



The Value of Quality

The Value of Quality: A real-world case study

A White Paper

by

Lonnie D. Franks and Jeff Van Fleet

December 2005

“Process Improvement should be done to help the business – not for its own sake.” – W. Edwards Deming

“In God We Trust – All others bring data.” – W. Edwards Deming



W. Edwards Deming

1 Introduction

Managing a large application development effort is a complex challenge. Knowing how much quality assurance and quality control to apply to a program are critical to maximizing the likelihood of success. Apply too little quality, and undiscovered defects cause schedule and cost overruns, and poor user acceptance. Apply too much quality, and the program costs more than it needs to meet schedule objectives. Many managers make decisions which ignore certain quantifiable principles.

1. Productivity decreases as system size increases. This productivity can and should be both measured and benchmarked. The productivity measures should include cost per function point, schedule based on size in function points and defects per function point.
2. Productivity decreases as organizational size and complexity increases. This means that smaller, more competent teams with simplified organizations can do highly productive work. Organizations with higher process maturity levels naturally tend to have more competent staff.
3. The use of good processes in specific areas greatly increases productivity. Small investments in process improvements can cause dramatic improvements in productivity.
4. All work products contain defects and these defects are best removed as soon as possible. The costs of defects are as follows:

Cost	When Discovered
\$250	if discovered at Unit Test (i.e., in-phase)
\$1,250	if discovered at Integration Test
\$6,000 - \$30,000	if discovered in Production

5. An average project has close to 45% rework costs, although these are not normally budgeted. In large projects, the rework costs can easily exceed the original budget. These rework costs can be reduced to approximately 10% with a reasonable investment in quality assurance and quality control.
6. Predicting the number of defects expected during specific life-cycle phases acts as a “forcing function” for the development and testing organizations. It drives behavior because now, the teams have specific goals for finding and fixing defects. For example, they are no longer just building one test case per requirement; they are building and executing test cases to find defects. They can build proactive quality control and test plans knowing how many defects they expect to find. Defect prediction and monitoring defect density is of maximum benefit when applied to all work products.
7. Although cost and schedule variance are critical to monitoring a project’s performance, much of the cost of program overruns is due to poor quality (too many defects discovered too late in a program). Defect prediction is the basis for Quality Variance™. It is a proactive measure of risk that, if calculated, monitored, and regularly reported, provides **early visibility** into program cost and schedule overruns.

2 Lighthouse Quality (V&V) Methodology

The following diagram illustrated Lighthouse’s methodology for reducing program risk by applying best quality (V&V) practices. We initially scope the system using function point analysis (FPA) of the requirements, use cases, etc. We measure and monitor defect density of the work products as they are developed (requirements, design, test cases, etc.). We make recommendations for improvement when the defect density or the defect leakage rate is too high. When code is available, we run it through our code analysis tool that calculates McCabe cyclomatic complexity and measures actual lines of code for virtually every front-end and

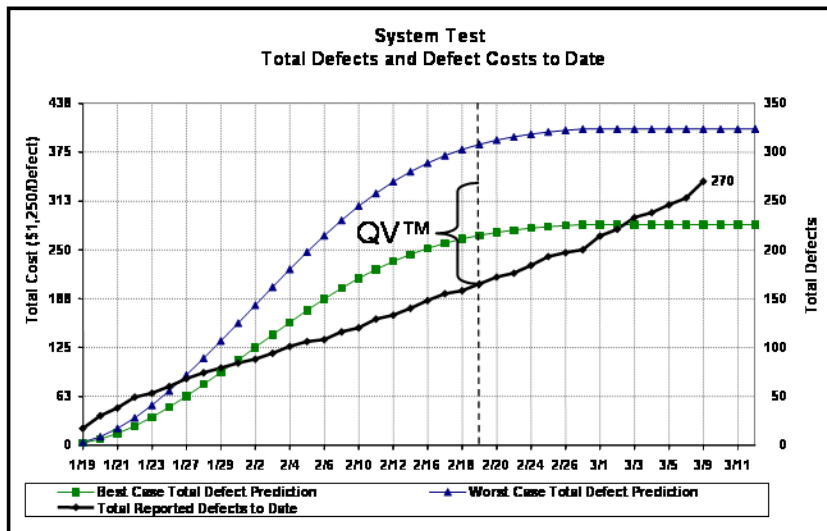
database language, including Java, C#, VB, JavaScript, C++, T-SQL, PL-SQL, etc. We use this actual line of code count to tighten up the defect predictions and to identify the highest risk modules.

We gather time tracking, defect tracking, and test case data from the development team and feed that into our quality monitoring and reporting system. We use that system to generate plan versus actual charts to control the exit criteria from each phase of the software life-cycle. We analyze this information, make recommendations, and report risk to program cost and schedule.

The following graph is an example of one of the Lighthouse quality monitoring charts showing a predicted band of expected defects (the green line is the best case and the blue line is the worst case) along with the actual discovered defects (the black line). This is data from a real customer's System Test.

