ISSN(Online) : 2319 - 8753
ISSN (Print) : 2347 - 6710

International Journal of Innovative Research in Science, Engineering and Technology

Volume 3, Special Issue 3, March 2014

2014 International Conference on Innovations in Engineering and Technology (ICIET'14)

On 21st & 22nd March Organized by

K.L.N. College of Engineering, Madurai, Tamil Nadu, India

The 3C Approach for Agile Scrum Software Methodology

Jisha Johns, Akhil P Sivan, Prof. K Balachandran, Prof. B R Prathap

Dept of Computer Science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer Science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer Science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Dept of Computer Science & Engineering, Christ University Faculty of Engineering, Bangalore, India.

Abstract— Agile software development methodology is becoming one of the widely used Software Development methodologies because of its light weight methods and its focus on customer satisfaction. Continuous Integration is the only technique which is used in Agile, to ensure the Software quality of the deliverable. It is the process of integrating the new source code developed to the base code, automated compiling, building the application and running the tests. SCRUM is one of the most popular Agile methodologies used in Software development. This paper introduces the implementation of 3C Approach in SCRUM. The 3C Approach adds Continuous Measurement and Continuous Improvement as subsequent activities to Continuous Integration, for ensuring quality. Continuous Measurement is the process where the metrics and measurements, which helps in the ensuring the software quality of the deliverable, are considered. Agile methodology emphasize on Agile metrics, which helps in tracking the project success and customer satisfaction. Metrics from Traditional methods will also be helpful, if combined in the right way. So, in the Continuous Measurement phase, along with Agile Metrics, traditional metrics are also considered. Continuous Improvement process helps in interpreting the measurement and metrics for planning the improvement tasks for achieving better Software Quality.

Keywords— Agile, Extreme Programming, SCRUM, 3C Approach, Sprint

I.INTRODUCTION

Software organizations implement one of the software development methodologies, which would help in the planning and controlling the process of developing a software product. Many models exist which helps in the softwaredevelopmentprocess.Onethemtwawidelusedinma nyofthsdevelopmenprojectsalmodelincsthesoftwareevelop mentactivitilikplan,implncationandaintenanceinordahwi.1. [2]Onlyafter each activity is finished, the next activity will be initiated. And also, it is a heavy weight model which required

extensive documentation. [4] And, one of the major disadvantages was, the requirements should be defined at the start of the project. Hence, this model could not be used where the requirements are constantly changing and also not suitable for complex projects.

Agile is one of the software development methodologies which is widely used in many of the small and medium projects in software organizations. [13] It is a group of software development methodologies which includes SCRUM, Extreme Programming, LEAN etc. It is also an iterative and incremental development methodology. Agile allows changes in the requirements for developing a software, which results in the improvement of the project efficiency, productivity etc.

Scrum is one of the Agile software development methodologies used in many of the software organizations for software or product development[14]. Scrum accepts the fact that there will be change in requirements, change in technology etc. Scrum can effectively increase the efficiency of the project by effectively collaborating between the teams and also focuses on delivering a high quality product on-time.

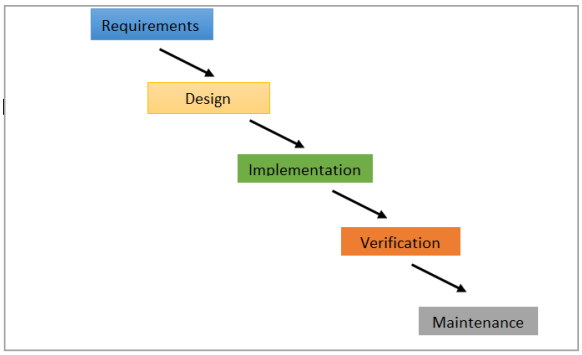

Fig. 1 Waterfall model

A sprint or iteration in Scrum, is the duration that is fixed in advance and is normally one week to four weeks. The three roles in Scrum: the product owner, the Scrum Team and the Scrum Master. [8] Each sprint is started by a Sprint planning meeting where the Product Owner, the Scrum Master and the management team participates and the user stories for the sprint are identified. The Scrum Master and the scrum team identifies the tasks, resources and the effort required to deliver those user stories. Once the stories are finalized, there will not be any re-prioritization or changing the scope of the user stories[6]. During the Sprint, there will be Stand Up meetings daily, where the Scrum Master and the Scrum Team (others may attend the meeting) discuss the progress of the user stories and the impediments/blockers are identified.

