

Are Software Quality Problems Costing Your Company?

Understanding how much you are spending on quality problems is an important first step to improving results

Everyone in the software arena seems so busy, but are workers spending their time on productive activities or spinning their wheels? With the average software project exceeding its budget and overrunning its schedule by more than 200 percent, the IT machine is skidding out of control. No wonder companies kill half of the software projects that run more than a year and don't deploy another 25 percent. To increase software engineering productivity and reduce development costs, IT managers often turn to what they know best—technology and automated tools. Software that automates source code version control, regression testing, and defect tracking are mere bandages. The underlying disease is the epidemic of defects that requires repeated testing, never-ending maintenance releases, and hordes of technical support people. However, increasing quality in software development processes is a more effective approach.

Many organizations assume that higher quality costs money and takes more time. With his 2001 IEEE paper, "Measuring Software Process Improvement," Capers Jones gave hope to software executives around the world. Compared to typical results, in which one half of most software development project budgets and two thirds of the typical development team's time are spent fixing poor quality, Mr. Jones found that software process improvement generates average results that are far from ordinary: an 80 percent reduction in post-release defects and a 65 percent increase in productivity, while reducing project schedules and costs by 20 percent.

Changing Quality from a Cost Center to a Profit Center

Improving quality can do more than reduce costs. Increased quality and productivity offer tremendous business opportunities. The resources no longer used fixing defects can work on developing new and improved products. More marketable products and faster time to market translate into significant competitive advantage and increased profit margins. Customer satisfaction increases, while turnover and training costs decrease. Employee morale often improves due to increased success, opportunities for advancement, and more interesting work. Even in organizations with tight budgets, executives like the 500 percent return on investment offered by the successful software quality initiative.

With improved processes and higher quality, organizations can expand their software development capability, which means they can complete more projects with the same time and resources. These additional projects produce higher revenues and profits.

The value of quality was proven originally in Japanese manufacturing companies. By implementing quality improvement programs, Toyota now produces half the defects per vehicle than the Big 3 U.S. auto makers do, while enjoying an operating profit three times as high. Although Toyota is the fourth largest car seller in the world, its stock market capitalization exceeds that of General Motors, Ford, and Daimler-Chrysler combined. Toyota, along with many other successful companies standardizes business processes so that success is repeatable across the entire organization.

The True Cost of Poor Software Quality

Every software development project comes with a price tag for the basics: development hardware and software, requirements development, design, coding and unit testing, and project management. Some software organizations consider fixing defects and technical support as part of the basic software development package. Dr. Joseph Juran, often called the father of quality, would disagree. Think about the software engineering activities that you could avoid if your "perfect world" development team delivered software with no defects. Well, the perfect world doesn't exist, but leading software engineering organizations are gaining on it.

Dr. Juran developed the concept of the cost of quality—a measure that takes into account all the costs associated with quality, both good and bad. In essence, the cost of quality represents the costs to achieve good quality as well as the costs incurred due to poor quality, shown in Figure 1. By correctly classifying your quality costs, you can determine how much poor quality truly costs your organization.

Internal failure costs are the costs related to software defects that are found before shipping products to customers. Internal failure activities include correcting flaws in requirements and design; re-reviewing requirements and design changes; correcting flaws in purchased software; retesting purchased software corrections; and most importantly, reporting, tracking, and fixing defects in the software being developed. Most organizations fail to recognize that each additional iteration of integration and system testing adds to the cost of internal failure.

External failure costs relate to defects found after a customer receives the software. External failure is appallingly expensive, because the cost of every corrective activity after the first release of software falls into the external failure bucket, as illustrated in Figure 2. Requirements, design, coding, testing, and quality costs for every maintenance release are part of external failure. Paying technical support personnel and refunding money for software returns are also external failure costs. Even contract penalties, lawsuits, lost sales, lost customers, poor reputation in the marketplace, and lower product prices contribute to the cost of external failure.

