming	Code written by two people at a single system	Correctness, Verifiability, Testability, Efficiency	Similar to programming alone	Throughout development
Refactoring	Modifying the source code without changing its external behaviour	Maintainability, Reliability, Extensibility	Increase in cost is recovered in maintenance phase	As and when needed
Test Driven Development	Unit tests written before writing the source code and executing after its implementation	Reduces Defects and increases Testability, Reliability	Similar to testing afterwards	Throughout development
Continuous Integration	Frequent daily integrations in the system	Maintainability, Portability, Testability, Integrity	May depend on speed of build and tests	Several times a day
Automated Acceptance Testing	Automated tests defined by user to verify system's functionality	Reliability, Flexibility	Expensive to build and update but cheap to execute	After each iteration

VI. CONCLUSION

Extreme Programming (XP) follows the best practices to achieve quality in the software. It aims on achieving product quality with techniques, like Test Driven Development, Automated Acceptance tests, Continuous Integration. XP work towards quality assurance as it rapidly responds to changes in requirements and assures that customer's needs are met as the customer is present full time at the development site. As the complete development proceeds in small iterations with regular unit tests and feedbacks from customer, XP fully addresses the needs of customer. Test Driven Development makes the software more acceptable to changes. XP is based on iterations that are usually short and incremental developments with constant feedback from customers. This paper has covered up the quality factors that are achieved of software built using XP. It is quite evident from the data reviewed that XP achieves requirement and design Quality Assurance using System Metaphor and On-Site customer and development quality assurance using pair programming, refactoring, acceptance tests and unit tests. These techniques while achieving sufficient quality benefits also aim to improve cost and time in finding bugs and removing them. Thus improving overall customer's satisfaction.

VII. FUTURE WORK

The basic essence 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. XP 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 weaknesses and the strengths of each practice must be recognized and how these practices support each other to achieve process assurance must be thoroughly worked upon by the participants.

REFERENCES

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

[2] Meyer B., Object-Oriented Software Construction, Prentice Hall PTR, pp.4-20, 2000.

[3] McCall,J.A., Richards,P. K., & Walters, G. F., Factors in Software Quality, Vols.1, 2 and 3, National Technical Information Service, 1977.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 5, Issue 4, October 2015

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

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

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

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

[8] Jeffries R., Anderson, A., Hendrickson, C., "Extreme Programming Installed", Boston: Addison-Wesley, 2001.

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

[10] 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.

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

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

[13] George, B., Williams L., "An Initial Investigation of Test-Driven development in Industry", Proceedings of the ACM symposium on applied computing, March 2003.

[14] Martin Fowler, Continuous Integration, www.martinfowler.com/articles/originalContinuousIntegration.html.

[15] Olli P. Timperi, "An Overview of Quality Assurance Practices in Agile Methodologies", Seminar in Software Engineering, SPRING, 2004.

AUTHOR'S PROFILE

Ayesha Saad holds a B.Tech degree in Information Technology from B.B.D.N.I.I.T, Lucknow. Currently pursuing M.tech in Computer Science from Integral University, Lucknow, India. Mobile No.: 8009954776. Email id: emu24jan@gmail.com

Mohammad Suaib is pursuing Phd from Integral University, Lucknow, India. Mobile No.:9044016183 Email id: suaibcs09@gmail.com