The daily Stand Up meetings or the Scrum Meetings are one of the essential component of Scrum. [5] The Sprint Review Meeting, at the end of the Sprint, is where the Scrum Master would review the project progress, demonstrate the features that were developed, to the Product Owner. The sprint is ended by a sprint retrospective meeting. In this meeting, the progress is reviewed and lessons learnt from this sprint are identified, that could be implemented in the next sprint[15].

Scrum emphasizes on delivering the working product at the end of the Sprint - a product/application that is integrated, fully tested and potentially deliverable.

Extreme Programming (XP) is another agile software development methodology, which includes several iterations that ends with releasing a product with zero or fewer defects. This methodology is intended for delivering high-quality software more productively.

II. THE 3C APPROACH

Continuous Integration is the only activity in Agile that ensures that the software product delivered is of high quality. Continuous Integration phase involves the process of integrating the new source code into the base code, which the developer has developed and also includes compiling, build & test execution[7]. Continuous Integration is common for any agile development methodology.

Continuous Integration is usually done by Continuous Integration Engines - JUnit Framework, Apache Maven etc. for Java applications. The developer will commit the code in his Integrated Development Environment (IDE) after implementation. These CI engines will check the Version Control Systems like Subversion/SVN, Concurrent Version System/CVS etc. for any new committed source code. If new Source Code is found, the Continuous Integration Engine compiles, builds the application and runs the tests.

The 3C Approach introduces a subsequent activity called **Continuous Measurement** to measure Software Metrics. Metrics is one of the key aspects of every software development process. Software metrics help in tracking project progress, achieving quality assurance, tracking project performance, estimating the project cost etc. It also helps in quantifying the size or complexity of the project, the number of defects reported etc.[12]

Agile software development methodologies rely on agile metrics in assessing whether the quality requirements and customer needs have been met. Agile metrics focus on the project success and customer satisfaction.

In the Agile projects, metrics that are used in the traditional methods are not considered. Traditional metrics focus on measuring the productivity and the quality of the software product. Agile projects would benefit, if traditional metrics and measurement approaches and Agile metrics are combined in the right way. But integrating the traditional Quality Assurance metrics and approaches into Agile Processes is the biggest challenge.

Traditional metrics like Lines of Code (LOC), coding standard violations etc. could also be measured in Agile projects. For measuring traditional Metrics in Java, many tools like FindBugs, PMD and Checkstyle can be integrated into Continuous Integration Engines. FindBugs is for detecting potential programming mistakes, Checkstyle for finding violations of Coding Standards and PMD is a hybrid-version of FindBugs & Checkstyle. For measuring the LOC, many tools like LocMetrics, CLOC etc. could be used. But measuring the LOC would be of more benefit in Agile Extreme Programming, since it could be more useful in the case of maintenance projects.

Also some of the Agile metrics - the number of tests and the test coverage could be measured during the testing process. The test coverage could be measured using many tools like Cobetura, Sonar etc. This is a helpful metric in agile frameworks like Scrum, Extreme programming etc. With the number of tests and the Lines of Code (LOC) the Test-Growth-Ratio could be calculated. The Test-Growth-Ratio has much significance in maintenance projects, where testing is given more importance. And the number of broken builds is also a helpful metric in ensuring the software quality[16].

The measurement results can be put into graphs, which helps in analyzing the changes of the measures/metrics and software quality over time. This will help in deriving the right improvement steps and also in assuring software quality of the product.

