International Journal of Database Theory and Application
Vol.7, No.3 (2014), pp.83-90
http://dx.doi.org/10.14257/ijdta.2014.7.3.09

Modeling Software Maintainability and Quality Assurance in the Agile Environment

Priyanka Upadhyay, Abhishek Singh and Naveen Garg

Department of Information Technology
Amity School of Engineering and Technology
Amity University Noida, Uttar Pradesh, India
{priyanka.upadhyay0991, singhabhishek.0815,er.gargnaveen}@gmail.com

Abstract

Software Maintainability is the ability of the system to undergo changes with a degree of ease and Quality assurance refers to planned and systematic production that provides confidence in a product. This paper describes the different issues of software maintainability and quality assurance in the agile environment. This paper also presents the different metrics that improve the overall quality of the software maintainability in the agile environment. This paper proposes the model of software maintainability and quality assurance in the agile environment. The approach that is used is based on the customer requirements by iterative interaction with customers to provide the best requirement with assured quality.

Keywords: *Software maintainability, Software quality Assurance, Agile Environment*

1. Introduction

Agile software development is a well known concept these days. The existing software development life cycle involves an efficient lookout for best efficiency of a software development. But despite all these efforts still the software's fail and give us undesired outputs. In this regard agile testing comes out with a very effective mechanism where the software is tested at each and every step as per the user specifications. Agile testing completely relies on the feedback that is provided from various stakeholders. Visualization is most critical to communicating the outcome of a simulation to a non-technical audience such as decision makers or the user itself. When it comes to maintainability and quality assurance in Agile software development it becomes one of the primary goal to deliver an efficient product to the customer by removing problems such that the code quality is not affected. Software Maintenance is the most important phase of the software development life cycle which requires rigorous efforts to satisfy the customer by delivering the best product with efficient maintenance. Maintainability also includes the very basic requirement that comes up in need with time i.e. the flexibility at the time of upgrading a software and system requirements as per the user needs. As we know that user requirements change with time and at any time interval it might change, so a developer should design the software such that it is maintainable at any phase of software development life cycle.

Quality of the software product has always been the ultimate goal of every company. But when talking about the Agile environment we need a strong collaboration between the customer and designer to best meet up with quality which can be easily done by involving the customer onsite so that useful feedback can be utilized efficiently[6].Constructing a 100% correct system is difficult task. Also better quality never always means that the product developed is free from all bugs or errors but might be it is acceptable by the customer if

ISSN: 2005-4270 IJDTA

meeting the specified requirements. It is considered that a software system is of better quality if the frequency of fault is acceptable while execution of a software. Agile software testing is a complete new approach that works towards providing a more maintainable and high quality software systems by testing it with agility and providing best results.

2. Issues of Software Maintainability and Quality Assurance in the Agile Environment

2.1. Delivery over Quality: It focuses the gradual development of the software focussing on its quality aspects. Also early visibility and recognizing errors at an early stage help us to redefine our software with best quality and developing best customer relationship. Continuous software development is little challenging task but merging it in the agile testing environment it ensures us to make better relationship with our customer by continuously interacting with them[4].

2.2 Development over Planning: The second shortcoming that targets the principle of responding to change over the plan is major concern. Practically, the designers code while referring the need of their customers and stakeholders to define overall design and testing specifications [5].

2.3 Prioritization: It is essential to prioritize the scheduling tasks of various programs in the agile environment so as to deliver best product quality in terms of quality assurance and maintainability.

2.4 Modularity: It is the key to support extended development process in the agile development over extended periods of time. If we can isolate our software product and its lines of code into small groups that can independently communicate with each other then we can easily enhance the features at any development stage of the product.

2.5 Quality Management: Quality management is a major concern in the agile development and can be easily enhanced when testers come up with stable requirements i.e. less test cases as less rework and maintenance has to be done.

2.6 Degree of Change of Requirements: As we move to the agile testing environment we come across rapid changes in the requirements of the customer which imposes a more challenging task on the testers. As the Agile environment welcomes the change in requirements even late in the development phase [2].

2.7 Adequate Unit Testing: We often see two problems confronting in the unit testing that conquers with Quality i.e. Unit testing has a limited bug finding and less effectiveness when there is a huge lines of code to be tested.

2.8 Poor, Varying and Missing Test Oracles: Agile testing focuses on face to face conversation with the customer. Even with sufficient test oracles Agile testing team requires to embrace the change. These test oracle problems put upto 20% to 30% test inefficiencies.

2.9 Long Hour Meetings and Sprint Durations: Sometimes when the commitment to deliver the software product is short then it ultimately becomes a squeezing shot for the agile testers as they have to work hard meeting the correct specifications and assuring best quality and maintainability in the software product. To resolve this issue maturity of the team is required.

