

Using Peer Review Data to Manage Software Defects

By Steven H. Lett

Abstract: Peer reviews, in particular software inspections, have become accepted within the software industry as a cost effective way of removing defects as early as possible in the software development cycle.[1] The peer review process is also quite easy to measure. However, relatively few software project managers take an active interest in the data from the peer reviews taking place on their project and miss a golden opportunity to identify potential product defect issues and take corrective action. This article describes one way in which peer review data can be utilized for managing software defects in much the same way cost and schedule data are used to manage project budget and schedule commitments.

Introduction

At a software process conference several years ago, the statement was made by a presenter that software engineering is the only engineering discipline that relies on testing for quality. Even if that is somewhat overstated, it stills rings true to a certain extent. In most software organizations, it isn't until integration testing occurs that it becomes apparent to management if a software product is or is not laden with defects. If there are an excessive amount of defects, it may be too late to prevent the project from finishing unsuccessfully. The time it will take during testing to find all the defects, and then debug, fix and retest them is very often unpredictable and can delay product shipping. [1] Pressure to complete the testing on time may lead to a product with significant undiscovered defects being delivered into operational use by a soon-to-be unhappy customer. The more the quantity of defects can be reduced prior to testing the more predictable testing will be and the fewer the quantity of defects in the delivered software.

Of course organizations can try and ensure successful project performance and acceptable product quality a number of ways, e.g., by hiring the best people, having a quality process, and ensuring employees are well trained. However, just as having all those prerequisites for success in place doesn't mean project managers don't need to manage their cost and schedule commitments, it also doesn't mean that managers don't need to proactively manage product defects.

Software project managers are normally taught how to manage project cost and schedule commitments through the measuring of actual work progress and cost against a schedule and a spending plan. Corrective actions are taken to address cost or schedule problems as soon as they are detected throughout the development cycle. Yet, a similar, proactive, consistent, measurement-based approach for managing product defects is usually nonexistent. Software development managers tend to keep their fingers crossed and wait for the testing results to see where they stand regarding defects in the product being developed.

So, the question is, can defects be managed using measurement data so that problems can be identified and dealt with prior to the start of testing? One way this can be done is by establishing a consistent and effective peer review process and then using the peer review data to assess the work product defect status and take corrective actions as needed to address defect-related issues.

Peer Reviews and Peer Review Measurements

A peer review process should be in place in any software development organization as a significant defect removal activity. Peer reviews are included in the Software Engineering Institute's Capability Maturity Model Integration (CMMI[®]) [3] as a required process for those organizations following the CMMI as a guide for process improvement. Peer reviews are especially valuable as a way to remove defects early in the development cycle and should be performed on all major software development work products including requirements, design, code, and test procedures. The software inspection method of performing peer reviews is generally considered the most effective method of performing peer reviews [2] and is an indisputable software engineering best practice. Other types of peer reviews that are practiced with varying degrees of formality are team reviews, walkthroughs, and pair programming. [1] Once peer reviews are an established practice, the data from each peer review can be used for defect management. For this purpose the following data from each peer review are recommended to be collected:

- Program, function, and work product identifiers
- Type and phase of review, e.g., design walkthrough or code inspection
- Who attended and how much time was spent preparing for the review meeting
- How long the review meeting(s) lasted and who attended the meeting
- Size of the work product, e.g., pages of design or source lines of code (SLOC)
- Total major defects and total minor defects detected

For each defect found the following data is recommended:

- Defect type, e.g., missing, wrong, or extra.
- Defect origin, i.e., the development phase where the defect originated, e.g., requirements, design or code. Note that it is possible that a defect can be discovered in a later phase than when it was first inserted, e.g., a design defect can be discovered during a peer review of code.
- Defect severity, i.e., major or minor. A major defect being any defect that could be discovered or observed by a customer during operational use of the software, or a defect that requires significant time to fix. Minor defects are everything else, e.g., documentation errors.
- Defect location, e.g., module or program element name

From the collected data the following derived measurements are recommended for each peer review:

- (Major) Defects per Detection Hour – the average number of major defects found for each hour of detection time derived by dividing the number of major defects found by the sum of the participants' total preparation time and total time (labor hours) spent in the review meeting
- Average Preparation Rate - the average rate at which each reviewer prepared, e.g., pages per hour, which indicates how quickly the material was reviewed
- Meeting Rate, e.g., pages per hour
- (Major) Defect Detection Rate – the ratio of the number of defects found per the size of the work product, e.g., defects per page or defects per 1000 SLOC (KSLOC)

After a sufficient amount of peer review data has been collected by similar projects and data integrity has been established, average rates can be established for the

preparation, meeting, and defect detection rates (for each type of peer review for each type of work product). This is usually done by the organization’s measurement guru or analyst. From these averages high and low detection rate thresholds can be established to trigger further analysis of the data and possible action. An advanced method is to apply statistical process control (SPC) [4] and calculate the normal range of performance for each of these rates. However, in an organization that isn’t ready for advanced methods, even applying somewhat arbitrary threshold limits, e.g., plus and minus 30% of the average, can be valuable in getting managers and technical leaders to begin to pay attention to the data and begin to make measurement-based decisions. Also, the managers and technical leaders should view the thresholds only as reference points and learn to scrutinize the data regardless if thresholds are exceeded. As historical performance data are accumulated, the thresholds should be revisited as needed.

Analysis Leading to Action

The key to managing software defects is to gauge if an excessive number of defects are being inserted into the software work products and/or if the defect removal activities of peer reviews and testing are not being executed effectively, and taking immediate corrective action to address issues in either area.

After each peer review an easy-to-read summary report from the peer review should be generated and made available for the project manager and/or technical lead to analyze. Besides the work product identification data and a link to the full peer review record, it should contain the calculated rates described above and the threshold limits that go with them, as illustrated in figure 1.

Figure 1 – Peer Review Summary Report Excerpt Example

Measurement Name	Value	Ave.	Thresholds		Over Threshold
			Upper	Lower	
# Participants	4.0				
# SLOC	500				
# Pages					
Meeting Length (Hours)	2.5				
Total Prep Time (Labor Hours)	3.0				
Total Mtg. Time (Labor Hours)	10.0				
Total Detection Effort (Labor Hours)	13.0				
Average Prep Time per Participant	0.8				
Average Prep Time Review Rate - SLOC/HR	666.7	300	500	100	Yes
Average Prep Time Review Rate - Pgs./HR					
Average Defects Found/Detection Effort Hr.	0.5				
Meeting Review Rate - SLOC/HR	200.0	300	500	100	
Meeting Review Rate - Pgs./HR					
Average Defects Found per Page					
Average Defects Found per KSLOC	14.0	8	12	2	Yes

Team Performance

The first step in analyzing the data of a peer review immediately after it is performed is to assess the performance of the review team. For example:

- Was the composition of the review team adequate, i.e., were the review participants capable of performing an effective review of the work product based upon their experience, their expertise relative to the work package being reviewed, and their proven ability to contribute effectively in peer reviews? [2]
- Was there sufficient preparation and meeting time? This can be ascertained by evaluating the average preparation rate and average meeting rate parameters against their upper and lower thresholds established from historical data.
- Were there any other pertinent factors that should be considered, e.g., whether the work product being assessed was unusually complex or simple, whether the work product was especially large or small, or whether the product contained reused material?

Once the peer review performance is assessed considering the review team composition, preparation time, and meeting time, the defect detection rate should be evaluated to determine if it is unusually low or high. It should be noted that a suspect team performance that yields an average number of defects may indicate an above average number may have been detected with a more normal team performance.

Defect Detection Rate

The key measurement indicator to evaluate next is the defect detection rate, i.e., the ratio of defects found to the work product size. This is an indication of the work product's defect density, though undetected defects may still exist. [2] The defect detection rate should be compared against the average rate and upper and lower thresholds that have been established from historical data. If the rate exceeds the limits, or even if the rate just seems too high or too low considering other pertinent factors, an investigation should be held to determine the root cause of the high or low rate. Once the cause is determined, appropriate corrective action should then be taken as needed.