The 3C Approach introduces another activity called **Continuous Improvement** along with Continuous Integration and Continuous Measurement. This process helps in the interpretation of the measurement results from the graphs and also helps in the planning of improvement tasks for achieving better Software Quality[8].

III. THE 3C APPROACH IN SCRUM

The 3C Approach in Scrum also has the three activities as mentioned above – Continuous Integration, Continuous Measurement and Continuous Improvement.

As explained above, **Continuous Integration** is the integration of new source code developed by the developer to the base code, along with automated compiling, building the application and executing the tests. Many Continuous Integration Engines like JUNIT, Apache Maven etc. could be used, which continuously checks the version control systems for any new code that is available. These CI Engines would retrieve the new code, compiles it, builds the application and runs the test[9].

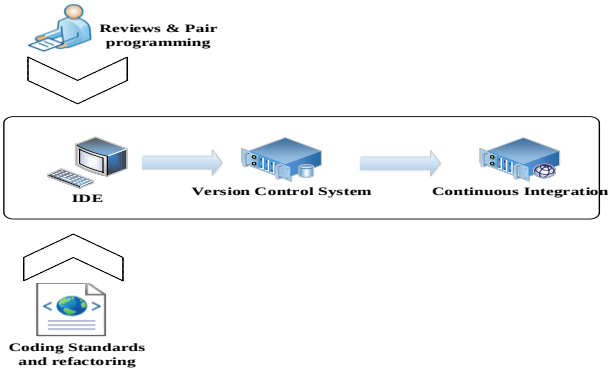

Fig 2. Continuous Integration

Continuous Measurement measures the software quality of the deliverable with the help of metrics.

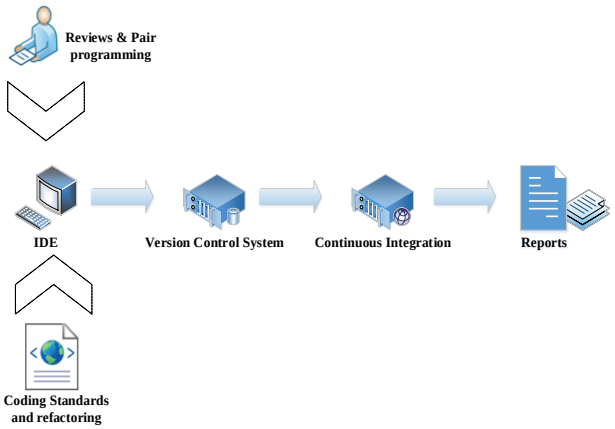

Fig 3. Continuous Measurement

The Continuous Integration Engines could provide some of the Agile Metrics like number of tests, Test Coverage, number of broken builds etc. The Test Coverage can be measured using many tools like Cobertura, Sonar etc. which are plugins and could be integrated with the CI Engines.

Some of the Agile metrics that would help in quantifying the quality of the product as well as the project progress are:

1) Number of Tests: The Number of Tests is a helpful Agile Metric and could be provided by the CI Engines. Although, the total number of tests has little significance in Agile, it provides first insights like LOC does in the context of complexity.[1]

2) Test-Coverage: The Test-Coverage is also an Agile Metric which measures how much of the Source Code is covered during Test Execution. For measuring the Test-Coverage, Line-Coverage and Branch-Coverage are considered. Line-Coverage measures the Lines of Code that has been covered whereas Branch-Coverage measures the code based on the number of branches like if-else statements[14]. The Test-Coverage helps in quantifying the percentage of the source code that is tested by the Test Suite. The Test Coverage should be 80% to 90% for any projects in the Software Industry.

Test Coverage = Code covered by the tests / Complete code with $0 \leqslant$ Test Coverage $\leqslant 1$[1]

3) Test-Growth-Ratio: Test-Growth-Ratio is also an Agile Metric which has more significance in development projects than maintenance projects. The number of Tests should increase when the Source Code increases unless code refactoring has been done. In that case, number of Tests may decrease, after removing the functionalities which are not needed anymore.

