Establishing an Effective Peer Review Strategy in the Software Development Lifecycle

Sayali Kumbhar

DOI: 10.29322/IJSRP.10.06.2020.p102103
http://dx.doi.org/10.29322/IJSRP.10.06.2020.p102103

Abstract- The objective of software delivery is to ensure that the software meets the requirements of the customer with a quality that conforms to the customer and domain benchmarks. The quality of the software can be jeopardized by factors like gaps in requirements specifications, bad design, ineffective coding practices, inexperienced programmers, insufficient test coverage, ineffective reviews etc.
All these factors are important to ensure the quality of the software and we have at our disposal a variety of manual and automated mechanisms for addressing most of these quality checks. However, a crucial factor in the quality assurance process is the manual peer review of artefacts which benefits from the the expertise and experience of the reviewer. This paper documents the approach to define an effective peer review strategy for the software development lifecycle. For ease of understanding, the term peer review and review are used interchangeably in this paper. The purpose of peer review is to identify the defects in the work product/ artefact before it is used as a baseline for software development.

Index Terms- Software Review, Review Efficiency, Peer Review

I. Introduction

While we spend thousands of man-hours and dollars on performing quality assurance and control activities during the lifecycle of a software, it is important to remember that Lean Six Sigma teaches us that this is a non-value-added activity for the customer. Which means that the customer is paying us to develop a product and review is an activity that we perform to find our own mistakes, hence adding no value to the customer. There is a lot of debate on this topic but going by the absolute definition of value-added activities in Six Sigma, this point still holds true.

So, to better understand the importance and effectiveness of a review, let us start with the basic questions. *What is the purpose of a review and then how do we know if it is done effectively?*
We are going to use the 5W1H approach of Six Sigma which is a method of asking questions about a process or problem that we are trying to solve. The 5 W (Why, What, Who, Where, When) and 1H (How) helps us understand a problem/ process in its entirety and makes sure that we do not miss any critical aspect of the problem/ process.[1]

Starting with the first W of **Why**. The purpose of a review is to identify if the artefact meets the objective for which the artefact is created. For example, does a Test Plan cover test cases for all the requirements documented. The reviewer shall review from the perspective of identifying any missing inputs in the test plan, coverage of the requirements, domain specific test cases etc.
In the second **W** of **What**, we shall identify the content to be reviewed within each artefacts of the software development lifecycle. Let us consider the example of an SRS (Software Requirements Specification) document.
An SRS document is created for serving as a basis for all design and development activities of a software, throughout its lifecycle. It elicits the needs of the customer, functionality, constraints, the objective that the software will achieve and the exclusions i.e. the functionality that the software will not perform. This documentation is then referred and updated throughout the lifecycle of the project and hence is extremely important for the future of the project. [2]
Essential contents of an SRS should address these broad topics

- An accurate description of the scope of work to be completed
- Clear, easy-to-manage details for software designers and developers
- Use case scenarios for the testing team
- Alignment of customer requirements to features
- Updatable single source of truth for software development
- Includes input from a variety of stakeholders
- Clearly states scope, inclusions and exclusions
- Not a one-time document but has regular updates based on discussion with the customer/s

Once this critical content of the artefact is identified, the Project Manager should facilitate creation of checklists and guidelines which ensure that the reviewer will not miss any critical checkpoint. This exercise should be repeated for every critical artefact of the project. The project team can then refer to the relevant material for self-review and peer review and meet the required quality standards.

The next W we look at is **Who.**
The reviewer needs to have the relevant experience of the subject. Simply having a senior person perform the review is not an effective review approach. Referring to the earlier example, an SRS is a perfect example of where a manual and expert review is critical. The reviewer needs to bring in his/ her subject matter expertise and experience in the domain of the software and the software development process. An experienced reviewer can question and identify the lacunas in any/ all of the key content identified for the artefact.

Reviewers should also ensure:

- Be clear, concise and direct in their findings
- Insists on self-review evidence before peer review
- Spend reasonable time on the review. For example, studies suggest 60-90 minutes for up to 400 lines of code [3]
- Report findings as specific as possible.
    - Incorrect finding: "Rename variables to more appropriate name"
    - Correct finding: "Rename variable "var1" in file "xyz.c" to "TimeInSeconds"
- Verification of the review finding fixes by the reviewers

So far, we have seen Why, What and Who of the Review Process.

The next **W** is for **Where**. In the interest of efficiency of the overall review process, the Project Manager should identify a platform for the review process which helps effectively plan, track and measure the review process. Most ALM platforms now provide a mechanism to plan the review as part of the project schedule, track the review report as well as generate review metrics that we speak about later in this paper. Review report can include information like reviewer, reviewee, review checklist, time spent on review and rework, review findings and tracking to closure after the reviewer verifies them etc.