3. Different Metrics and Their Effect on Software Maintainability in Agile Environment

Different metrics help us to improve the overall quality of the software and enhance quality of the software product. Implementation of the available and upcoming metrics will help us to standardize the quality of the software product. Strict following of the metrics helps us to prevent future errors in the software to improve software products maintainability and reliability.

Metrics for the Maintainability

Table I. Metrics and Their Effect on Software Maintainability in Agile Environment [1]

Reliability factor	The extent upto which the software gives best performance without any error or failure.
Efficacy	The level upto which a software best uses its available resources with maximum utilization.
Usability	The ease of use of the software for a user defines the usability.
Testability	The ability of a software product to check the acceptance rate of errors supporting the evaluation criteria of a software product.
Portability	The ability of the software to be used effectively in various operating environments.
Complexity	The different path flows in the line of code of the program and its ease of understanding to the user. More complex code increases more test cases and becomes less efficient.
Modularity	Independent execution of the components in the program and their clarity of usability increasing its reliability and efficiency.
Class Coupling	It can be quite difficult to maintain the Types and Methods of those that have a relative high class coupling although it is a good practise having Methods and types that function of low coupling and high cohesion.
Maintainability Index	It can be defined as software metric which is capable of measuring the ease of support and change in the source code. It can be calculated as a factored formula that consists of SLOC i.e. source lines of code, cyclomatic complexity and Halestead Volume.

Lines of code (LOC)	The source line of code denotes the size of a computer program. It is a metric to depict the effort required to develop a program. It becomes the responsibility of the programmer to design the program such that the complexity is reduced and the testers find it easy to maintain and quality is achieved.

4. Model of Improvement in Software Maintainability and Quality Assurance in Agile Environment

Agile software development has helped us to value the customer requirements by iteratively interacting with the customer to best meet the requirements with assured quality. If at any step we face any problem immediately agile software development adapts customer requirements and respond to change in an efficient manner. This process focuses on iterative efficient delivery of software processed in agile testing environment.

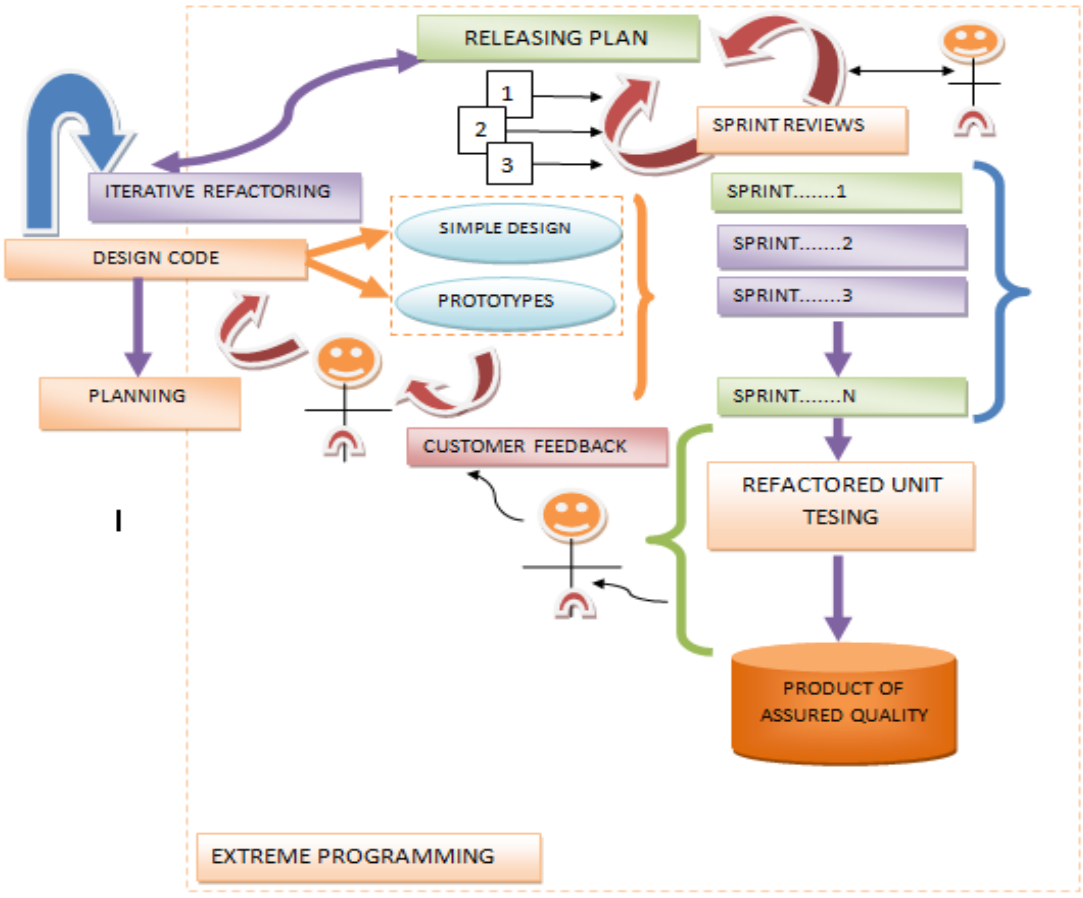

Figure 1. Agile Testing Software Development Model Assuring Maintainability and Quality

