

Understanding the Role of Defect Potential in Assessing the Economic Value of Process Improvements¹

David Raffo², Warren Harrison³, John Settle², and Nancy Eickelmann⁴

Introduction

Software companies face relentless pressure to reduce costs, improve quality, and improve time-to-market. To accomplish these objectives and to remain competitive, companies must improve their software development processes. Before companies invest in process improvement activities, management would like to know the economic value of potential process changes and to prioritize process changes based upon potential return.

Financial techniques have been applied to assess the economic value of various process improvement activities. These include simple payback and cost-benefit ratios [Curtis,1995; McGibbon, 1996], standard present value techniques [Slaughter,1998; Vienneau,1995; Harrison, Raffo, and Settle, 1999b, 1999c], risk and return formulations [Harrison, Raffo, and Settle, 1999a] and options theory [Sullivan, 1996; Sullivan, Chalasani, and Jha, 1997], to name just a few. Such techniques have provided insight and tools for managers making economic based decisions about software process improvements.

Many process changes have been proposed in the literature. Many more process changes are creatively defined within companies to address specific company needs. Most process changes that are proposed have a structure where an initial investment of time and effort yields some improvement later in the process. Some examples include:

- Investment of time in inspections reduces defects later in the process. Not only is quality improved, but effort and schedule are saved as well.
- Investment in purchasing a new tool and training employees on how to use it yields better requirements, which again not only improves quality, but also provides savings in terms of effort (i.e. costs) and schedule during subsequent process steps.
- Investment in developing test plans up-front improves coverage, quality, and efficiency of conducting tests.
- Investment to learn and apply a new test technique enables QA to detect a new type of defect before it goes out to the customer improving quality, the company's reputation, and saving costs.

The list can go on. From these examples, we find that the notion of quality and how it is valued is a central issue in evaluating the impact of most process improvements. It is the authors' position that costs associated with quality (or defects) have been modeled incorrectly in the past - in the sense that activities that are made up of fixed costs have been analyzed as variable costs. We further take the position that a number of significant costs associated with defect processing among others have been overlooked by most analyses.

In this position paper, we introduce the notion of *defect potential* and propose a framework and high level model that, in our view, better accounts for fixed and variable costs and can be used for assessing the defect potential.

¹ This work has been funded in part under NASA Contract NCC 2-1152 with NASA's IV&V Facility, Fairmont, WV.

² School of Business Administration, Portland State University, Portland, OR 97207-0751

³ Department of Computer Science, Portland State University, Portland, OR 97207-0751

⁴ Software Research Laboratory, NASA IV&V, Fairmont, WV 26554

Defect Potential and Its Impact on Software Development Organizations

Software developers understand that all software has the potential of having defects. This potential causes organizations to perform numerous expensive activities to either prevent, or find and correct, errors in their software. Many of these efforts are undertaken whether any defects exist or not. However, most studies ignore the phenomenon of defect potential and its impact, and choose instead to focus on defects actually found. This is misleading, because software development organizations perform various QA activities not because defects are known to exist, but because they have the potential to exist in the product.

For instance, if one focuses on defects found, error prevention activities such as inspections or cleanroom might appear quite costly. This is because, if they perform correctly, the result should be very few errors. If the cost-benefit measure of the activities is cost per defect, they may appear to be overly expensive activities. This counter-intuitive result is because prevention activities, which are actually a fixed cost, are being presented as a variable cost. To gain an understanding of the value of Validation and Verification activities throughout the lifecycle, we must understand the true impact of software defect potential on cost. The cost of defect potential comprises the following five components:

1. *The cost of preventing defects* - these resources are expended in preventing defects from occurring – for instance better design practices, cleanroom, process certifications or inspections⁵, etc. are all intended to reduce the number of defects by preventing them from occurring. In general, this is a fixed cost since it is not functionally dependent upon the number of defects actually found. We may consider the cost of preventing defects to be the same whether we find ten defects or ten thousand. However, as with many fixed costs, they are fixed only over certain ranges and may increase as a step function once a certain number of defects are found.
2. *The cost of searching for defects* – these resources are expended in looking for defects that may have occurred – for instance inspections and walk throughs of the software product, developing software test plans, writing test scripts and running test cases. In general, these are fixed costs since they are not directly related to the number of defects actually found⁶. Costs in this category *do not* include efforts undertaken after a defect is actually found (such as regression testing or developing workarounds). Costs incurred after a defect is found would fall into other categories as described below. As with the cost of preventing defects (above), searching costs are actually fixed only over certain ranges. For instance, if we find an unexpectedly large number of defects during testing, we may increase our testing budget.
3. *The cost of isolating and verifying (i.e. processing) defects* – these resources are expended to isolate and verify the defect as well as to record, track and establish the disposition of an anomaly once it is detected. For instance, when a defect is detected, it is customary to analyze the defect to see if it is a duplicate, if the behavior is correct and the test case is in error, etc. Likewise, once an anomaly is determined to be an error, a decision must be made as to whether it should be corrected or not. For instance, many non-critical defects are never corrected, or at best their correction is deferred until the next scheduled upgrade. This analysis is not free, and in fact can represent a significant portion of the effort expended when dealing with a defect (especially in mission critical systems). In general, these are variable costs (though there may be a fixed component to provide the necessary infrastructure) and are directly related to the number of anomalies found since every anomaly, once found must be reported, logged, tracked, resolved, etc.
4. *The cost of fixing defects* – these resources are expended to correct defects that have been found, and determined to require correction. In general, these are variable costs because the total cost of fixing defects is proportional to the number of defects corrected. It should be noted that the relationship between the total cost to fix defects and the number of defects corrected is not linear, since (a) each

⁵ Inspections can belong to either of the first two cost categories (prevention or searching).

⁶ However, the number of defects found is a function of the effort spent looking for defects and the effort expended trying to prevent them

defect (or defect type) costs a different amount to fix and (b) some defects are not fixed but simply resolved by documenting them, providing a work-around, or having some other resolution.

5. *The cost of defect occurrences* – defects that “slipped through” the defect detection process, or defects that were found during the search activity but not fixed and are subsequently encountered after delivery, usually will have some measurable (and in some cases, immeasurable) impact (cost) associated with them. In a large sense, these costs are the most speculative of the all the component costs. Data regarding program costs as well as the probability a given type of defect will cause a failure are needed.

A Model of the Cost of Defect Potential and Information Needs

Identification of the various cost components leads to a straightforward model of the cost impact that software defect potential has on a project:

$$\text{Cost Impact} = \text{Cost}_{\text{prevention}} + \text{Cost}_{\text{search}} + [\text{Cost}_{\text{processing}} \times \text{Anomalies}] + [\text{Cost}_{\text{repair}} \times \text{Anomalies} \times \text{Repair}_{\text{rate}}] + [\text{Cost}_{\text{occurrence}} \times \text{Defects}_{\text{experienced}}]$$

where

- $\text{Cost}_{\text{prevention}}$ - a measure of the resources spent on activities intended to keep defects from being made, such as process improvement.
- $\text{Cost}_{\text{search}}$ - a measure of the resources spent on activities intended to find defects that have been made, such as plan-based testing.
- $\text{Cost}_{\text{processing}}$ - a measure of the resources spent on the activities that occur when an anomaly is observed, such as logging, categorizing and tracking.
- $\text{Cost}_{\text{repair}}$ - a measure of the resources spent making corrections and validating those corrections.
- Anomalies - an issue found during a search activity - sometimes these turn out to be defects, and other times they turn out to be features, duplicates, etc. which do not trigger a corrective action.
- $\text{Repair}_{\text{rate}}$ - the average percentage of anomalies that actually get *repaired* (not the number that are defects).
- $\text{Cost}_{\text{occurrence}}$ - the average cost of a defect that causes a failure in the product after it goes into production.
- $\text{Defects}_{\text{experienced}}$ - the number of defects that actually cause a production failure.

Based on this model, we require metrics allowing us to:

- (a) Establish the cost of error prevention activities – i.e., the cost of planning and performing the various prevention activities, such as inspections, walkthroughs, etc.
- (b) Establish the cost of implementing the QA plan – i.e., the cost of planning, administering and applying the various test suites.
- (c) Establish the average cost of processing an issue/anomaly once discovered – that is, the amount of effort expended in logging, analyzing, and overseeing the issue/anomaly up to the point where a disposition is assigned, as well as the percentage of each type of disposition assignment (i.e. percentage of defects, duplicates, non-defect, etc.)
- (d) Establish the percentage of defects actually assigned for correction, as well as the average effort required to make the correction, verify it and reintegrate the corrected component on a per defect basis, as well as percentage of defects that have work-arounds assigned, and the average effort involved in producing a work-around.