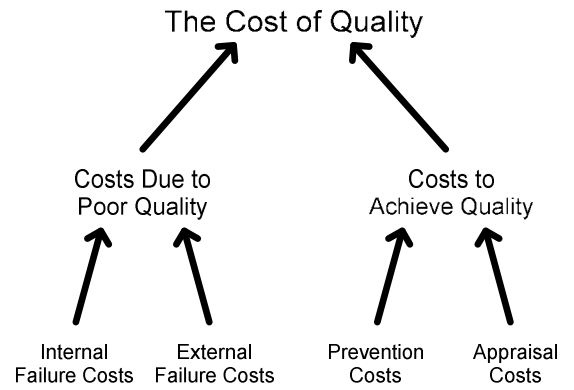


Figure 1: The true cost of poor quality becomes clear when you evaluate all the costs related to both good and bad quality.

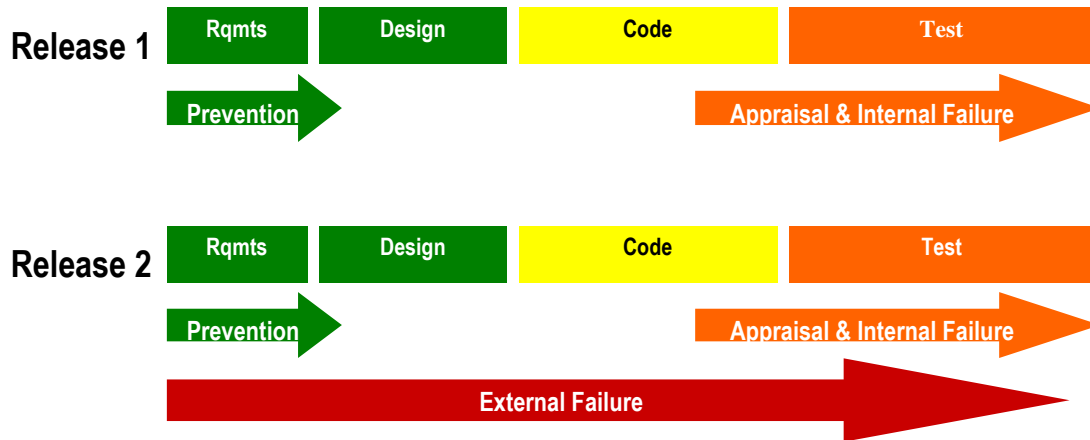


Figure 2: All software development costs incurred after the first release of software are the cost of external failure.

Why High Quality Is Affordable

Compared to the costs of internal and external failure, the costs to achieve good quality in the first place seem quite manageable. In the software world, an ounce of prevention is worth a pound of maintenance. Prevention costs stem from activities that help prevent failure as well as minimize the cost of appraising the conformance to the quality requirements for the software. Prevention activities include quality planning, training team members on quality topics, performing quality assurance, employing software configuration management and tools, managing software reuse, and assessing supplier capability. In addition, performing reviews of requirements, design, and code prevent failure by identifying defects as early in the development process as possible. Assessing internal and external processes and then initiating process improvement also prevents failure in future projects.

Appraisal costs are the costs incurred to determine how well software conforms to its quality requirements. Measuring conformance includes performing the first iteration of integration and system tests, and reporting and tracking the defects found in the first iteration of integration testing. The cost of software to automate testing is also an appraisal cost.

Increasing software quality requires an upfront investment. However, as Figure 3 demonstrates, the result of that expenditure is an overall reduction in the cost of software development. Organizations that spend little or no time or money on preventing defects and appraising quality pay astronomical and ongoing amounts to correct their failures. Over time, this can mean the difference between success and failure in the marketplace. Even small investments in quality improvement can produce exponential decreases in the number of defects. In addition, qual-

ity efforts help your organization identify the defects that do arise earlier in the development process. External failures become less costly internal failures. Defects in code become easier to correct defects in design or requirements.