In the above described Agile software development model the main emphasis is given on the customer interaction and iterative analysis of the developed code. Initially the customer requirements are gathered and the simple design code and prototype is generated. The built prototype is sent for the customer feedback. If the developed plan is acceptable by customer

and only few other requirements are added further then the plan is released and sent for iterative execution.

Then further in sprint execution it is sent for acceptance testing and Stand up meeting where if still any error exists then again interaction with the customer is conducted for further analysis to improve Quality of the developed software product. Further the code is given to the pair programmers to further check if there is any error.

At the end to assure the Quality and improve the internal structure Refactored unit testing is implemented which further removes errors if any.

This total framework is executed in extreme programming where the agile team has to review whole software development [7].

In agile environment the plans are very short lived and iterative planning has a heavy emphasis on the software construction activities.

4.1. Extreme Programming: It begins by creating user requirements and it is the duty of every agile team to review whole software development and assign cost incurred. In Extreme programming new versions are built several times per day and increments are delivered to customers frequently [8].

The whole procedure is divided into sprints. Before programming starts it is better if we opt for unit testing and encourage a healthy team of pair programmers to achieve better quality product. Extreme programming focuses on daily execution of unit testing. Acceptance testing is defined by having a strong collaboration with customer and designer.

4.2. Refactoring: It is a process of making changes to a software system in a manner that while improving the internal structure it does not alter the external behaviour of the code. Refactoring helps us to find faults, So as to make the development code efficient in our model we apply the refactoring cycle in the designing phase and build prototypes. If any error is occurred it is resolved and fixed the same moment which helps us reduce the efforts of the testers at the end [3].

4.3. Refactoring Cycle: It basically frames with the very basic and simple to understand coding and designing that can be easily tested upon and is less complex. The necessity of refactoring arises from the fact that as the business environments are rapidly changing the responding task becomes more challenging [12]. As the challenge increases often the businesses may be willing to accept the lower quality product if all the functionality is acceptable. So the refactoring cycle helps us to modify the system efficiently with high quality by reworking on the simple design and prototypes.

4.4. Acceptance Testing: It is quite similar to the black box system testing and represents some expected result from a software system. Then the customers verify the correctness of acceptance testing and review test scores. We need to develop acceptance tests for every iteration. Main focus of implementing acceptance testing in extreme programming is Quality assurance [9].

4.5. Stand up Meeting: This meeting is scheduled for a very short period of time and are intended for reviewing the work done. This meeting allows the team members to easily share

their own status and see others status as well. In these meeting discussions about quality envisioning by following extreme programming is followed for efficient software development.

4.6. Pair Programming: The mechanism of pair programming involves two programmers working on a single task. As per the research made so far considering pair programming in agile environment results show that short term productivity might decrease but due to the high quality produced code the resulting long term productivity and quality goes high.[12]

4.7. Refector Unit Testing: Main aim of the refactored unit testing is to make smallest possible refactoring to get the code into a testable state. Unit testing is the most important step which makes small changes incrementally. Frequent refactoring makes the design code more efficient and improves the quality.

5. Advantages of the Agile Testing Software Development Model Assuring Maintainability and Quality:

5.1. Improved customer interaction by seeking customer assistance after short iterative completion: As we can see in the above described model the customer interaction is deployed at short iterations so that at any point customer requirements are fulfilled and customer gets the product with assured maintainability and Quality.

5.2. Defects are Detected at an Early Stage: Due to customer interaction at various phases of the model the defects that might creep in are detected at a very early stage and resolved so as to deliver the product with quality in much lesser time.

5.3. Flexibility to new Requirements: After the testing team discusses with the customer and the customer wants to add some more requirements then in this model we can easily add as per the choice of customer and then again the testing cycle in the refactored way is executed with extreme programming.

5.4. No Useless Meetings: This model also eliminates unnecessary meetings to discuss what customer wants and meeting is done only when the prepared documentation is to be discussed for customer approval if he is satisfied with the specifications. This further reduces the documentations as well and only required work is done for faster product delivery [11].