Test-Growth-Ratio = Δ Tests / Δ Source Code with (usually) Δ Source Code $\geqslant 0$, Δ Tests $\geqslant 0$[1]

4) Number of Broken Builds: Continuous Integration Engine integrates the new source code to the base code. A build is counted as a 'Broken Build', when it fails. A broken Build has much significance in Agile because it shows that the code developed has not met the required Quality Requirement. A Broken Build might be a potential defect which could fail after the release to the customer.

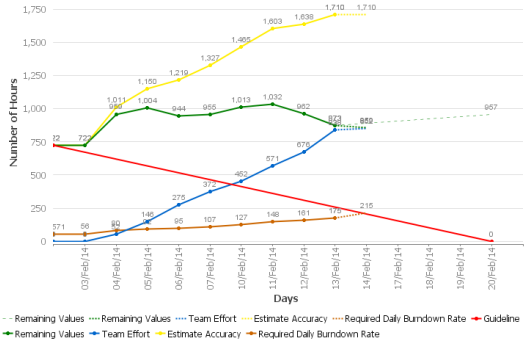

Fig 4. Burn-down Chart

5) Sprint Burn-down: Sprint Burn-down is one of the Agile metrics, that is very helpful in tracking the effort burn-down of each Sprint on a daily basis.The burn-down shows the remaining effort that has to be burnt in Sprint, how much effort has been logged so far, required burn down rate etc. Since this chart provides daily feedback on the effort, traditional metrics like Effort overrun could be tracked early in the Sprint and mitigations can be planned early rather than waiting till the end.

6) Sprint Velocity: Sprint Velocity is a powerful Agile Metric. It helps in identifying the volume of work that could be completed in each sprint with the available team capacity.

Sprint Velocity = Total Effort Available for each Sprint / Total Story points of that Sprint.

7) Release burn-up: Release burn-up is also one of the agile metrics, in measuring the volume of work delivered to the customer in terms of Story Points.

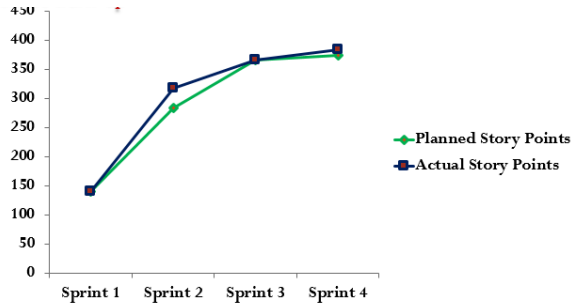

Fig 5. Release Burn-up chart

Along with the Agile Metrics, some of the traditional metrics that could be taken into account are:

- Coding Standard Violations
- Violations from Static Code Check
- Review Effectiveness
- Defect Removal Efficiency
- Test Metrics

The above metrics would help in determining the measures for reducing the defects over time.

1) Coding Standard Violations: Tools like CheckStyle, PMD etc. could be integrated with the IDE (Eclipse & NetBeans for Java Applications), would help the developer to identify the violations before committing the code to the version control systems. This will prevent the release of deliverable to the customer with critical violations.

2) Violations from Static Code Checks: Tools like FindBugs, PMD etc. could be integrated with the IDE (Eclipse & NetBeans for Java Applications), would help the developer to detecting potential programming mistakes and could be fixed by the developer before committing the code. This will prevent the release of the deliverable to the customer with critial violations. PMD is a hybrid-version of FindBugs and CheckStyle[11].

3) Review Effectiveness: The Review Effectiveness measures how effective the review methods implemented in the project are, in detecting the defects. Code reviews or peer reviews help in identifying the defects in advance, potential mistakes made by the developer and also, helps in determining whether the code developed by the developer is according to the requirement.