Once the mechanism to plan and track reviews is identified, the last **W** to examine is **When.** This is a very important question because if reviews are not done at the right point of time in the project, then the efforts spent on the review and QA activity will increase exponentially. [] It is therefore important that the Project Manager plans the reviews for each phase and artefact in the project plan at the right time. It is ideal to plan for review of artefacts as soon as they are created since this ensures that it is fresh in the mind of the reviewee and time taken to fix the review comments is much lesser. For artefacts that get developed over a longer period of time like the SRS, it is helpful to plan multiple intermediate reviews rather than waiting to finish the entire document.

Now to move to the **How** part.

As we spoke about earlier, review is not a value-added activity to the customer. Which means that the customer is paying to get the end-product delivered and any time we spend on review is the time we are spending to fix our own mistakes. Considering this, reviews should be made as efficient as possible.

- Automate all the reviews that can be automated. There are many technology specific review tools available in the market
- Self-review is crucial before sending it for peer review
- Encourage team members to make a self-checklist to catch the typical mistakes that the individual knows they make
- Utilize tools/ platforms as much as possible to execute the review process (e.g. JIRA/ Github/ PTC etc.) to optimize the planning, tracking and recording of reviews.

Creating a Review Strategy

Since we now understand the 5W and 1H behind the effectiveness of review, we should use this understanding to formulate a review strategy for a software project. The first step towards this is to identify the phases of the project and all the critical artefacts in each phase.

Considering the example of a classic waterfall development cycle, illustrated below are the key artefacts.

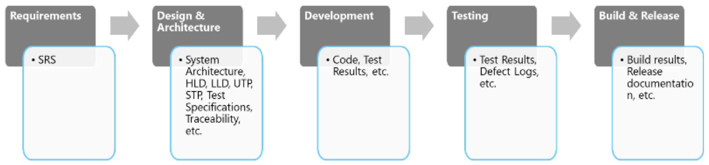
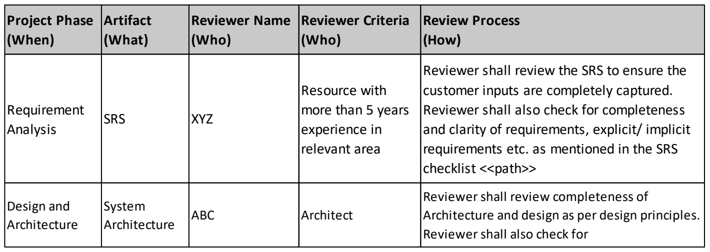

Requirements
• SRS
Design & Architecture
• System Architecture, HLD, LLD, UTP, STP, Test Specifications, Traceability, etc.
Development
• Code, Test Results, etc.
Testing
• Test Results, Defect Logs, etc.
Build & Release
• Build results, Release documentatio n, etc.

Figure 1: Waterfall SDLC and the outputs of each phase

Each project can choose the granularity that they want to achieve in selecting the artefacts depending on the schedule and budget constraints, team skills, project complexity, customer's project management approach etc. Higher the granularity, better the quality but also more efforts and cost on the project. For example, a Project Manager might choose the highest granularity where every single artefact in the project is reviewed by a peer. This will require the Project Manager to identify as many qualified resources, plan for the review time in the project budget and track each of these reviews for completion. This strategy will help in ensuring highest quality but will also create a significant burden on the schedule and effort of the project. A more moderate approach to this would be to select the key outputs of each phase of the project as illustrated above and plan for thorough reviews of these artefacts.

Once the key artefacts are selected from each process, the review strategy needs to be defined. As illustrated below, the strategy covers the When, What, Who and How part of the 5W1H.



Project Phase (When)	Artifact (What)	Reviewer Name (Who)	Reviewer Criteria (Who)	Review Process (How)
Requirement Analysis	SRS	XYZ	Resource with more than 5 years experience in relevant area	Reviewer shall review the SRS to ensure the customer inputs are completely captured. Reviewer shall also check for completeness and clarity of requirements, explicit/ implicit requirements etc. as mentioned in the SRS checklist <<path>>
Design and Architecture	System Architecture	ABC	Architect	Reviewer shall review completeness of Architecture and design as per design principles. Reviewer shall also check for

Figure 2: Review strategy documentation sample

Project Manager's viewpoint

Project Managers need to foster the right culture within the team with respect to reviews. Without this support, the review process cannot be implemented and sustained in the right spirit. An environment of trust and learning is essential in the project.

Do's

- Project Managers should promote reviews as a mechanism to learn, grow and communicate better within the teams

- Treat defects positively and not to point fingers at individuals
- Closely track the defect trend. It is ok to have more defects when the review process is in its initial days or team is still in a forming stage [4]
- Promote usage of tools for review (Lint/PMD/Checkstyle etc.) and review process (JIRA/Github etc.)