5.5. Early Access to Software for Testing: This model gives the testers an early access to the software product developed to be checked whether it meets the customer satisfaction and is meeting the Quality standards.

5.6. Reduced Risk: As the customer is involved at various levels of the model so the chances of not satisfying the customer are less which happens with other models where the customer gets the product after it has been fully developed.

5.7. Increased Transparency and Better Visibility: The above model has been designed keeping in view the mutual coordination among the stakeholders and developing a bond of trust so that when the product is being shaped up each stakeholder has a better visibility.

6. Conclusion

This paper deals with the software maintainability and quality assurance in agile environment, the maintainability of the software depends on the several factors like size of the projects, availability of the customers, knowledge of the projects personnel and various different factors which affect the software maintainability and quality Assurance.

Agile methods provide an efficient software product by delivering it early to the Working Software environment, simplifying communication and increasing the customer satisfactions.

In Agile, there is a continuous interaction with the customer, so according to the need of the customer, the new features are added to satisfy the customer which reduces the time and cost which further help in modelling the software maintainability and quality assurance of software.

References

[1] Osama Sohaib and Khalid Khan, "Integrating Usability Engineering and Agile Software Development::A Literature Review" published in Computer Design and Applications (ICCDA), 2010 International Conference page(s) V2-32 –V2-38, June 2010.
[2] Hans-Peter Samios, "Overcoming Traditional Project Release Reporting with an Agile Approach Focused on Change" in Agile Conference (AGILE) 2012, page(s) 131-135, August 2012.
[3] Amani Mahdi Mohammed Hamed and Hisham Abushama, "Popular Agile Approaches in Software Development: Review and Analysis" published in Computing, Electrical and Electronics Engineering (ICCEEE), 2013 International Conference, page(s) 160-166, August 2013.
[4] Ahmed, A.; Ahmad, S. ; Ehsan, N. ; Mirza, E. ; Sarwar, S.Z., "Agile Software Development: Impact on Productivity and Quality" published in Management of Innovation and Technology (ICMIT), 2010 IEEE International Conference, page(s) 287-291, June 2010.
[5] Lagerberg, L. ; Skude, T. ; Emanuelsson, P. ; Sandahl, K. ; Stahl, D., "The Impact of Agile Principles and Practices on Large-Scale Software Development Projects: A Multiple-Case Study of Two Projects at Ericsson" published in Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium, page(s) 348-356, October 2013.
[6] E. Mnkandla and B. Dwolatzky, "Defining Agile Software Quality Assurance", published in international conference Software Engineering Advances, October 2006.
[7] Gu Hongying and Yang Cheng, "A customizable agile software Quality Assurance model" published in Information Science and Service Science (NISS), 2011 5th International Conference on New Trends in (Volume:2), page(s) 382-387, October 2011.
[8] Andrew Marrington, James M. Hogan and Richard Thomas, "Quality assurance in a student-based agile software engineering process", published in Software Engineering Conference, 2005, page(s) 324-331, April 2005.
[9] Scharff, C., "Guiding global software development projects using Scrum and Agile with quality assurance" published in Software Engineering Education and Training (CSEE&T), 2011 24th IEEE-CS Conference, page(s) 274-283, May 2011.
[10] Sonali Bhasin, "Quality Assurance in Agile: A Study towards Achieving Excellence" published in AGILE India (AGILE INDIA), 2012, page(s) 64-67, February 2012.
[11] Wang Xiaohua, Wu Zhi and Zhao Ming, "The Relationship between Developers and Customers in Agile Methodology" published in international conference on Computer Science and Information Technology, 2008 page(s) 566-572, September 2008.
[12] Mauricio Finavaro Aniche, Guilherme de Azevedo Silveira, "Increasing Learning in an Agile Environment: Lessons Learned in an Agile Team" published in Agile Conference (AGILE), 2011, page(s) 289-295, August 2011.

Authors

Ms. Priyanka Upadhyay was born on February 9, 1991.She is pursuing her Masters in Information Technology at Amity university Noida, India. Her research interests include software engineering, cloud computing, cyber security etc. She received her Bachelor's of technology from Gautam Budha Technical University Uttar Pradesh, India where she was awarded with the "Chandrakanta Academic Excellence Award 2013".

Mr. Abhishek Singh was born on August 15, 1987. He is pursuing his Masters in Information Technology at Amity University Noida, India. He completed his Bachelor's of technology from Jaypee Institute of Information Technology Noida, India. His area of interest is Software Engineering, DBMS, and Algorithm. He qualified the GATE exam conducted by IIT.

Mr. Naveen Garg completed his M.tech from Thapar University, Punjab. Currently he is working as Assistant professor in Amity University, Noida. He has published many papers in national and international journals. His area of interest is software engineering and software testing.