Food for Thought™

An e-newsletter published by
Software Quality Consulting, Inc.

November 2011
Vol. 8 No. 4

Welcome to **Food for Thought**™, an e-newsletter from **Software Quality Consulting**. I've created free subscriptions for my valued business contacts. If you find this newsletter informative, I encourage you to continue reading. Feel free to pass this newsletter along to colleagues by using this Forward Email link. If you've received this newsletter from a colleague and would like to subscribe, please use this Enter New Subscription link. If you don't wish to receive this newsletter, use the SafeUnSubscribe link at the bottom of this newsletter, and you won't be bothered again.

Your continued feedback on this newsletter is most welcome. Please send your comments to **steve@swqual.com**

In This Issue

In This Month's Topic I discuss the need to balance defect prevention and defect detection activities.

Regular features to look for each month are:

- Monthly Morsels
  Hints, tips, techniques, and references related to this month's topic

This Month's Topic

An ounce of prevention…

The October snowstorm in the northeastern US left many people without electricity and heat for more than a week. In Connecticut, people were without power for more than 10 days. In Massachusetts, an elderly woman froze to death in her unheated home. Further, local town officials have noted that electric utilities have drastically scaled back maintenance activities such as tree trimming. As a result, the question many people are asking is could this power outage have been prevented or shortened?

In fact, Prof. John Sterman from the MIT Sloan School of Management observed that not only have utilities significantly reduced preventive maintenance, such as simple tree trimming, but also they have reduced infrastructure investments that could have made the power grid more resilient. [1]

Ben Franklin coined the adage

While most people recognize that an **ounce of prevention is worth a pound of cure**, many organizations fail to put this into practice – even though Prof. Sterman's research has demonstrated that preventive maintenance can produce significant returns on investment. Let's look at why this is…

Preventing problems is not sexy

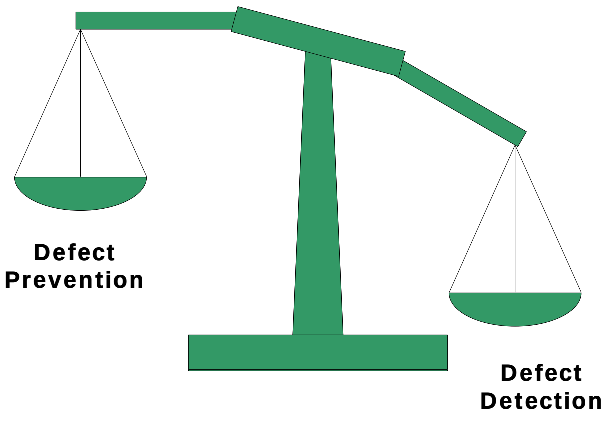
Our culture is fixated on heroes – people who perform extraordinary feats risking life and limb are deservedly given hero status. For example, police and firefighters are often put into situations where they risk their lives to save others from a disaster that could have been prevented. We praise heroic behavior, and rightly so, but we totally ignore people who work often behind the scenes to prevent disasters before they occur.

No one is a hero for preventing problems that never occur.

“An ounce of prevention is worth a pound of cure.”

Within the software engineering community, we have our own “heroes” - developers who are rewarded for fixing problems at the last minute so that the product can ship on time. Upon further review, we often find that these same “heroes” often created the problems that were holding up the release in the first place.

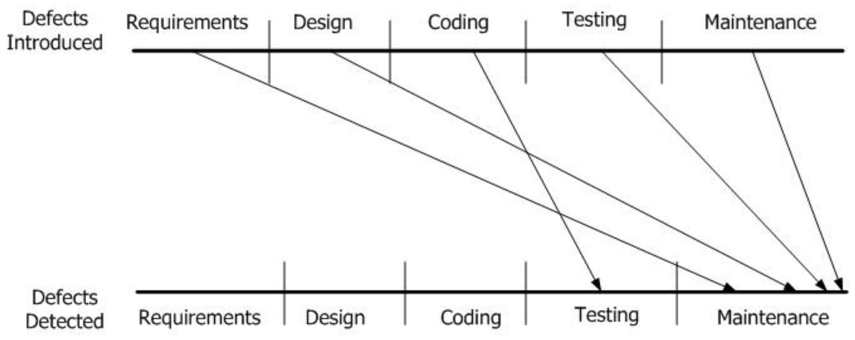
Within the software engineering community, there is very little emphasis on defect prevention. People are not rewarded or even recognized for preventing defects. In fact, we spend much more time and money on **defect detection** than on **defect prevention even though the data clearly shows that it should be the other way around…**



