Article can be accessed online at http://www.publishingindia.com

Comparison of 3C Approach in Scrum & Extreme Programming

Jisha Johns*, B.R. Prathap, Akhil P Sivan*****

Abstract

Agile software development is one of the widely used software development methodologies today. It is used in many software organisations because of its lightweight methods and its focus on customer satisfaction. It is a group of software development methodologies which includes many frameworks like Scrum, Extreme Programming, Leanetc. Scrum is one of the agile development methodologies which is often used for project management, to manage software and product development. The Scrum methodology significantly increases productivity and also reduces the cost and time compared to water fall model. It also helps software organisations to incorporate the changing requirements, and develop a product that meets the customer requirements. Continuous Integration is the only quality assuring technique which is used in Agile as of today. This process includes the integration of new source code developed into the base code including compile, build and execution of tests. Extreme Programming (XP) is also an agile software development methodology intended to improve the quality of the software developed based on the evolving customer requirements. The 3C approach adds two more activities - Continuous Measurement and Continuous Improvement, to Continuous Integration for Agile Quality Assurance. This would also help in improving the Software Quality and also assures the Software Quality for future development. In this paper, we are comparing the 3C approach that is being introduced in SCRUM with the 3C approach that is already implemented in Extreme Programming.

Keyword: Agile, Scrum, Extreme Programming, 3C Approach

JEL Codes: D4, D21, E5, E42, 44.

Introduction

One of the software development methodologies that was used in many of the software development projects was the Waterfall model. It is a heavyweight model that required extensive documentation. With this model, a perfect understanding of the requirements from the start of the project is needed for developing the product, which even the stakeholders often would not know what is needed and would not be possible to articulate their requirements.

Agile is a group of software development methodologies which includes frameworks like Scrum, Extreme Programming, Lean etc. Agile methods emphasize on the iterative mechanism for developing a software rather than gathering all the requirements at the beginning of the project and then developing the product as in the Waterfall model. The advantage of agile is the ability to revisit the different phases of development, which helps in improving the project efficiency.

Scrum is one of the Agile software development methodologies for managing the software or product

* Faculty of Engineering, Christ University, Bangalore, Karnataka, India. E-mail: jisha.johns@cse.christuniversity.in
** Faculty of Engineering, Christ University, Bangalore, Karnataka, India. E-mail: boppuru.prathap@christuniversity.in
*** Faculty of Engineering, Christ University, Bangalore, Karnataka, India. E-mail: akhil.sivan@cse.christuniversity.in

development. A key principle of Scrum is its empirical approach, ie. accepting the fact that there will be change in requirements, change in technology etc. Scrum teams are self-organising teams, which has a defined goal for a sprint and focuses on achieving it.

Extreme Programming (XP) is one of the Agile software development methodologies which includes several iterations that ends with user acceptance testing. The product will be released only when the customers are satisfied with the features. This methodology is intended for delivering high-quality software more productively. XP emphasizes on some of the basic activities like coding, testing, customer feedback etc.

The 3C Approach

The only quality assuring technique in Agile is the Continuous Integration. Continuous Integration is the process of integrating the new source code which the developer has developed, into the base code which includes compiling, build and running of tests. It is usually done by various Continuous Integration engines. The developer will commit the code in his Integrated Development Environment (IDE) after implementation. These CI engines will regularly check the version control systems like subversion/SVN, concurrent version system/ CVS etc. for new source code. If new source code is found, the Continuous Integration engine compiles, builds the application and runs the tests. For Java applications, CI engines which are quite often used are the J Unit Framework, Apache Maven etc.

Metrics is a key aspect of every software development process. Software metrics could be used for quality assurance, performance improvement, delivering a high quality product etc. The well-known metrics and measurement approaches from traditional quality assurance are not used in Agile development methodology. Agile software development may benefit, if these metrics and measurement approaches are also used along with the Agile metrics. But, the challenge is to integrate the traditional quality assurance metrics and approaches into Agile processes.

In the 3C approach, along with the Agile Practice-Continuous Integration, a subsequent activity is added to measure the software metrics. For measuring traditional Metrics, in Java, apart from the lines of code, 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. The number of tests that are executed and the test coverage are measured during test execution. The test coverage could be measured using many tools like Cobetura, Sonar etc. With the number of tests and the Lines of Code (LOC), the test-growth ratio could be calculated. And the number of broken builds is also a helpful metric in assuring the quality. The measurement results can then be put into graphs so that the changes of the measures/metrics and software quality over time is available. This will help in deriving the right improvement steps and in ensuring software quality.