- (e) Establish the cost of a defect occurrence in the field.

These five components (i.e. the cost of searching for a defect and so forth) are notable because they include many activities and costs not traditionally associated with software defects. The traditional view of software defects is that their cost is predominately the effort required to repair them (a subset of the above-mentioned “cost of fixing defects”). We take a different view of this.

Technology, Costs and Defect Potential

To discover the true benefit of adopting new technology meant to reduce the cost of defects, we must consider each of these costs separately. Increases in resources expended in some components may result in fewer resources being spent in other components and vice versa.

Adopting a prevention technology (such as inspections) may increase the costs of one component, while reducing the costs involved in the later components with variable costs because there are fewer defects. Other technologies may do nothing for the number of defects (e.g. a new defect tracking system), but reduce the cost of handling a defect once it is found, and thus the cost per defect will be reduced. Increases in some components with fixed costs may allow other fixed costs to be reduced. For instance, many advocates of cleanroom (a prevention technology) suggest that fewer resources need be spent on testing (a searching technology) because cleanroom makes testing redundant.

This model provides a framework for understanding the impact of defects that can be further developed to provide added granularity as data are collected. For instance, defect costs may be represented as distributions by various defect types. Moreover, the cost of defect occurrences (impact) could include a probabilistic assessment for various risks given the distribution of defects that might occur.

The model also has implications for metrics programs that typically focus almost exclusively on rework costs while neglecting the variety of other costs involved (i.e. costs of developing work around procedures, the cost of recording and tracking defects and so forth).

Clearly when introducing a new technology in one component, such as prevention, its additional costs must be balanced by reductions in other components.

Reducing the cost of defect potential is probably the most significant activity that can be undertaken by a software development organization to manage costs. However, in order to determine the economic benefits of taking steps to reduce the cost of defect potential, we must know the expected difference between the actual cost of defects with our current process and project what the costs will be after adoption of any new technology or process improvement. In our current work with the NASA IV&V facility, we are applying this model to assess the cost of defects and the value of IV&V activities.

References

[Curtis,1995] Curtis, W., “Building a Cost-Benefit Case for Software Process Improvement,” Notes from Tutorial given at the *Seventh Software Engineering Process Group Conference*, Boston, MA, May 1995.

[Harrison, Raffo, and Settle, 1999a], W. Harrison, D. Raffo and J. Settle, “Process Improvement as a Capital Investment: Risks and Deferred Paybacks”, *Proceedings of the Pacific Northwest Software Quality Conference (PNSQC)*, Portland, Oregon, October, 1999.

[Harrison, Raffo, and Settle, 1999b] Warren Harrison, David Raffo and John Settle, “Measuring the Value from Improved Predictions of Software Process Improvement Outcomes Using Risk-Based Discount Rates”, *First Workshop on Economics-Driven Software Engineering Research*, in Conjunction with *21st International Conference on Software Engineering*, May 17, 1999, Los Angeles, California.

[Harrison, Raffo, and Settle, 1999c] David Raffo, John Settle and Warren Harrison, “ Estimating the Financial Benefit and Risk Associated with Process Changes”, *First Workshop on Economics-Driven Software Engineering Research*, in Conjunction with *21st International Conference on Software Engineering*, May 17, 1999, Los Angeles, California.

[McGibbon, 1996] Thomas McGibbon, *A Business Case for Software Process Improvement*, Data Analysis Center for Software State-of-the-Art Report, prepared for Rome Laboratory, September 30, 1996

[Slaughter,1998]. S. Slaughter, D. Harter and M. Krishnan, “Evaluating the Cost of Software Quality”, *Communications of the ACM*, August 1998, pp 67-73.

[Sullivan, 1996] Sullivan, K., [Software Design: The Options Approach](#), 2nd International Workshop on Software Architecture, Proceedings of the Second Joint SIGSOFT/ACM Conference on the Foundations of Software Engineering, October, 1996.

[Sullivan, Chalasani, and Jha, 1997] Sullivan, K., P. Chalasani and S. Jha, [Software Design Decisions as Real Options](#), University of Virginia Computer Science Department, Technical Report. June 1997. Submitted for publication.

[Vienneau,1995]. R. Vienneau, “The Present Value of Software Maintenance”, *Journal of Parametrics*, April 1995, pp. 18-36