



Software Inspections: Pay Now, or Pay More Later

by **Gary Gack**

When we're making decisions about our cars, most of us believe that 4 quarts of prevention every few thousand miles is a worthy investment. Why is it most organizations don't apply the same idea to software development? Surely most of us know by now that inspections can prevent software breakdowns in the same way that oil changes prevent engine repairs.

The published evidence provided by Humphrey, Gilb, Grady and others certainly establishes beyond doubt that formal "Fagan style" inspections are effective. So why don't we use proven defect detection techniques? What could we expect to gain if we did?

Watts Humphrey, in *A Discipline for Software Engineering*, makes the following observation. "You might ask why organizations do not do more reviews and inspections. There are two reasons why not. First, few organizations have the necessary data to make sound development plans. Their schedules are based largely on guesses, and these guesses are unrealistic. When their plans are treated as accurate projections, the schedule pressure builds so quickly that all the engineers can do is react to the periodic crises. No one has any time to think of anything but test, debug, and fix. Second, yield is not managed... the engineers have no data on the number of defects they inject or the cost to find and fix those defects. They thus rarely appreciate the enormous costs that can be avoided by fixing defects before test."

In this article I will explore (with real data from a client company) how the economics of a project will develop under two distinct scenarios: in the first scenario the project is managed by Snidely Outsource Unlimited, who commits to deliver 2500 Function Points (FP) in 14 months. A crafty old in-house veteran manages second project of the same size but will require 24 months to deliver. (In neither case was there an explicit statement about the expected quality levels of the delivered products.)

In the first scenario, Snidely actually delivered a system in 17 months at a cost of slightly less than \$ 1,200 per FP (total developer's fee \$ 3.1mm). In the first year, this system experienced approximately **2500** category 1 (Abends) and category 2 (user reported problems) defects, or about 1 per FP. Category 3 (test) and category 4 (inspection) defects were not tracked, and inspections were not performed. In other words, this project conformed precisely to Humphreys description.

© 2007 Gary A. Gack
Process-Fusion.net, LLC
1568 Linkside Drive ♦ Orange Park, FL ♦ 32003 USA
+1 (904) 579-1894 ♦ info@Process-Fusion.net

In the second scenario, Crafty actually delivered the system in 24 months at a cost of less than \$ 900 per FP (total \$ 2.1mm). In the first year, this system experienced **18** (!!) C1 + C2 defects. In this project, *everything* was inspected - requirements, high-level design, detail design, code, test plans, test results, - *everything*.

Who do you suppose still has their job?

When we follow these two scenarios through to their conclusions, taking into account the remedial maintenance costs and additional support required associated with Snidely scenario, we find that the 7 year life cycle cost of scenario 1 is over \$6mm, while the life cycle cost of the Crafty scenario is less than \$3mm. In other words, to save 7 months, the company spent an additional \$3 mm. – while that may sometimes (rarely) make sense, it definitely did not in this instance. Why did this occur? I believe that Humphrey’s analysis is “right on” in this real world instance. Snidely had no metrics history about defects or delivery rates and hence made unrealistic commitments - they had no understanding of (or perhaps no concern about) the impact on the defect rate. They were operating in “take the money and run” mode.

Crafty, on the other hand, had extensive metrics on defects and delivery rates and had used that information to build a patient, mature development culture in which preventive investments in inspections are *a/ways* made, and high quality, cost-effective systems are nearly always delivered. Crafty was operating in “we have to live with this” mode.

Use of software inspections pays - in reputation, in user satisfaction and in job security!