Don'ts

- Don't pit the team against one another when it comes to finding defects
- Defect data should not be used in performance reviews. It will deter the reviewee and reviewer from reporting defects

Measurement

As Peter Drucker says, "You can't manage what you don't measure". But before starting to measure, it is important to identify the outcomes that you expect.
In an ideal world, everything is flawless and there are no defects. But since that is rarely the case, a Project Manager would want to catch all the defects internally before the product is given to the customer. Considering this, the below listed metrics are most relevant to measure the effectiveness of the review.

- Inspection Time = Number of hours spent in review
- Defect count = Number of defects identified
- Defect rate = defect count/ inspection hours
- Defect density = defect count/ size measure (eg: LOC/ Use cases)

It is difficult to suggest a high/ low threshold for these metrics. The thresholds of each of the metrics will differ based on the type of project, the team competency, and the stage in which the project is. Let us discuss this with two examples below.

Example A: Consider an AUTOSAR stack that has substantial legacy code and is going to undergo continuous enhancement. A developer makes changes to a few lines of code in one of the libraries and sends the changes for review. The reviewer understands the impact of any change to such a huge legacy code and carefully traces the code changes and impacts to various files and reports 8 defects after spending 2 hours. (Inspection Time = 2 hours; Defect count = 8; Defect rate = 8/2 = 4)

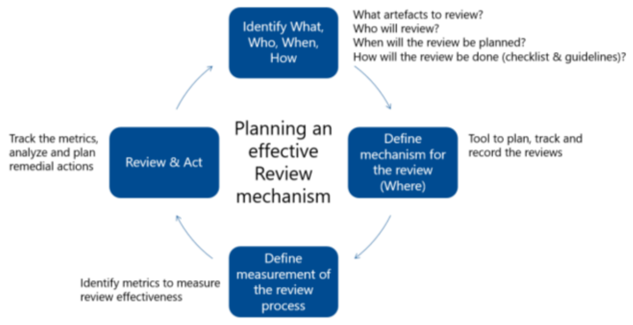
Example B: Developer adds a validation message to an input box and sends the code for review. The reviewer goes through the changes, checks for completeness of the validations and reports in 2 defects after spending 30 minutes. (Inspection Time = 0.5 hours; Defect count 2; Defect rate = 2/0.5 = 4)

In both these examples the defect rates are the same, but it is difficult to say which review was more effective. Hence it is difficult to put a generalized ideal threshold value for these metrics. Instead threshold can be defined by the Project Manager at the beginning of the project based on the team's skill and expertise and then refined as the project matures.

The Project Manager should also analyze the metrics to understand the trends and the relevance to the phase of the project. For example, higher internal defects at the beginning of the project is acceptable as the team is usually still in a forming stage, they are new to the product and the processes. A gradual decline in this number is also important as the team and the project matures. This decline indicates that the team is learning from the review process. The review findings should also be analyzed to look at the type of findings and repeated findings. Analysis of the type of findings gives an insight to the Project Manager on the areas of improvement within the processes and skill sets. Repetition in the findings will indicate that the learnings from the reviews are not sufficiently assimilated by the team and hence similar mistakes are being repeated. This indicates an organizational issue and can be addressed by training the team, refining the guidelines, and strengthening the self-review process.

Conclusion

This paper proposes a clear path to defining an effective review mechanism. An effective review mechanism is an important step in identifying defects at an earlier stage in the development lifecycle, reducing the overall cost of fixing these defects [8] and ensuring the delivery of a good quality work product.


Identify What, Who, When, How
What artefacts to review?
Who will review?
When will the review be planned?
How will the review be done (checklist & guidelines)?
Planning an effective Review mechanism
Track the metrics, analyze and plan remedial actions
Review & Act
Define mechanism for the review (Where)
Tool to plan, track and record the reviews
Identify metrics to measure review effectiveness
Define measurement of the review process
Figure 3: Summary: Planning an effective Review mechanism

REFERENCES

[1] https://www.isixsigma.com/implementation/basics/using-five-ws-and-one-h-approach-six-sigma/

[2] https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document

[3] https://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/

[4] https://www.mindtools.com/pages/article/newLDR_86.htm#:~:text=Team%20formation%20usually%20follows%20easily,adjourning%22%20or%20%22mourning.%22

[5] https://www.cio.com/article/2431557/running-an-effective-code-review.html

[6] Best kept secrets of peer code review by Jason Cohen, Steven Teleki, Eric Brown

[7] Applying Peer Reviews in Software Engineering Education: An Experiment and Lessons Learned by Vahid Garousi

[8] https://www.researchgate.net/publication/255965523_Integrating_Software_Assurance_into_the_Software_Development_Life_Cycle_SDLC

AUTHOR

Sayali Kumbhar
B.E. Computers, MBA (Operations), Six Sigma Black Belt
sayali.kumbhar@gmail.com