Another activity called Continuous Improvement is added along with Continuous Integration and Continuous Measurement which helps in the interpretation of the measures and the planning of improvement tasks for achieving better software quality. In this way, we can integrate the traditional quality assurance into an Agile software development process.

The 3C Approach in XP

The 3C approach in XP has the three activities - Continuous Integration, Continuous Measurement and Continuous Improvement. The only quality assuring activity in any of the Agile software development methodologies is Continuous Integration. This activity is common for all Agile software development methodologies.

In the Continuous Measurement phase, traditional metrics along with the Agile metrics are measured and put into graphs.

Some of the traditional metrics that were taken into account are :

- Violations of the Checkstyle rules with severity as error/warning/info
- Violations of Findbugs rules with priority 1/2/3
- Violations of PMD rules with priority 1/2/3

The above metrics would help in identifying the defects that would cause the build to break.

Some of the Agile metrics that were considered are:

(1) *Number of Tests:* As XP enforces test-driven development, the Number of Tests is a helpful Agile metric. The total number of tests does not have much significance in Agile but it provides the first insight in terms of complexity.

(2) *Test-Coverage:* The Test-Coverage is also a helpful Agile metric and it measures how much of the source code developed is executed during the execution of tests. There are two types of test-coverage namely, Line-Coverage and Branch-Coverage: Line-Coverage measures the lines of code that has been executed as part of the test execution whereas Branch-Coverage measures the code executed based on the number of branches. The Test-Coverage quantifies the degree to which the source code is tested by the test suite. The Test-Coverage should be as high as possible for any projects in the software industry.

Test Coverage = Code covered by the tests/Complete code with $0 \leq$ Test Coverage ≤ 1

(3) *Test-Growth-Ratio:* In the case of maintenance projects, the Test-Coverage does not have that much significance, where there are no tests on the existing Base Code. But, in development projects, it makes more sense to measure the Test-Growth-Ratio, where there is growth of the source code. The number of tests executed should increase when there is an increase in the source code unless source code refactoring has been done. In that scenario, the number of tests executed may decrease since some of the functionalities are removed, which are not needed anymore.

Test-Growth-Ratio = Δ Tests / Δ Source Code with (usually) Δ Source Code ≥ 0 , Δ Tests ≥ 0

4) *Number of Broken Builds:* Continuous Integration Engine integrates the new source code developed by the developer to the base code. If the integration fails it is considered as a 'Broken build'. It is a helpful metric in Agile, because a 'Broken Build' indicates that the code developed failed to meet the requirement.

In the Continuous Improvement phase, the results from the Continuous Integration phase are interpreted, especially to deduct improvement steps. The improvement steps could be planning necessary re-factorings or it could be changes to the coding standards or the coding style guide. Also, new thresholds could be defined for certain metrics or could add new metrics to be measured. The GQM (Goal-Question-Metric) approach is used to define new metrics and to deduce improvement steps.

The 3C Approach in Scrum

The 3C approach in Scrum also has the three activities – Continuous Integration, Continuous Measurement and Continuous Improvement. Continuous Integration is the integration of new source code to the base code along with automated compile, build and running tests with the help of Continuous Integration engines like JUNIT, Apache Maven etc. Like in XP, in the Continuous Measurement phase, traditional metrics along with the Agile metrics are measured and put into graphs and it could be done sprint-wise. In addition to the above metrics which are described in 'The 3C Approach in Scrum', some of the traditional metrics that were taken into account are:

- Review Effectiveness
- Defect Removal Efficiency
- Test Metrics

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

(1) Review Effectiveness: The Review Effectiveness is a helpful metric which is used in traditional methods. It measures how effective the review methods are in finding the defects before the Integration Testing of the deliverable.

Review Efectiveness = Number of defects found before testing(before releasing)/Total number of defects

(2) Defect Removal Efficiency: The Defect Removal Efficiency is also a helpful metric which is used in the traditional methodologies and it measures how effective the defect removal methods are in removing the defects before the release of the deliverable to the customer.

Defect Removal Efficiency = Number of defects found internally (before the releasing the product)/ Total Number of defects

(3) Test Metrics is a mechanism to quantify the efficiency of Software testing process. Test Metrics reports number of test cases prepared, number of test cases executed, number of test cases passed, number of defects found etc. Test Metrics include:

- Pass Rate: The Pass Rate helps to measure the number of test cases that are successfully executed during the testing process.

  Pass Rate = Number of Test Cases passed/Total Number of Test cases

- Fail Rate: The Fail Rate helps to measure the number of test cases that are failed in the test execution.