We have long been aware that the longer a defect remains in a product, the more expensive and time consuming it is to fix. Steve McConnell has reported that:

> “… early, upstream defects generally cost 10 to 100 times as much to remove late downstream as they do to remove close to the point where they are created. [2]

The following diagram from Capers Jones [3] illustrates one of the effects of not focusing on defect prevention. All of the defects introduced over the course of development are found at the end of the project – when they are much more costly to fix and much more disruptive to project schedules….



Balancing defect detection vs. defect prevention

The challenge for the software industry is how can we make preventing problems just as important as fixing problems? Here are some suggestions for how to accomplish this…

1. **Take responsibility for your work.**

   Every member of every project team must take responsibility for quality of their work. This especially includes developers and testers! Developers ought to be willing to say:

"I believe my code is as good as it can be and correctly implements defined requirements. I challenge anyone to prove me wrong."

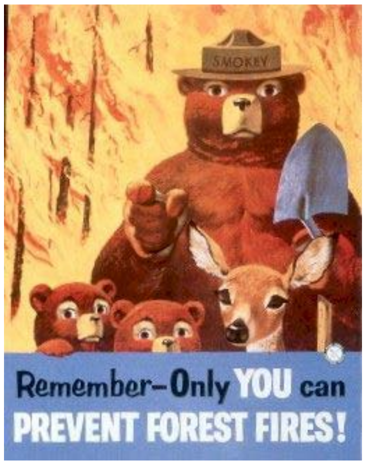
Similarly, testers should be willing to say:

"I believe my tests are as good as they can be and accurately reflect defined requirements and how our customers use our software. I challenge anyone to prove me wrong."

2. Have a good process and follow it.

Preventing defects requires a good, solid development process – that could be Waterfall or Agile – it really doesn't matter. What does matter is that there is a process and the process is followed.

Who is responsible for ensuring that there (a) is a good process and (b) that the process is followed? Management indirectly owns that responsibility. Management must demand that a process be put in place and hold people accountable for following it. In more mature software organizations, Management often delegates some of this responsibility to SQA…



3. Get the requirements right.

Problems with requirements account for the largest percentage of defects in software. Simply getting the requirements right can prevent a significant number of defects.

Requirements training provides project teams with the skills most teams are presumed to have but often don't have. Learning how to write clear, concise, correct requirements is essential to preventing problems.

4. Conduct effective Peer Reviews

Once project teams have been trained in writing requirements, the team needs to learn how to perform effective peer reviews. By far, the most important peer review to perform is a Requirements Review.

People invited to participate in a peer review need to take that responsibility seriously. This means:

- Coming to the review meeting prepared – having read the relevant documents and identified issues and concerns beforehand.
- Paying attention to the discussion during the meeting and not to your blackberry or iPhone.
- Following up with the author to ensure that issues are resolved appropriately and clearly.

5. Create Rapid Prototypes

Rapid prototyping has long been an effective tool for preventing problems since it can enable users to provide critical feedback on issues of usability and functionality early in the development process.

6. Proactively manage project risks

Many project failures are attributable to lack of risk management. Project managers can prevent many problems by learning how to proactively identify, mitigate and manage project risks.

7. Expand the role of SQA beyond testing

Within many organizations, SQA often focuses only on testing. Testing is a

defect detection technique, not a defect prevention technique. By expanding SQA's role to encompass the entire software lifecycle, SQA can add value by performing defect prevention activities.

