Inspiring, Enabling and Driving Quality Improvement

Jim Sartain
Jim_Sartain@McAfee.com

Abstract

This paper discusses some approaches used at McAfee, Adobe Systems and Intuit to drive continuous significant quality and engineering process improvement through the adoption of engineering best practices including Team Software Process (TSP), Scrum, Peer Reviews, Unit Testing and Static Code Analysis. It covers what has worked well and some of the challenges encountered. Key success factors and an overall methodology are outlined including:

- A multi-year improvement approach that can work in large organizations that may include both eager adopters and resisters of software quality and engineering process improvement.
- Customer satisfaction and defect removal metrics, including what is world class vs. what is typical, and the best practices that can drive major quality and engineering process improvement

Biography

Jim Sartain is a Senior Vice President responsible for World-wide Product Quality at McAfee, the world's largest dedicated security software company. He leads a team responsible for inspiring, driving and enabling continuous quality improvement across McAfee world-wide. Prior to joining McAfee, Jim held quality and engineering process improvement positions at both Adobe Systems and Intuit where he helped drive significant quality improvements in products such as Acrobat, Photoshop and QuickBooks. Jim also worked at Hewlett-Packard for seventeen years in a variety of software engineering and product development roles. His last job at HP was in the role of CTO/CIO for an Airline Reservation Systems business that serviced low-cost airlines including JetBlue and RyanAir. Jim received a bachelor's degree in computer science and psychology from the University of Oregon and a M.S. degree in management of technology from Walden University.

1 The Business Opportunity for Quality Improvement

Many software development projects are destined to be cancelled, significantly delayed or end up being delivered with a level of functionality, reliability or usability that reduces customer satisfaction. Even projects that deliver software may spend a significant portion of development effort on defect driven rework. Some projects spend more effort in finding and fixing defects than feature implementation. In fact, it is not uncommon in the industry to see teams consume more than half their overall development effort on testing and defect repair. Development organizations that make effective use of early defect removal methods including peer reviews and static analysis can reduce their rework costs to less than 10 percent. These world-class teams invest ~10% of their project effort in the early defect removal methods and deliver software on a frequent basis (e.g. monthly) that is at or near release quality.

Another significant driver of software quality costs is customer service and support expenses. Often, a majority of customer service costs are driven by usability, reliability or performance issues. The customer experience can be improved and support costs significantly reduced by identifying and eliminating the root causes for the most common product issues. Post-release issues are another common driver of rework that may consume up to 50% of organizational capacity to deliver post-release fixes.

The direct costs of poor quality (e.g. engineering rework, customer care costs) are usually the tip of the iceberg for quality improvement opportunities. Westinghouse Electric Corporation found that their indirect costs of quality (e.g. reduced sales, less time for innovative work) were three to four times the directly measured costs of poor quality (Campanella 1999).

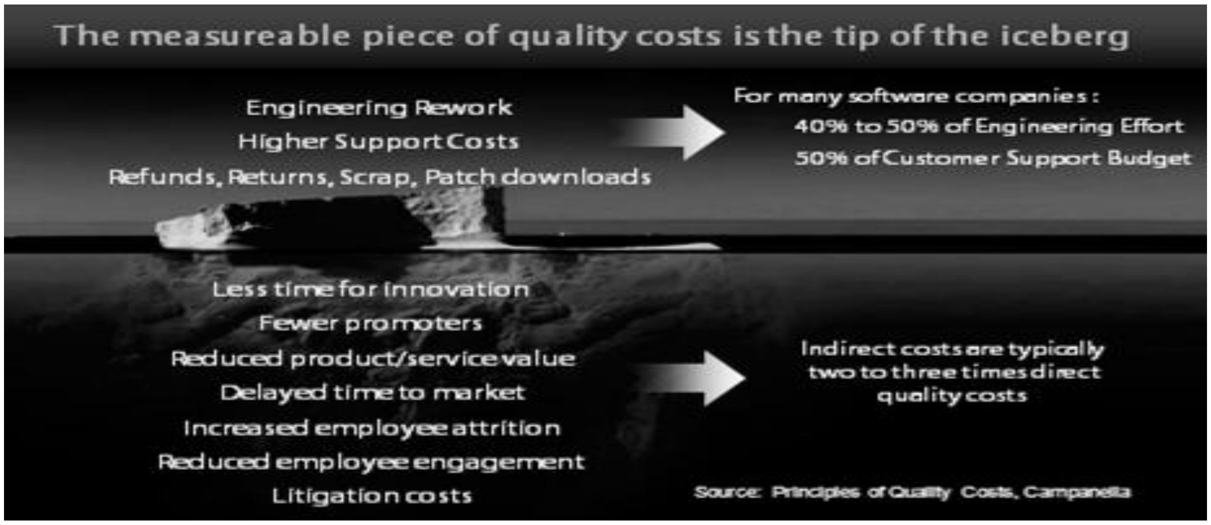
The measureable piece of quality costs is the tip of the iceberg
Engineering Rework
Higher Support Costs
Refunds, Returns, Scrap, Patch downloads
For many software companies:
40% to 50% of Engineering Effort
50% of Customer Support Budget
Less time for innovation
Fewer promoters
Reduced product/service value
Delayed time to market
Increased employee attrition
Reduced employee engagement
Litigation costs
Indirect costs are typically two to three times direct quality costs
Source: Principles of Quality Costs, Campanella