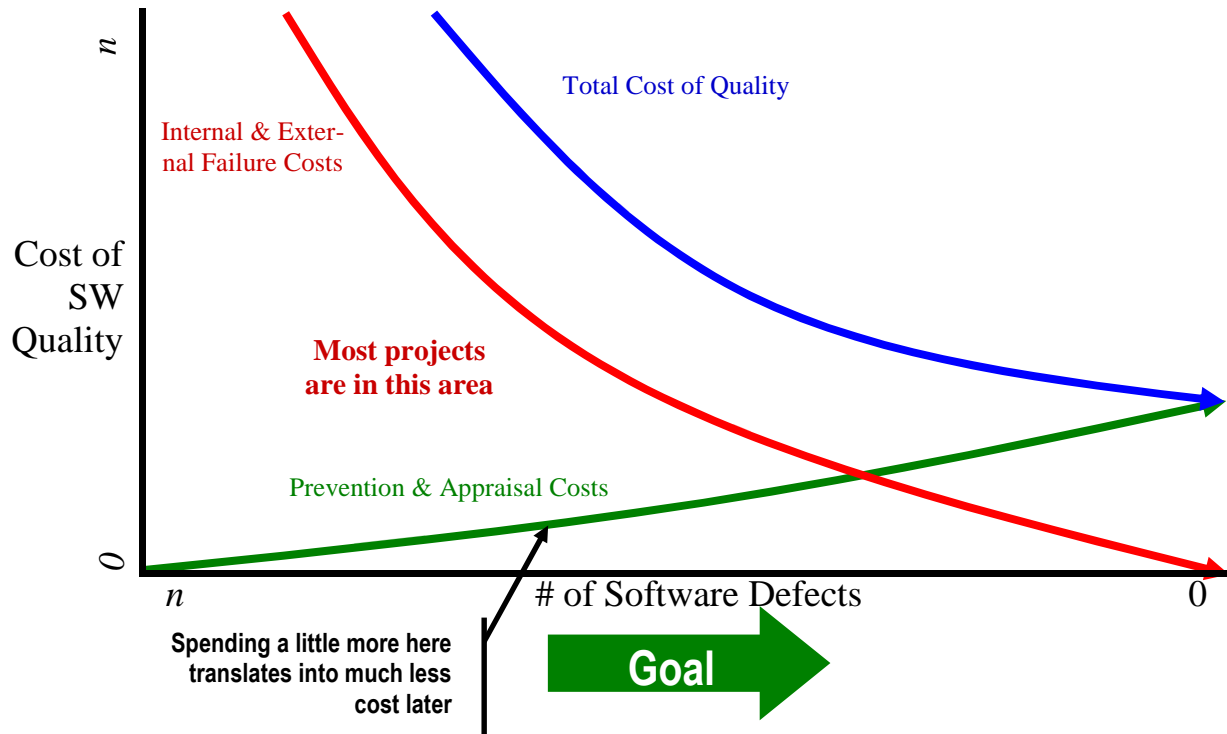


Figure 3: Spending money to increase quality can reduce the overall cost of software development.

Moving Processes to Higher Quality

Without consistent processes, project teams constantly reinvent the wheel, so any quality and success that they achieve is rarely duplicated. As in many industries, the leaders in software engineering often run their operations like franchises. By following standardized quality processes, every employee in the organization can contribute to the business objectives of reducing costs, increasing productivity, and increasing profits. Leading software engineering organizations produce 80 percent fewer defects while delivering twice the productivity of their average counterparts. Instead of fixing problems from the most recent release, these top organizations devote their time and resources to developing new products or enhancing existing ones.

The first step to improving software engineering performance is to identify weaknesses in processes that contribute to poor software quality. Then, by shifting focus to quality within the existing software engineering processes, the development organization can reduce defects and cost, and increase quality and productivity. Table 1 highlights the small but potent differences between average and leading development organizations. The table shows how each type of organization allocates 55 percent of the project resources. For example, the average organization spends 4 percent on requirements, 12 percent on design, and 8 percent on software quality assurance. Leading software organizations spend 5 percent on requirements and 16 percent on design. These slight increases on early phases help hone the goals and objectives of the project, which reduces costly changes in later phases. More importantly, leading organizations spend 19 percent of the project resources on software quality assurance and reviews of deliverables. This attention to detail is immediately offset by a reduction in testing. However, the leading organizations reduce post-release defects, reduce project costs and schedules, and increase development productivity.