- Participate in Requirements Reviews to identify poorly written requirements as well as requirements that are not testable.

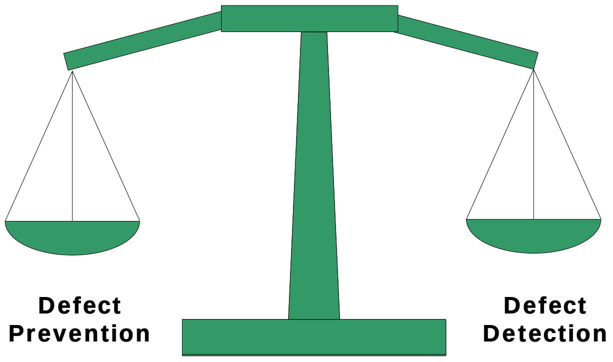
- Provide Management with data on process effectiveness and whether the process is being followed.

- Perform Root Cause Analysis on defects to identify process improvements that can prevent similar defects on future projects.

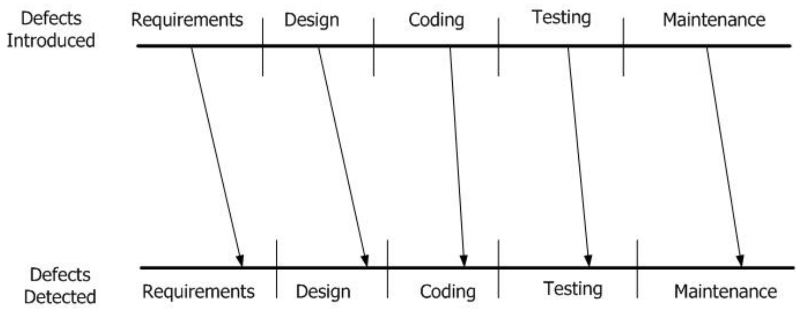
8. **Provide incentives and rewards for preventing problems**

   Management needs to recognize the significant cost savings that can be realized by focusing on preventing problems rather than fixing problems. Management can do this by providing incentives and recognition for those who proactively prevent defects on project teams.

   Project teams need the ability to balance defect prevention and defect detection activities to achieve the best outcomes.



One benefit of using a more balanced approach to defect prevention is illustrated by Capers Jones [3]



Here, we see that defects introduced into the product are found close to the point where they are introduced – which is much more cost-effective.

Try something radical…

Mike McGonagle, a colleague of mine, recently told me of a test team that wanted to push developers to take responsibility for the quality of their code. The test team had clearly defined release criteria, developed tests against requirements and ran those tests. If the test results met the release criteria, the product could ship.

If the test results didn't meet the release criteria, the product wouldn't ship. In this case, the test team **would not** provide the developers with a list of the tests that failed. It was left to the developers to figure out what wasn't working.

If more project teams worked this way, developers would eventually take responsibility for the quality of their code – which means that they would deliver code to the test team with far fewer defects. This is an interesting example of providing the right kind of incentives to prevent defects.

Imagine if electric utilities were fined $10 million a day for every day that customers were without power. While we would still have occasional power outages, they would be far shorter in duration than the outages we experienced with the October snowstorm. An ounce of prevention really is worth a pound of cure.

'till next time…

Monthly Morsels

Every month in this space, you'll find additional information related to this month's topic.

1. Sterman, J., "Utilities need to cut trees, not costs", Boston Globe Nov. 4, 2011

2. McConnell, S., Rapid Development, Microsoft Press, Redmond, Wash., 1996.

3. Jones, C., Software Quality: Analysis and Guidelines for Success, Thomson Computer Press, 1997.

About SQC

Software Quality Consulting provides a full-range of software engineering services for safety-critical industries and mission-critical projects. Our goal is to help create safety-critical and mission-critical software that meets our client's needs, complies with all applicable standards and regulations, with the highest level of quality possible, and in the most cost-effective and timely manner possible.

To learn more about how we can help your organization, **visit our web site** or **send us an email**.