  Fail Rate = Number of Test Cases failed/Total Number of Test cases

- Test Effectiveness: Test Effectiveness helps to ensure quality by identifying the defects during the software testing of the deliverable. It also helps in identifying how effective the test cases are in finding the defects during the testing process.

Test Effectiveness: Number of test cases that resulted in defects/Total number of test cases.

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

Some of the additional Agile metrics that could be considered are:

(1) *Sprint Burn-down:* Sprint Burn-down is an Agile metric that helps in tracking the effort that has been burnt in each sprint. It also helps in planning and tracking using the burn-down tool. Since this chart provides daily feedback on the effort and the schedule, traditional metrics like Effort overrun and Schedule overrun could be tracked early in the sprint and mitigations can be planned early rather than waiting till the end. It also helps in tracking the progress of the project on a daily basis.

(2) *Sprint Velocity:* Sprint Velocity is a powerful method for measuring the improvement which helps in identifying how many story points worth of work could be completed in each sprint with the given available team.

Sprint Velocity = Total Effort completed/Total Story points

In the Continuous Improvement Phase, the metrics from the Continuous Integration Phase are analyzed to decide on the improvement steps. New thresholds could be defined for certain metrics or could add new metrics to be measured in the next sprint.

Conclusion

Traditional metrics could be used along with Agile metrics in the Agile software development which would be very helpful in deducting the improvement steps. 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

Beck, K. (1999). *Extreme Programming Explained: Embrace Change* (1st ed.) Addison-Wesley Professional.

Hayata, T. & Han, J. (2011). *A Hybrid Model for IT Project with Scrum*. International Conference on Information Technology: New Generations.

Janus, A., Schmietendorf, A., Dumke, R. & Jäger, J. (2012). *The 3C Approach for Agile Quality Assurance*. 3rd International Workshop on Emerging Trends in Software Metrics.

Juric. R. (2000). *Extreme Programming and its Development Practices*. 22nd International Conference on Information Technology lnterfaces /TI 2000 (June pp. 13-16). Pula, Croatia.

Wellington, C. A. (2005). *Managing a Project Course using Extreme Programming*. In 35th ASEE/IEEE Frontiers in Education Conference (October, pp. 19-22) Indianapolis.

Appendix

Figure 1: Continuous Integration

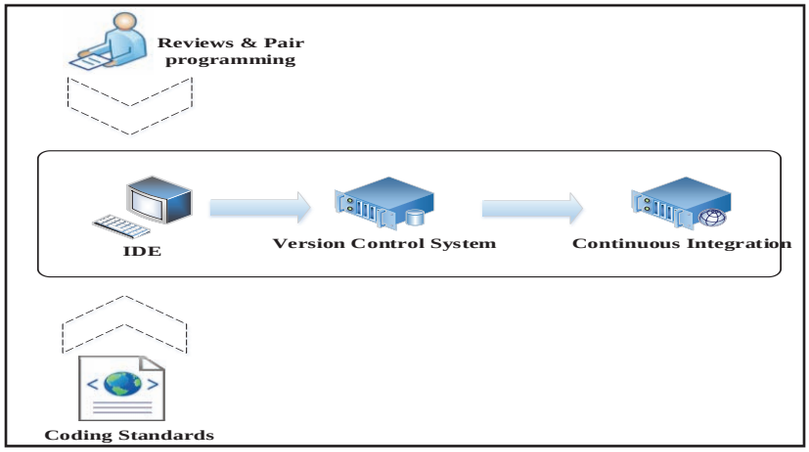

Figure 2: Continuous Measurement

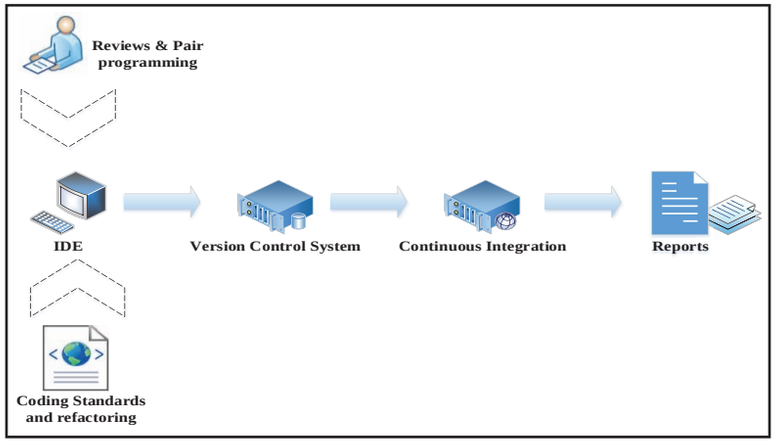

Figure 3: Continuous Improvement