The timeline in Figure 4 compares the amount of time that average and leading organizations consume for each phase in a software development project. For example, a leading organization requires only 6.8 months for coding compared to the average organizations 8.8 months. Testing benefits even more with a leading organization using only 3.75 months compared to an average organizations 9.4 months. However, the reduction in rework due to fewer software defects produces the lion's share of productivity gains.

| Allocation of Resources | Average Organization | Leading Organization |
|-----------------------------|----------------------|----------------------|
| Requirements | 4% | 5% |
| Design | 12% | 16% |
| Testing | 31% | 15% |
| SQA | 8% | 19% |
| External Failure Costs | >50% | 15-25% |
| Total Cost of Quality | 50-70% | 40-50% |
| Change in Cost and Schedule | | 20% decrease |
| Change in Released Defects | | 80% decrease |
| Change in Productivity | | 65% increase |

Table 1: Characteristics of Average and Leading Benchmarks

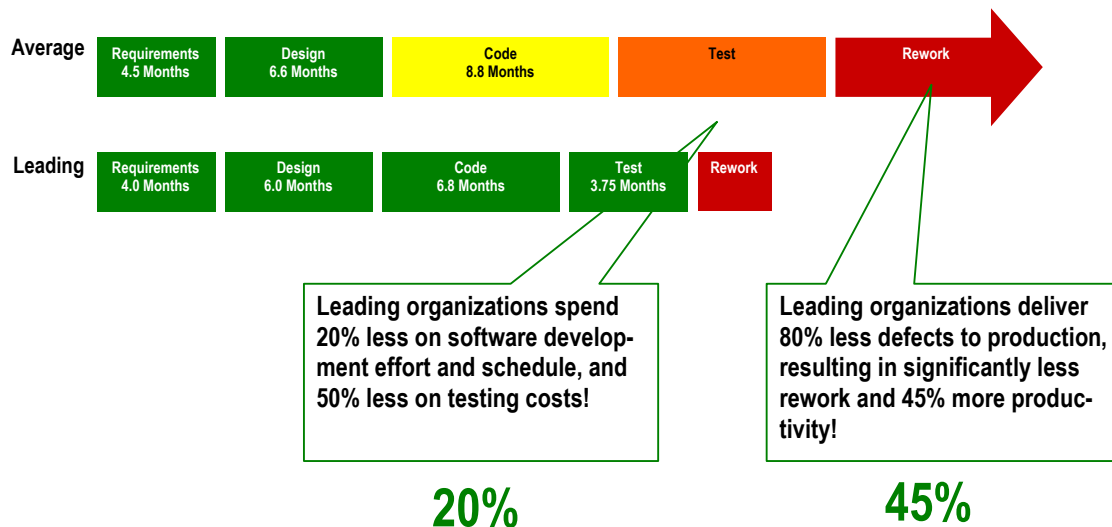


Figure 4: The source of increased productivity

Where to Go From Here

If improving software quality sounds like the solution you've been looking for, here are the next steps you can take to begin your quality improvement initiative. In every improvement program, documenting the current state of affairs is a key step. With a financially-based cost of quality appraisal, you audit the activities performed by your development organization and quantify how much time you spend on each activity.

Understanding how your development organization currently spends money and how much it spends not only provides a yardstick for measuring improvement, but also identifies your organizations' weaknesses. The next step in the quality appraisal is correctly categorizing the costs of your development activities to basic development costs or the appropriate cost of quality categories. Once you visualize how you spend money on quality, you can adjust your allocation to finance high quality activities. If you want guidance in this endeavor, improvement methods such as Six Sigma and CMMI offer structures that support your improvement.

After you have implemented your improvement program, and the changes have become second nature to the organization, you perform a second quality appraisal. This follow-up measures your new performance and determines the return on investment for your program. With the right approach and focus, you too can unlock millions of dollars of productivity in your software organization.