**Lighthouse Technologies
Defect Prediction Tools**

- Used in Each Lifecycle Stage as Exit Criteria
- Quality Drives:
 - 99% Defect Free in Production
 - Defect Containment
 - Risk Strategy
 - Test Cases
 - Staffing
 - Cost/Schedule



$$\text{Quality Variance (QV)} = \text{Quality Plan (Qp)} - \text{Quality Actual (Qa)}$$

$$\text{Quality Performance Index (QPI)} = \text{Quality Plan (Qp)} / \text{Quality Actual (Qa)}$$

It is worth noting that the customer scheduled the System Test for 30 days, but as they approached the 30-day mark, they realized that their Quality Variance™ was approximately 100 defects (see the right axis) which equated to approximately \$125,000 (see the left axis). It took an additional 3 weeks of testing to find the expected number of defects. Had they released on the planned date, their users would have discovered those 100 defects multiple times over. They continue to use the Lighthouse defect prediction model to plan their unit test, integration, system, and pilot test schedule, test team size, and testing budget.

An Example Quality Analysis for System Testing

The following is an actual report delivered by Lighthouse to identify program cost and schedule risks and plan appropriate System Test resources and budget.

Executive Summary

System Testing is critically important to ensuring rapid maturation of software at minimal cost. The analyzed data indicates that a rigorous system testing phase will take 17.8 staff months and discover approximately 42% of the remaining defects. Without good System Testing, it can be expected that the system will either go to production with an unacceptably high defect density or take an additional 3 to 6 months to make ready for deployment.

Introduction

The purpose of this report is to analyze the cost, schedule, and effort implications associated with the System Testing.

Some fundamental assumptions of this analysis include:

- A rigorous System Test will be conducted.
- \$72 per hour developer cost
- \$72 per hour tester cost
- A minimal number of requirements and design defects are still present in the system
- An average number of code defects are still present in the system
- No code inspections have been performed
- The report is based on the data available on 2/14/2005. This code is likely to grow, but this growth has not been accounted for.
 - C# SLOC count was 29,036
 - SQL SLOC count was 3,111 (Solid and T-SQL) from a count by Lighthouse staff on 2/14/2005
- 636 Function Points (based on 52 SLOCs per FP in C# and 40 SLOCs per FP in SQL)
- Code size is fixed (not growing)

This report uses a standard rate of \$72/hour for both developer and tester. This may understate the cost of quality, but the correct rate can be calculated by multiplying any of the costs listed in this report by (actual rate/\$72). Note that if the actual rate is \$72, nothing changes. Note that rework costs include all rework costs, not just the cost for developers to rework the code.

This report uses standard cost ratios from the IEEE for the cost of defects by phase and standard defect injection rates by phase from Capers Jones.

Report Findings

Since assumptions in this report predict a low defect injection rate and a high removal rate to this point of development, the number of predicted defects is probably understated. It is possible, considering that no design or requirements documentation were developed for the system, more latent defects may exist than in industry average systems. Therefore, it is possible that significantly more defects will be detected during System Testing than in an average system.

Overall, it is expected that there are 678 defects remaining in the system. Requirements accounts for approximately 138; design accounts for approximately 125; code accounts for approximately 415.

Of the 678 total defects, approximately 42%, or 288 defects will be detected during the System Test. This breaks down into 41 requirements defects, 57 design defects, and 190 code defects. This is a significant number of defects to be discovered during System Tests.

Generally speaking, on a system this size, it would be expected to have 4 test cases per defect found, 2 hours per test case created, 40 minutes per test case execution, and 4 test executions. This means that per defect found, it will take approximately 10.7 staff hours to develop test cases and perform execution of those tests.

Given that 288 defects will be discovered during System Test, the cost of developing test cases and executing the tests is \$220,631 based on a \$72/hour rate. This will require 17.8 staff months of effort. The generation of the required test cases will take 13.4 staff months and should be initiated early to minimize staffing requirements during the actual System Test period. Execution of the test cases will take 4.4 staff months and

will achieve a peak of 26 defects per day detection rate. Assuming a detection rate of 2 staff hours per defect, the peak staffing required is 13 testers.

With the currently estimated defect density, the total effort of all testing (integration, systems, and acceptance) is 31.8 staff months. This also assumes that thorough System Testing takes place. The defects discovered during all phases of testing will require 76.3 staff months of rework through the initial pilot. Even with the above assumptions, the defect density will be approximately 11.9% of the defects remaining when integration starts, or 81 defects released to production.

Strictly based on the competency of the vendors and Subject Matter Experts, a significant possibility exists that the defects injected will be higher than this report estimated.

Summary

The data in this report has been generated using industry standard data (as integrated into the Lighthouse defect prediction models), and other adjusted inputs specific to the development approach used on this project. All of the inputs used were very conservative. Requirements and design defect injection rates were estimated to be very low, when in reality they may be very high because of lack of formal documentation.

Given that the report is a very conservative view of the project, it is apparent that if rigorous System Tests are not conducted, the cost, schedule, and effort for the project will increase significantly.

About Lighthouse

Lighthouse Software Testing is committed to software quality from the point of project initiation through the software testing life cycle. Whether you are considering offshore software development or looking to outsource software testing for an upcoming project, Lighthouse's expertise in the software development life cycle (SDLC) can help you achieve maximum efficiencies in all phases of the project. We use industry benchmarks and carefully measured project tracking assessments to lower development costs, shorten time-to-market and provide optimal software performance.

To learn how Lighthouse Software Testing can help with your next software project, please contact Jeff Van Fleet at (937) 458-0055 or visit our [contact page](#) to submit an inquiry.