Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1996 Proceedings

Americas Conference on Information Systems (AMCIS)

8-16-1996

Using the Software Peer Review Process to Obtain Measurable Pay Back

Alan N. Pryor
University of North Texas, runr@msg.ti.com

Eugene G. McGuire
The American University, mcguire@american.edu

Using the Software Peer Review Process to Obtain Measurable Pay Back

Alan N. Pryor
Business Computer Information Systems Department
University of North Texas
Denton, Texas
runr@msg.ti.com
Eugene G. McGuire
Department of Computer Science & Information Systems
The American University
Washington, D.C.
mcguire@american.edu

Introduction

This paper presents an overview of the Software Engineering Institute's Level-3 Key Process Area (KPA) - Peer Reviews. A recent case study of a large U.S.-based fortune 100 company provides suggestions on a tailored (dependent on size and scope of the project) approach to locate software defects early in the product life cycle and provide causal analysis of the process (necessary for improving the process). In addition, a pay back scheme is also presented which shows the return on investment (ROI) of using the peer review process. The pay back on performing peer reviews has been, to-date, subjective and lacking in quantitative financial measures.

Capability Maturity Model

The Software Engineering Institute (SEI) has worked with industry and government to develop a software model and software process maturity framework. As a result the Capability Maturity Model (CMM) was developed to guide and improve the maturity of software processes for the software developing and supporting communities. The CMM consists of five levels of progression. Each level has a direct focus and sub-components known as Key Process Areas (KPAs) (Paulk, et al., 1993).

Every corporation utilizing the CMM is challenged with the transition from one level to the next. Many of the companies that have chosen to adopt this framework are currently focused on the transition from Level-2 to Level-3. At Level-3 (Defined Level), the standard process for developing and maintaining software across the organization is documented, including both software engineering and management processes, and these processes are integrated into a coherent whole. Processes established at Level-3 are used (and changed, as appropriate) to help the software managers and technical staff perform more effectively. The software process capability of Level-3 organizations can be summarized as standard and consistent because both software engineering and management activities are stable and repeatable (Paulk, et al., 1993). The peer review process is a KPA in the SEI Level-3 CMM which has recently received a lot of attention due to its implementation challenges and subjective pay back measures.

Peer Reviews

Peer reviews are a tool for removing defects from the software product early and efficiently. They involve a methodical examination of the software product by the producers' peers to identify defects and areas where changes are needed. The specific products that will undergo a peer review are identified in the project's

defined software process and scheduled as part of the software project planning activities. The CMM defines the goals of peer reviews to be (1) planning review activities, and (2) identifying and removing software defects early in the process (Paulk, et al., 1993).

The CMM requires that the project follow a written organizational policy for performing peer reviews. Additionally, adequate resources and funding must be provided for performing peer reviews on each software product to be reviewed. Leaders receive training in how to lead peer reviews. Reviewers who participate receive required training in the objectives, principles, and methods of peer reviews. Peer reviews are planned, and the plans are performed according to a documented procedure. Data on the conduct and results are recorded. Measurements are made and used to determine the status of the peer review activities. The software quality assurance group reviews and/or audits the activities and work products for peer reviews and reports the results (Paulk et al., 1993).

A recent case study, conducted by the software process improvement organization of a large U.S.-based fortune 100 company, showed several processes are in use to accomplish peer reviews. Peer reviews are often costly and timely and as a result are considered by the project to not be worth the effort. In addition, they are slow to evolve with a company's software process and thus are reactive instead of proactive.

Types of Peer Reviews

In an attempt to meet the SEI Level-3 peer review requirement, there are several types of peer reviews in use, each with the intent of locating software defects. Some involve significant investment from the project teams; others are a simple inspection of someone else's code. The key to obtaining pay back on a peer review is to select the appropriate technique, depending on the magnitude of the project. In the past, the technique used to meet the peer review requirement has been the traditional "code walk-through," where the programmer walks into a room with his/her peers and defends his/her process and data flow creation. This technique, though better than none, has not yielded ideal results for many reasons. The formality of such meetings is costly and generally involves players that need not attend, and omits players that should. Additionally SEI Level-3 requires that the process for conducting peer reviews be defined, not an ad hoc event, with random attendees. An adequate peer review should include technologists of expertise from outside the organization, and exclude those who have no value to add or gain.

Below are five of the major peer review techniques in use identified from the study. Note that not only does a defined process exist for conducting peer reviews, but the process allows for five implementation techniques dependent on project scope and impact. The scope in software development varies as drastically as does that of constructing a building. A two ton crane is not needed to lift a two-by-four; however, many companies are doing just that in regard to the implementation of peer reviews. The five methods proposed are inspections, structured walkthroughs, hybrid inspection/walkthroughs, desk-checking, and round-robin reviews. Note as these techniques are discussed that the amount of effort (cost) to implement each one varies significantly, as it should in that the effort (cost) of each software development project varies significantly.

Inspections are a rigorous static analysis technique that requires training of the inspection team, well-defined roles that include a facilitator and scribe, and the complete measurement of defects encountered. Inspections cover software products including requirements, design, code, user documentation, and even test plans and test cases (Capers, 1994). Inspection advantages and disadvantages are: it is the highest measured efficiency of any known form of defect removal, the only technique to achieve efficiencies of 75% or more in field conditions (Capers, 1994), and it may be tailored for the project. But it is also the most costly and time consuming.