Figure 1: Quality Improvement Opportunities

2 Net Promoter Methodology

Software development teams should have a way of understanding what is most important to customers and how they are doing with delivering great customer experiences. The Net Promoter Score (NPS) methodology is an excellent approach for doing this. The NPS methodology was developed by Satmetrix, Bain & Company and Fred Reichheld and is described in Reichheld's book the *Ultimate Question* (Reichheld 2006). There is more information at http://www.netpromoter.com.

When using NPS customers are asked about their likelihood to recommend a product or service. Delighted customers, called “promoters”, are so satisfied with their overall product experience and relationship with their supplier that they rate their likelihood to recommend as nine or 10 on a zero to 10 scale. Promoters will tell their friends, families and business associates to buy and use the product and they are responsible for generating up to 90% of positive product referrals. Customers rating their likelihood to recommend from zero to six are considered “detractors” and are responsible for up to 90% of negative word-of-mouth. Social media (e.g. twitter, blogs) provides an increasingly visible means for product promoters and detractors to share their views with thousands of customers world-wide.

The NPS methodology includes an all-important follow-on question: “What change would make you more likely to recommend the product?” Product teams use these questions to prioritize development priorities and to continuously improve the customer experience. To calculate NPS you take the percentage of customers that are promoters and subtract the percentage of customers that are detractors. A world-class NPS for commercial software products is 60%. An example of an NPS score of 60% would be having 70% promoters and only 10% detractors. Figure 2 illustrates the NPS calculation.

