SQA Technique for Low Cost Software Project (LCSQA)

Farooq Bashir  
Department Of Computer Science  
University OF Agriculture Faisalabad, Pakistan  
farooqbashir1@hotmail.com

Tasleem Mustafa  
Department Of Computer Science  
University OF Agriculture Faisalabad, Pakistan  
tasleem_mustafa@uaf.edu.pk

Abstract— LCSQA means Low Cost Software Quality Assurance .Which introduce a technique that ‘how to Quality’ the low cost software projects. The explosive growth of the software industry in recent years has focused attention on the problems along associated with software development, uncontrollable costs, missed schedules, and unpredictable quality. To remain competitive, software firms must deliver high quality products on time line within budget. However, to bring their products more quickly to market, software managers may avoid quality improvement processes such as design reviews and code inspections, believing that these processes only add time to the development cycle. Certainly the economics of improving quality are not well understood in the software development world.

1. INTRODUCTION

The LCSQA is a Technique used to reduce the cost of the software when the software has been build. LCSQA provides technology solutions and services for business. And focus on traditional technologies to provide rapid and cost effective solutions for business problems. What are the essential features your organization will need right now, and what do you anticipate needing in the future. Draw up a "wish list" and prioritize the features you're looking for, marking those that are critical and cannot be compromised. A clear view of what your organization requires the software to accomplish both now and in the longer term. That will help to narrow down the numbers for a detailed side-by-side comparison of features.

There is some confusion about the business value of quality even outside the software development context. On the one hand, there are those who believe that it is economical to maximize quality. Their key argument is that as the voluntary costs of defect prevention are increased, the involuntary costs of rework decrease by much more than the increase in prevention costs. The net result is lower total costs, and thus quality is free. On the other hand, there are those who believe it is uneconomical to have high levels of quality and assume they must sacrifice quality to achieve other objectives such as reduced development cycles.

The objective of this research is to study and analyze the new technique of the SQA for the low cost software projects. The purpose of research is to introduce some new methods which are easy to use especially for cost effective software. By applying some effective methods we can reduce the cost of the software when the software has been developed.

Low Cost Software Quality Assurance (LCSQA). The rationale behind LSQ is that software quality expenditures must be financially justified. Increasingly, the chief financial officers in many companies are promoting disciplines for financial evaluation to encourage investments that yield the greatest response for limited resources. Such disciplines are particularly important in the context of software engineering, as software expenditures account for larger portions of capital budgets. Software quality investment that should provide a financial return relative to the initial and ongoing expenditures in the software quality improvement initiatives. One way to evaluate software quality improvement efforts is to consider them in terms of specific initiatives. Examples of software quality improvement initiatives include implementation of design reviews, testing and debugging tools, code walkthroughs, and quality audits. Initiatives require an initial investment—the software quality investment (SQI)—that includes the initial expenses for training, tools, effort, and materials required to implement the quality initiative. There are also ongoing expenditures for meetings, tool upgrades, and training that are required to maintain the quality process once it is in place. We call this software quality maintenance (SQM).
Finally, each software quality improvement initiative should result in annual revenues. These software quality revenues (SQR) can be derived from the projected increases in sales or estimated cost savings due to the software quality improvement.

2. PROCESS IMPROVEMENT

• Process Improvement #1: Creation of life-cycle development standards and the introduction of computer-aided software engineering (CASE) tools.

• Process Improvement #2: Increasing minimum educational requirements for hiring, integration of LCSQA Software Blueprint methodology with the CASE tools, creation of detailed style guides for documentation, and institutionalization of weekly program management status reviews.

• Process Improvement #3: Seamless integration of the CASE technology with the publications department, addition of schedule and performance metrics, automated development cost estimation, automated software configuration, and Pareto analysis.

• Process Improvement #4: Cycle time analysis and development of an automated support cost estimation methodology.

LCSQA basic objective is to drive failure costs to zero by implementing quality initiatives that would dramatically reduce defect rates. For example, key problem areas were identified that were causing the majority of the defects. An early Pareto analysis suggested that most of the defects were related to job control language (JCL) errors. To analyze the cause and effect of the defects arising from JCL errors, BDM used fishbone analysis.

The causes of the errors were incorrect syntax, parameters, volume serial numbers, and data set names. As a result of this analysis, LCSQA instituted increased emphasis on JCL walkthroughs, mandatory use of automated JCL check software, and team leader approval for data set names and hard coded volume serial numbers. Impact of Quality Initiatives, Were the quality improvement initiatives over the life of the project successful at LCSQA?

Defect Finding

I have calculated the marginal cost effects of defect reduction by estimating parsimonious log linear regressions that related each nonconformance cost type to lines of code and defects. The coefficients from these regressions were used to determine the marginal impact of 10% reductions in defect rates at different levels of initial defect rates. Marginal analysis of nonconformance costs at three different defect density levels indicates that the greatest marginal cost effects of defect reduction were in the Development, Management, Operations, and Quality Assurance areas.