There are many review application tools available - Crucible, Collaborator etc., which would help in integrating the code review into the development process. [3]

Review Effectiveness = Total number of defects found before releasing the user stories for testing in each Sprint / Total Number of defects of that Sprint

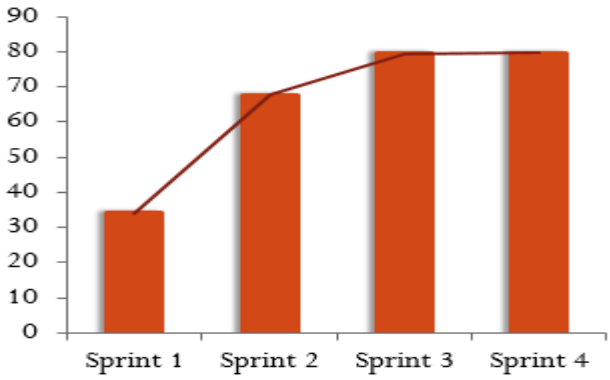

Fig 6. Review Effectiveness graph

4) Defect Removal Efficiency: This is also a traditional metric which could be incorporated in the Agile process. The Defect Removal Efficiency (DRE) measures how effective the defect removal methods are, which in turn determines the quality of the product.

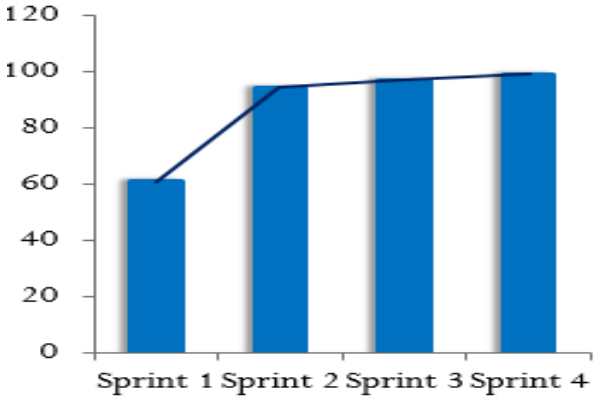

Fig 7. Defect Removal Efficiency graph

Defect Removal Efficiency = Total number of defects found internally (before the releasing the user stories) of each Sprint / Total Number of defects of that Sprint

5) Test Metrics: It is also one of the traditional metrics which helps to quantify the effectiveness of the testing. It focuses on identifying the quality defects of the deliverable. It helps in identifying the number of test cases that are passed while testing, number of test cases failed and how effective the testing methods are in finding the defects. The inputs for Test Metrics are Number of Test cases prepared for each user story. Test Metrics include:

a) Pass Rate: In scrum, the Pass Rate helps to identify the Number of Test cases that are successfully executed for each user story.As we move from one Sprint to another, there should be a increasing trend for this metric.

Pass Rate = Total number of Test Cases passed for all user stories in that Sprint / Total Number of Test cases for that Sprint

b) Fail Rate: In Scrum, the fail rate helps to identify the Number of Test cases that are failed in the Test Execution for each user story. As we move from one Sprint to another, there should be a decreasing trend for this metric.

Fail Rate = Total number of Test Cases failed for all the user stories in the Sprint / Total Number of Test cases for that Sprint

c) Test Effectiveness: Test Effectiveness helps to measure the effectiveness of the testing in identifying the defects. This will help in measuring how effective the test cases are in finding the defects.

Test Effectiveness = Total number of test cases that resulted in defects for each Sprint / Total number of test cases for that Sprint.

All the metrics mentioned above, could be measured Sprint-wise which will help in identifying the improvement steps for the next Sprint.

In the Continuous Measurement phase, these measurement results are put into graphs which helps in identifying the change of measures of each Sprint[14].

In the **Continuous Improvement** Phase, the metrics from the Continuous Integration Phase are analyzed to decide on the improvement steps.