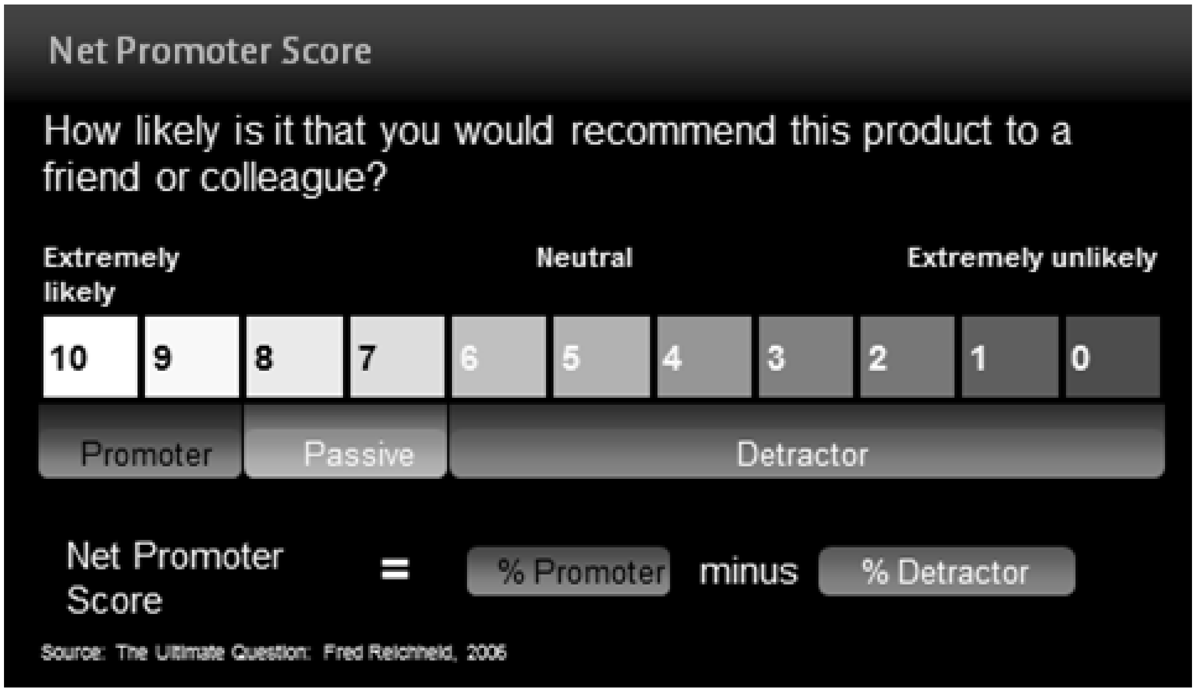

Figure 2: Net Promoter Score Formula

3 Quality Improvement Vision

To drive improvement there must be a vision describing how much better things can be when significant quality improvement are achieved. The vision is a tool to help ensure quality is a priority at all levels of the organization from senior leadership all the way down to front-line employees. The vision should include compelling benefits for key stakeholders of quality including customers, employees and shareholders.

Having an improvement vision that includes continuous improvement is important even for teams that are currently delivering very good quality; otherwise a natural tendency is to accept the current level of customer experience and organizational effectiveness as “good enough.” I’ve found that teams with the best current quality in an organization frequently are not the first to get onboard with mew improvement initiatives. The teams with significant quality issues, especially if they work directly with customers are usually the easiest to get buy in from. Organizations that work directly with customers, are the most likely to understand and feel the customer pain from product issues. Figure 3 outlines the McAfee vision for continuous quality improvement.

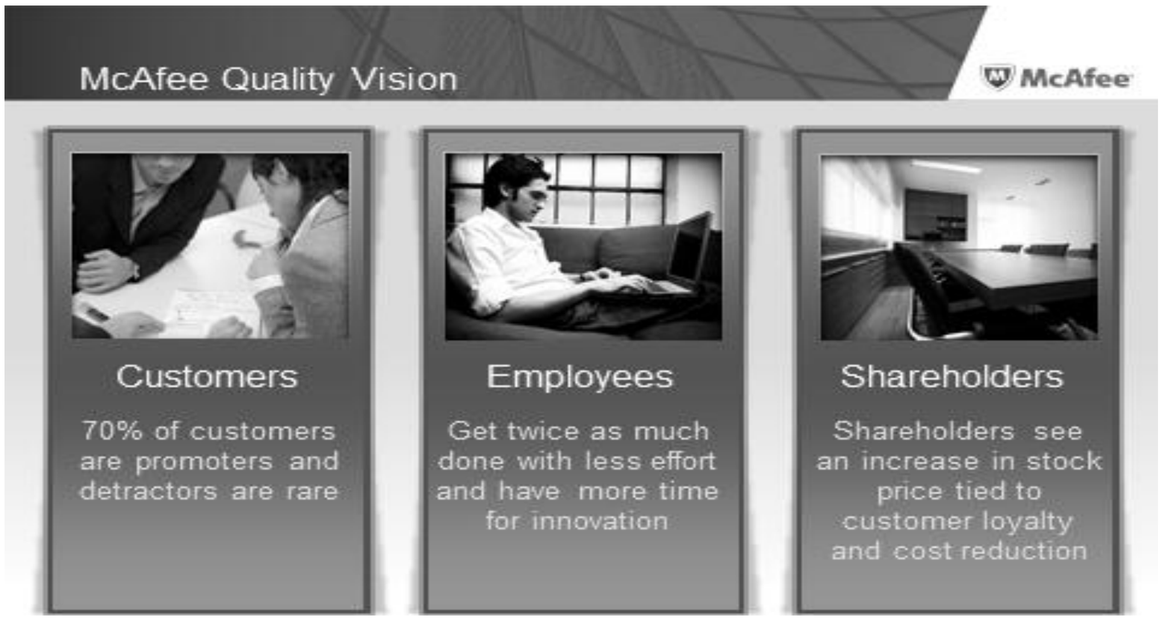
McAfee Quality Vision
McAfee
Customers
70% of customers are promoters and detractors are rare
Employees
Get twice as much done with less effort and have more time for innovation
Shareholders
Shareholders see an increase in stock price tied to customer loyalty and cost reduction