The analysis, investigation and corrective actions would normally be the responsibility of the project manager, or may be delegated to a technical leader. Also, the peer review team themselves may decide if the work product should be reviewed a second time. However, the project manager, who has overall responsibility for the project, should be actively overseeing the defect management actions being taken.

Investigating Root Cause

The most common potential causes for exceeding the upper threshold of the defect detection rate are:

- A better than average detection effort occurred during the review. This might be indicated if an unusual number of the organization's better defect detectors participated in the review. These are usually very experienced software engineers with an excellent attention to detail and a proven superior ability to detect defects during peer reviews. If this is the case, no further action may be required.
- An excessive number of latent defects from predecessor work products were discovered. This could be verified by reviewing the defect origin of each defect discovered during the review.

- An excessive number of defects were inserted in the current development phase. If the previous two potential causes of exceeding the upper threshold can be eliminated, then it may be concluded that an excessive number of defects originated during the current phase of development.

The most common potential causes for exceeding the lower threshold of the defect detection rate are:

- A worse than average detection effort occurred during the review. This would be known after performing the peer review performance assessment described earlier.
- A lower than average number of defects were inserted this phase. If the peer review performance was judged to be adequate, then most likely the author or authors of the work product did not insert as many defects as normal. This may be readily evident by knowing the author's ability, experience and past performance. If this is the case, no further action may be necessary.

Further investigation may be required to determine the absolute root cause of an abnormally high or low number of defects detected:

- For an excessive number of latent defects in predecessor work products, determine the source of the latent defects, e.g., ambiguous requirements or a poor design. Looking at the defect type categories for the defects found in the review, if a significant number indicate missing or extra processing, it usually indicates poor requirements or design.
- For an excessive number of defects inserted in the product that was reviewed, determine why, e.g., was the author not properly prepared for the assignment, or was the author under schedule pressure?
- Did a faulty software process contribute in any way to a high number of defects or a poor performance by the peer review team?

Taking Action

Once the most likely cause of a defect detection rate threshold being exceeded has been identified, a corrective action plan should be formulated that addresses the root cause of the problem. Possible actions for high or low defect detection rates:

- Require additional or alternate peer reviews, if the peer review team hasn't already done so. This is one of the most important ways a project manager can influence product quality. If an excessive number of defects were found in the peer review, after the work product has been updated with defect fixes, it is often a good idea to have the updated version undergo a second review because defect fixes often introduce new defects. Also, if the performance of the first peer review was questionable, a second review with different personnel may be advisable.
- Judiciously make personnel adjustments. This may apply to the peer review team for a second review or it may apply to the need to change the author of the work product.
- Identify and address training or coaching needs. This may be required to solve personnel performance issues or software process problems.
- Make process or tool changes to remove process inefficiencies.

Success Factors

As with any process, the usual factors for success apply to defect management, such as, management support [1] [2] at all levels, defined and consistently applied software management and engineering practices, effective training, and an established measurement program.

Summary

Project managers can utilize the peer review data to gain insight throughout the development cycle into the defect density of software work products that have a direct bearing on the quality of the final software product. By looking at key peer review measurements an assessment can be made both in how well each peer review was performed and into the defect density of each work product relative to historical averages.

From this data analysis potential problems can be identified and corrective actions taken to address them. By doing this project managers can directly influence the quality of the final work product, reduce the number of latent defects entering the final stages of testing and increase the project's chances of completing successfully.

References

[1] Karl E. Wiegers, *Peer Reviews in Software*, Addison-Wesley, 2002.

[2] Ronald A. Radice, *High Quality Low Cost Software Inspections*, Paradoxicon Publishing, 2002.

[3] Mary Beth Chrissis, et al., *CMMI[®] for Development*, Version 1.2, Addison Wesley, 2007.

[4] *Measuring the Software Process*, Addison Wesley, 1999.

About the Author

Steven Lett, a senior consultant and affiliate of the David Consulting Group, has over thirty years of software engineering experience as an engineer, project manager, and process improvement leader consultant. He has played significant leadership roles in helping organizations achieve Software Engineering Institute (SEI) Capability Maturity Model (CMM[®] and CMMI[®]) Maturity Levels 2, 3, 4 and 5. His experience spans both DoD and IT software development.