There are a number of reasons for these findings. Development costs were directly impacted by defect reduction due to less rework in debugging, testing, and programming. Management costs experienced a large marginal impact from defect reduction due to the involvement of senior managers when errors occurred and the high cost of their time. Quality Assurance costs were also impacted because fewer defects led to less reinsertion, reappraisal, and re-testing activities. Operations costs are driven by software testing, production, and maintenance support. Defect rates have a high marginal impact on operations costs because they influence allocation of operations staff support for regression testing and maintenance workload. Return on Software Quality. To determine the return on investment of software quality improvement at LCSQA, I have calculated the Net Present Value.

3. IMPACT OF QUALITY INITIATIVES

4. MANAGERIAL IMPLIMANTAIONS

The word "Quality" has many meanings, ranging from luxury and merit to excellence. It is often defined simply as "fitness for purpose". Quality is a multi dimensional concept, different dimensions of quality will be important to different users. Every day we use the word "Quality" as an attribute for a human made product just to compare it with other products in the same category we say it has a high quality or that one has a low quality. We use it, but never pay attention to the fact that this concept is likely unclear.

There are two goals of the software quality system. The first goal is to build quality into the software from the beginning. This means assuring that the problem or need
to be addressed is clearly and accurately stated, and that the requirements for the solution are properly defined, expressed, and understood. Nearly all the elements of the SQS are oriented toward requirements validity and satisfaction.

Now a day software development has taken such a shape that the most visible thing in the whole process is change. The traditional process models were designed to work in linear fashion and it was very difficult to manage rapid changes, while using these traditional process models. In order to manage the rapid pace of change a need was felt to develop such process models which can accommodate this rapid pace of change. To fulfill this need customized process models according to their requirements and resources. But they neglect the quality of the software. Which effect the overall software because they less emphasis on the quality of the software. In Pakistan software develops through customized process models are not quality software because they pays less attention on the quality of the software.

For the small scale projects where customized process model are used there must be use the exercise to maintain the minimum quality of the software through requirement gathering tool, coding standards, reviews and testing. The requirements should be gathers through requirement gather tool. In every organization every employee has some set of roles and responsibilities to fulfill its job. For example a specific employee has role as cashier and its one responsibility is to collect cash from the customer/client. In this software record every employee’s role and unit responsibility in the organization and what he wants from the software system to fulfill its responsibilities. Coding standards are also strictly follows. Reviews are the first and primary form of quality control activity. Quality control is concerned with the search for faults or defects in the various products of software development. To check that weather standards are implemented are not. Customized Quality Model is a draft version for testing and will request feedback on it so that amendment can be made if necessary. Furthermore I think that there is lot of chances of improvement in testing phase. It is not write on a stone, it can be improved. In testing more work is require to refine this model.

Standard Implementation

The implication is to avoid making large investments in software quality toward the end of a project. I did not explore this at LCSQA, but it may be profitable to invest early in software quality improvement initiatives that have synergies with future initiatives or that make future improvements possible. For example, investing in a software metrics program at time t may enable use of other quality techniques like Pareto analysis and Statistical Process Control at time t + 1. Such investments create an option for future software quality improvements, and their value can be assessed using option pricing analysis.

5. Changes Based on the Results

The intent of our analysis has been to emphasize that software quality improvement should be viewed as an investment. It is possible to spend too much on software quality. Thus, it is important that companies financially justify each software quality improvement effort. Finally, we have seen that it is important to monitor software conformance and nonconformance costs so that conformance policies can be adjusted to reduce the total costs.

6. Conclusions

A number of interesting observations emerged from my analysis. I have found that defect density improved at LCSQA with each software quality initiative, but at a decreasing rate. This could reflect LCSQA strategy for quality improvement, which was to focus on eliminating the major problems first. An implication of this finding is that much of the effect of quality improvement may be realized from the initial quality improvement efforts. My analysis of LCSQA software quality costs reveals that conformance costs per line of code were relatively fixed over the life of the project, which could have resulted in part from LCSQA contractual obligations to keep certain processes in place. I have speculated that conformance costs may be difficult to change in general, as they may involve relatively fixed components, counter to the claims. It could also be, as that companies must be diligent in reevaluating and adjusting their appraisal and preventive efforts so that they do not over invest in conformance activities as quality improves. Further studies of software conformance and nonconformance costs to clarify this issue would be helpful. Finally, I have found that the largest marginal returns in terms of nonconformance cost reduction at LCSQA were for the Development, Management, Quality Assurance, and Operations cost centers. This result suggests that it may
be most cost-effective to implement software quality improvement initiatives that are specifically directed at reducing effort in these areas.

I found the larger returns from quality improvement occur early in the project (both LCSQA and SQA are highest then) and the rest of the project can benefit from these improvements.

REFERENCES