Figure 3: McAfee Quality Improvement Vision

3.1 Customer Benefits

The vision should emphasize directly engaging with customers to understand what is important to them and how you are doing with product quality. Leveraging this information a team can continuously improve to maximize the number of product promoters. In a 2010 study of its customers, Adobe found that that it’s most satisfied customers:

- Were more likely to recommend Adobe products to others
- Invested a greater percentage of their budgets with Adobe
- Were less likely to consider alternative solutions

3.2 Employee Benefits

Employee benefits should include improved work-life balance. Also, employees will benefit from having more time available to do valuable and more innovative work. Providing a less stressful work environment that offers more time to do truly innovative work is a great way to increase employee engagement and ultimately retention. I’ve seen teams at several companies free up 20% more time from reductions in defect-driven rework.

3.3 Shareholder Benefits

Improving product quality can increase the number of product promoters and decrease the number of detractors, which will lead to increased sales, customer retention and the ability to charge premium prices.

Also, productivity improvements from early and efficient removal of defects can free up capacity for work that has a higher return-on-investment. Organizations adopting a quality first paradigm can avoid a vicious cycle where they must spend an increasing amount of their efforts on customer support, software maintenance and bug fixing. Because they are spending an increasing proportion of time on those activities they may not be able to direct adequate resources to innovation. This can ultimately lead to a lack of competitiveness, business decline or even bankruptcy (Humphrey 2001).

Quality improvements will also benefit business efficiency. Customer care budgets can be decreased by reducing the number of customer service calls driven by software defects, usability issues or overly complicated business processes. Other business functions (e.g. legal, PR, engineering) are frequently impacted with the emergence of and aftermath from quality issues that reduce the reliability, usability or performance of products.

4 Required Leadership Support

Leaders are responsible for ensuring the highest priority for their organization is delivering high quality software that meets business goals which will typically include the delivery of quality software by a specified date. This is not a contradiction -- focusing on delivering quality software is the best way to ensure delivery dates are met. A frequent root cause for missing schedule commitments is finding an unexpectedly high number of defects late in the development cycle.

Most software engineers have a strong desire to deliver software of high quality and will if they are provided the right environment: one that includes strong support for quality from all levels of management. To ensure strong leadership support for quality, managers should have quality and engineering process improvement as part of their performance goals. To ensure that all teams aggressively drive to improve quality senior leadership should require all product development leaders to show how every release is measurably better than the last.

I remember a situation from my first Quality Leadership role. The Product Development leaders were asking me why the level of quality was not improving for their products. The simple truth was that it was not an organizational priority for them, and therefore their teams. They expected that the manager responsible for QA or testing and engineering process improvement would somehow be able to drive improvement without any explicit support from them. This was clearly a problem. The solution was that I asked our common boss to ensure that he held each Product Development leader accountable for showing measurable improvement. His initial response was that he was fine with this, but he didn't know how to measure their progress. My response was: "This is not your problem. Any leader that can't assess how their projects are doing with respect to measurable improvement is not doing an adequate job." Soon they were asking for my assistance in helping them identify ways they could measurably assess how they were doing with quality and quality improvement. These leaders now considered it to be *their* priority to improve quality.

4.1 Quality Improvement Objectives, Goals and Metrics

A quality plan including goals and a strategy for achieving them should be established at the start of a project and teams should have compelling quality improvement goals that include specific measurable goals and realistic improvement targets. Teams should also set their own improvement targets as this approach helps ensure ownership and commitment for achieving them. If targets are set tops-down by managers then there is a risk of the teams not buying into them because they don't "own" them. Or in some cases, teams will believe their tops-down targets are impossible to achieve. Any lack of confidence or commitment may not be discovered until it is too late to prevent a schedule slippage and/or release quality issues. The quality goals must be clearly aligned with what is important to the organization.

To help set aggressive but achievable goals, it is useful to have benchmark metrics such as the percentage of defects found and removed using peer reviews. The best benchmarks are from other teams within the organization. The next best source of benchmarks would be from other organizations in the same enterprise. If this isn't possible, then industry data can be used (Jones 2010). Benchmarks closer to the team are not only more likely to be relevant, they are also more likely to be accepted by the team as valid. The collection and sharing of metrics across an organization will help provide relevant benchmarks. These benchmarks can help support a cycle of teams learning about best-in-class performance in the org and then raising the bar when they set their next set of improvement goals.