Structured walkthroughs are a static analysis technique in which a designer or programmer leads the members of the development team and other interested parties through documentation or code, and the participants ask questions and make comments about possible errors, violation of development standards, and other problems. Defects should be identified and logged during the review with action items assigned.

Upon conclusion of the review, a determination of how to proceed is made. Outside of the review, the identified defects are resolved with their status being tracked and communicated (STEP, 1995). Structured walkthroughs advantages and disadvantages are: it is the second most efficient method for removing defects, it may be tailored to the project. But it is not as costly as full inspection and usually requires a moderate size group of reviewers.

The hybrid inspection/walkthrough technique is a modified approach in which a group of participants, consisting of author, moderator, reviewers (minimum of two) and scribe, perform the review. Combinations of roles are allowable, i.e., moderator/scribe, and/or author/reviewer. Reviews are structured so that the anticipated benefit exceeds the minimum necessary support required of the individuals who are asked to participate. Software review metrics are collected and monitored so as to determine the effectiveness of the reviews (DIVP, 1995). Hybrid inspection/walkthroughs advantages and disadvantages are: it requires a smaller group of reviewers, the roles may be combined, it concentrates on finding defects; however it could lose effectiveness if too much analysis is done on defect evaluation.

Desk-checking is a private review and debugging carried out by individual programmers and analysts who were not involved in the software product creation. Defects are identified and logged during the review. Defect resolution, status tracking and communication take place after the review (Capers, 1994). Desk-checking advantages and disadvantages are: it is the least expensive, it is easy to schedule and complete; however it is typically the least effective review method and effectiveness depends largely on the reviewer.

The round-robin review technique is a process of desk-checking by multiple peers in a sequential manner. The initial checker makes his/her review, identifies and logs defects, then passes the folder to the next reviewer who performs the review adding and logging any additional defects. This continues until all the reviewers have participated and the folder is returned to the author. Round-robin review advantages and disadvantages are: it is more efficient than simple desk-checking, multiple reviewers are involved, roles can be assigned, typically a lower cost than other review techniques; however it is not as efficient as inspections.

Peer Review Process

Having the previously listed variety of techniques available is an improvement, but how does one know which technique to use in which situation? The company under study documented the following steps as its process for tailoring and implementing the peer review process in order to locate and remove software defects early in the process and meet the SEI Level-3 KPA peer review requirement. The peer review process should be tailored to do what makes sense by project. To do this:

1. Identify which software work products are to be reviewed (usually includes more than code).
2. Select the appropriate peer review technique by weighing the software work product's scope and business impact with the amount of time appropriate to spend on a peer review(s). (For example, mission critical requirements specifications or applications require more stringent peer review techniques).
3. Document the peer review plan, schedule, and types. When scheduling a peer review, allow adequate time for planning, preparation, meeting, rework, and follow-up.
4. Publish the peer review schedule during the project planning stage.

Peer Review Return On Investment (ROI)

As with every business investment, the question of pay back and cost benefit continues to gain attention in regard to peer reviews. Many software managers and programmers view peer reviews as an overhead function that is unnecessary and has a negative impact on system delivery cycle time. The following formula is a first pass at measuring the benefit of investment of performing peer reviews. The variables are

straightforward and emphasize the down-stream impact when defects are not found early in the requirements or design phases. The ROI may be calculated as the total cost to the project if defects were not identified, less the total cost of those defects that evaded the peer review, divided by the actual cost of doing the peer review. Stated in an algorithm it is:

ROI = [(Total cost without Peer Review) minus (Total cost with Peer Review)] divided by (Actual cost of the Peer Review)

A spreadsheet will calculate ROI based on the above algorithm using the raw data gathered during the peer review process (contact first author for a hard or soft copy of an ROI spreadsheet). This allows for the monitoring of the peer review process at the organizational level, and provides input to causal analysis. Causal analysis will be performed to identify areas where the process needs improving. The minimum set of metrics that a project team must collect from each peer review are: total time spent on peer review (planning, preparation, logging, review meetings), defects found (number, severity, criticality); and origination stage (requirements, design, code, test, integration, and installation). This formula may have room for improvement, as do most attempts at objectivity; however, it is a good start and quite a simple technique to document defects which are located within the "traditional" systems analysis and design phases. It is recommended that software shops of all types (whether an SEI fan or not) give this a try and monitor the overall system performance relative to previous implementations not following this discussed procedure and process.

Peer reviews are necessary in order to identify defects early in the software product life cycle. Peer reviews can be costly and time consuming and as a result are often not properly done or are sometimes omitted completely. However, if the appropriate peer review technique is selected, the investment will result in measurable pay back. Depending on the scope and size of the software product under development, the appropriate technique to perform peer reviews should be selected in order to locate defects in a cost effective manner.

References

Capers Jones, *Assessment and Control of Software Risks*, Englewood Cliffs, N.J.: Yourdon Press. 1994.

Digital Imaging Venture Projects (DIVP), *Software Review Version 1.0*, Dallas Texas, Texas Instruments, September 29, 1995.

Motorola Inc. Presentation given to Texas Instruments, Dallas, Texas. 1995.

Paulk, Mark C., Bill Curtis, Mary Beth Chrissis, Charles V. Weber, Capability Maturity Model for Software, Version 1.1, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. February, 1993.

STEP Group, *Glossary of Software Engineering Terms, Version 1*.2, Dallas, Texas: Texas Instruments. 1995.