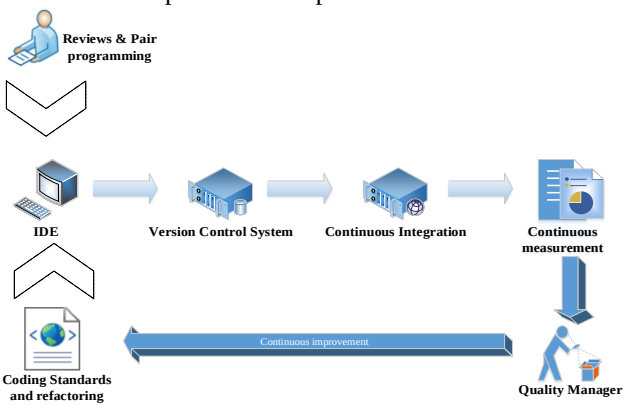

Fig 8. Continuous Improvement

New thresholds could be defined for certain metrics or could add new metrics to be measured in the next sprint or new tools could be introduced.

The GQM (Goal-Question-Metric) Approach is used for determining whether new metrics, tools or new thresholds should be introduced which will help and also to deduce necessary improvement steps. The improvement process does not only lead to selective improvements but to automated quality gates to preserve the improvements enduringly. [1]

IV. Conclusion

Agile Software Development Methodology is being widely used in many software organizations since it focuses on assessing the project progress throughout the development lifecycle and also helps in delivering the right product to the customer. Agile metrics focuses on planning or understanding the project progress and customer satisfaction whereas Traditional Metrics focuses more on productivity and Software Quality. Metrics that were used in the Traditional models could be used along with Agile Metrics, which would be very helpful in determining the improvement steps that could be implemented in the upcoming Sprints. Using this 3C Approach, the project could significantly improve the quality of the deliverable. The metrics collected in the Continuous Measurement Phase will help in identifying the needed improvement activities as part of the Continuous Improvement activity via the GQM Approach.

REFERENCES

[1] André Janus, Andreas Schmietendorf, Reiner Dumke, Jens Jäger, " The 3C Approach for Agile Quality Assurance", 2012 IEEE

[2] Duka D, "Adoption of Agile Methodology in Software Development", 2013 IEEE

[3] Murphy, B., Bird, C., Zimmermann, T., Williams, L., Nagappan, N., Begel, A., "Have Agile Techniques been the Silver Bullet for Software Development at Microsoft", 2013 IEEE

[4] Khalane, T., Tanner, M., "Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations", 2013 IEEE

[5] Tomohiro Hayata, Jianchao Han, "A Hybrid Model for IT Project with Scrum", 2011 IEEE

[6] Mirnalini k, Venkata R Raya, "A software Development Approach for Quality Software", 2010 IEEE

[7] K. Beck, et aI., Manifesto for Agile Software Development, 2001.www.agilemanifesto.org/

[8] Ken Schwaber , "Agile Project Management with Scrum"

[9] Ken Schwaber and Mike Beedle, "Agile Software Development with Scrum"

[10] H. Landim, A. Albuquerque and T. Macedo, "Procedures and conditions that influence on the efficiency of some agile practices", 2010 IEEE

[11] J. Cho and R. Huff, MANAGEMENT GUIDELINES FOR SCRUM AGILE SOFTWARE DEVELOPMENT PROCESS, Issues in Information Systems, Volume XII, No. 1, pp. 213-223, 2011

[12] Jeffrey A. Livermore, "Factors that impact implementing an Agile Software Development Methodology", IEEE 2007

[13] A. Ahmed, S. Ahmad, Dr. N. Ehsan, E. Mirza, S.Z. Sarwar, "Agile Software Development:Impact on Productivity and Quality", IEEE 2010

[14] Jiangping Wan, Weiping Luo, Xiaoyao Wan, "Case Study on Critical Success Factors of Agile Software Process Improvement", pp. 628-631, IEEE 2011

[15] Vladan Devedzic and Sasa R. Milenkovic, "Teaching Agile Software Development: A Case Study", IEEE transactions on education, May 2011

[16] Soundararajan S., Arthur J.D., Balci O., "A Methodology for Assessing Agile Software Development Methods", Agile Conference(AGILE), 2012