4.2 Quality Review Meetings

The senior leadership team must regularly review the status of software quality as a team. During these review meetings each team leader summarizes improvement progress and status relative to their quality goals and plans. This provides an opportunity to recognize teams making progress. By publicly recognizing progress, leaders reinforce the importance of improvement. In cases where progress is not being made there should be discussion as to why, as well as the commitment of necessary support to get improvement on track. Senior leader's regularly spending time discussing quality goals and improvement progress will highlight their importance. Also, these meetings serve as an educational opportunity for business leaders to better understand software quality economics. This deeper understanding enables the business leaders to ask the right questions and to be more specific (and genuine) in their praise about improvement progress.

Another benefit of these quality review meetings is that the teams that are make significant progress become role models for the rest of the organization (especially when they are publicly recognized for their progress). Many times I've seen first surprise and then support for improvement goals from other teams when they saw how well the best teams performed. For example, a common goal is to challenge teams to set a goal for the percentage of defects that are found and removed using early defect removal techniques such as peer reviews. When teams see other teams, working under the same constraints, pressures and same code base, finding 50%, 60% or even 80% of their defects early using engineering best practices then they are likely to set their own goals.

5 Engineering Best Practice Rollout Strategy

Most engineering best practices have been around for years and are known to many engineers as best known practices. So why aren't they common practice? One of the major reasons teams hesitate to adopt best practices is that changing how they work involves taking a risk. The conundrum is that teams often don't see the benefit of the "new way" and therefore become convinced unless they try it. The most credible recommender of a new practice is a team that already adopted it and seen practical benefits.

Also, the adoption of new best practices may require support from the entire team. Unless there is strong leadership support for the need to change, one or two individuals on a team can veto or otherwise obstruct improvement. It is a leadership responsibility to challenge teams to support new ways of doing things and reward and recognize teams that take the risks to use them.

The level of motivation for adopting change normally varies from team to team, from highly receptive to resistant so it is a bad idea to force teams to adopt a best practice. Teams forced to adopt a new practice are rarely effective at doing so. I recommend using a viral approach where pioneers encourage teams to be the next set of adopters. When early adopters demonstrate results their initiative, risk-taking and results are publicly recognized. These teams then become promoters. The most credible promoter for a practice is a well-regarded engineer versus a senior manager or quality/process improvement leader.

6 Metrics Key to driving Effective Best Practice Adoption

Software metrics are a critical tool for driving effective engineering best practice adoption. Figure 4 below shows a key effectiveness metric for defect removal best practices – the number of minutes of effort to discover a defect by method. Defects discovered through personal reviews or peer reviews required an average of one hour of person-time to find and resolve, whereas the average defect found in system test required about eight to ten hours based on metrics I've seen from some teams at Adobe and Intuit.

It is best to use metrics that measure outcomes and not just activity. For example, having teams count the number of peer reviews they do is insufficient. Even if many peer reviews are being done if they are not finding defects then they are not accomplishing their purpose. What matters most are metrics like the percentage of defects removed using early defect removal methods. Ideally, defects are removed in the same phase they are injected when they are frequently an order of magnitude less costly than the next phase. Quality cannot be "tested into" a product.

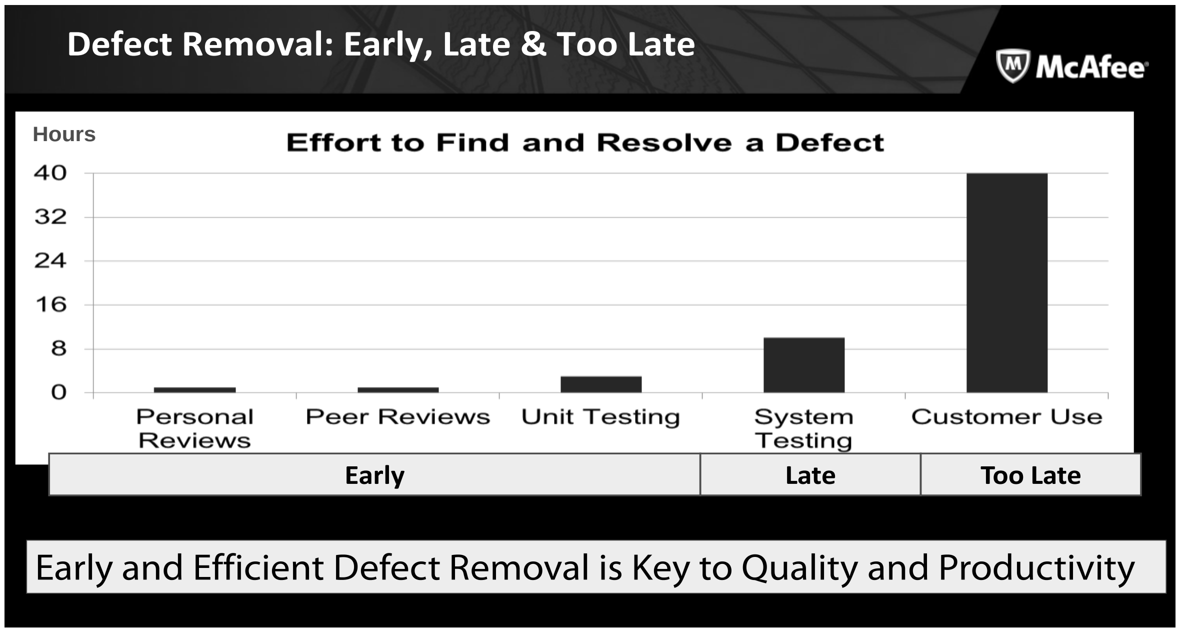

Figure 4: Defect Removal Effort by Method

The data from Figure 4 was collected from both McAfee and Adobe projects. You must measure the effectiveness of engineering practices to understand their return on investment and justify the necessary investments of time. Some teams tell me "Yes, we are already doing peer reviews". When I ask them how many defects they found, what the ROI was for that activity, and how many defects they are finding for each hour of time invested, some teams do not have answers.

To teach engineers the value of using software metrics, I ask them to review a real work product during a peer review workshop where we collect and use key metrics such as the amount of time invested and the number of operational defects found. Without personal experience collecting and using software metrics to improve their own work, engineers often believe metrics are mainly for management.

I was approached by a team leader who said his team would be unable to participate in a Peer Review Workshop I was scheduled to teach because they lacked the time as his team was committed to deliver a software release to QA on the following day. After I mentioned that most teams find an average of three operational defects during the class he agreed to have his team participate. Ultimately, his team found four operational defects during their peer review, at least one of which would have certainly resulted in their missing a commitment to deliver a working software build the next day. After this experience the team leader and team agreed it was a good call to participate in the workshop and that peer reviews could be a very good investment of time.

7 Reducing Defect Driven Rework

A benefit of removing most defects prior to System Test is a significant reduction in engineering rework. Figure 5 shows the percentage of defects removed before System Test for a sample of seven Adobe TSP projects. TSP projects establish and manage goals and plans for removal of defects at each phase of the software engineering lifecycle. They also establish quality plans where they estimate the number of defects that will be injected and removed in each software development activity (e.g. design, coding) for each major component. These plans are used to assess whether a team is being efficient and effective in using early defect removal techniques.

Project	% of Defects Found Early	%Effort in Post-Dev Testing
A	94%	11%
B	83%	15%
C	75 %	16%
D	78%	32%
E	88%	18%
F	83%	9%
G	75%	13%
Average	82%	16%

Figure 5: Team Software Process Early Defect Removal Results

This reduction in post-release rework translated directly into these teams having 30% more time available for value-added work (see Figure 6). In this chart the total time to find and remove defects through all methods is compared (Total Cost of Quality) as well as the average pre-system test removal rate (% of Defects Found Early).

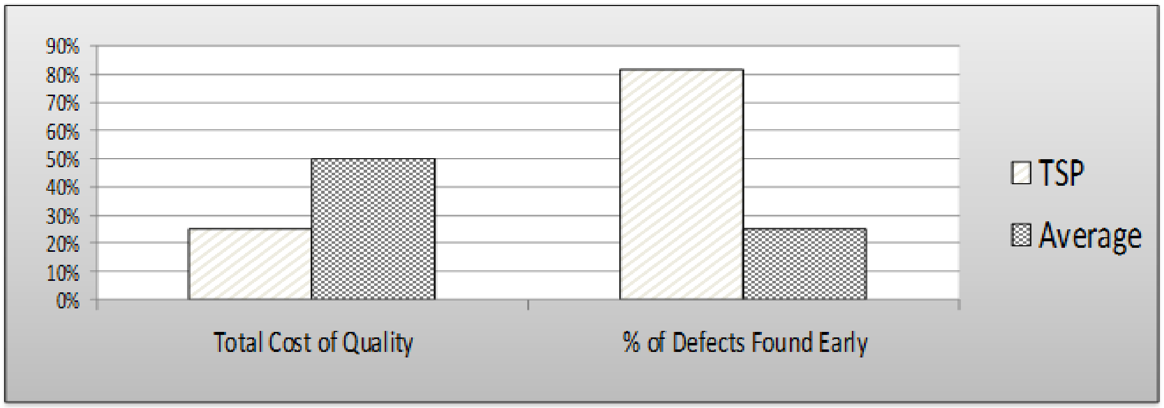

Figure 6: A comparison between TSP Projects and similar projects using traditional methodologies

These teams found a greater percentage of software defects prior to system testing. Defects found before system test are significantly less expensive to resolve. Also, the percentage of defects found before system test is directly correlated with post-release quality since System Testing will not find all defects. These teams were less frequently in fire-fighting mode and able to maintain more predictable work schedules and reasonable work-life balance. As a result, most of these teams had a larger portion of their development cycle available and allocated to design and implementation, versus being spent in a code-and-fix during system testing.

8 Improved Quality through Early Defect Removal

I've done studies at several companies that correlated the number of defects deferred for repair post-release with the number of defects discovered in System (QA) testing. The correlation between these two numbers ranged between .7 and .9. In other words, you can predict the number of defects that a project team will elect to not fix before production release (defer) if you know the number of defects found in System Testing. The two major reasons for the high correlation between these two numbers are:

1. When defects are found during QA testing they are relatively expensive to resolve. As a consequence, teams simply run out of time to fix all of the defects they find.
2. When defects are found in QA they are usually found late in the development cycle. Therefore, they are relatively more risky to fix and some are deferred to avoid the risk of introducing a more serious issue as part of a regression problem.

The bottom line is that an excellent way to increase both quality and engineering productivity is to find and remove a greater percentage of defects early.

9 Summary and Conclusions

To deliver quality work and improve how it is done, the senior leadership of the organization must expect and encourage it. Senior leadership support includes ensuring managers have quality improvement as part of their performance goals as well as regularly review progress with the teams. These quality reviews are also an opportunity to ensure quality is a priority and that the necessary time, dollars and headcount for engineering quality into the software are available and used. They also serve as a mechanism for identifying and spreading adoption of best practices across an organization.

A key enabler of quality is the adoption of engineering best practices including Peer Review, Scrum, Team Software Process and Unit Testing. Self-directed teams adopting TSP and/or Scrum are able to ensure that quality, schedule, scope and cost were not strict trade-offs. Through a combination of better planning and estimation, improved project execution and effective use of quality best practices such as unit testing and peer reviews, teams are able to deliver high quality software, on schedule and budget, with good work-life balance for the team. Learnings from retrospectives and improved metrics helped drive continuous and significant improvement in product and process.

Acknowledgements

I would like to thank Watts Humphrey and Noopur Davis for the transformative influence they have had on many of the teams that they worked with at Adobe and Intuit. I've learned so much about software engineering best practices and how to make this stick from Watts and Noopur. I would also like to thank Senior Leaders I've worked for at McAfee, Adobe Systems and Intuit including Bryan Barney, Barry Hills, Miles Lewitt and Bill Ihrie for their support.

References

1. Campanella 1999, *Principles of Quality Costs: Principles, Implementation and Use*, ASQ Quality Press.
2. Davis, N. 2003, *Team Software Process (TSP) in Practice*, SEI Technical Report CMU/SEI-2003-TR-014 (September 2003), SEI.
3. Humphrey, W. 2001, *Winning with Software*, Addison Wesley Professional.
4. Jones, C. 2010, *Software Engineering Best Practices*, McGraw Hill
5. Kotter, J. 1996, *Leading Change*, Harvard Business School Press.
6. Moore, G. 1991, *Crossing the Chasm*, Harper Business.
7. Reichfield F. 2006, *The Ultimate Question: Driving Good Profits and True Growth,* Harvard Business School Press*. and http://www.netpromoter.com*