Opinions from the Testing Front

From: Terry Horwath

Date: May 2002

Subject: Why Substituting Test Engineers with Development Engineers is Ineffective

---

Over the last 25 years I have been involved in several business downturns in Silicon Valley. In each of those downturns I have been with companies that have decimated the software QA group during layoffs using the rationale that the missing test engineers can be replaced occasionally by software development engineers when it is "*time to test*". This whitepaper points out why this strategy is ineffective.

1. **"What I don't understand must be easy."** —Dilbert's pointy haired boss

   Performing good testing is a hard, full time activity. The job requires attention to detail, strategic test planning skills, tactical test case design skills, as well as writing and verbal communication skills. It also demands the ability to remain focused and vigilant—often while executing tasks and test cases repeatedly—yet to perform those tasks with slight variations on each iteration to tease serious bugs out of the product. The *repetitive yet focused* nature of testing is the skill set which is most often opposite that required to be a good developer of new features and functionality.

2. **Innocent until proven guilty.** *Or, it's the focus baby.*

   In a courtroom setting most participants in the legal process are required to assume that the accused is innocent until proven guilty. The only exception to this rule is the prosecutor, who is required to assume that the accused is guilty in order to perform their role effectively. In the software development environment the development engineer is the defense attorney while the QA engineer assumes the role of the prosecutor. No one would think it sensible to ask a defense attorney to switch sides and *prosecute the case for a few days* at the end of a trial, when they assumed the accused in innocent. So why is it thought sensible to ask a development engineer to *test the product for a few days* at the end of a release cycle, when they assume that the product is bug free?

3. **Forced conscripts make bad volunteers.** *Or, it's the attitude baby*.

   Given the choice most development engineers would prefer to have a root canal, rather than perform testing of any kind. But in these downturns they are *volunteered* by their managers with assurances that they will perform *careful, thorough system testing*. Under these conditions development engineers will tepidly test exactly as directed. In the absence of detailed testplans—which rarely exist—this direction will be minimal at best, and in most cases so will the testing that is performed by the *volunteers*.

4. Good testing should produce documented results.

The primary purpose of a test group is to produce information used to make a business decision as to when a software release is *fit to ship*. This requires the maintenance of detailed statistics associated with a plethora of test related activities *as the testing progresses*. Just when the development engineer thought the job of testing could not get any worse they encounter this additional responsibility. Most often this aspect of the *volunteer* work is ignored—thereby leaving only the verbal opinion of the developer as to the product's fitness to ship.

5. Good test engineers have an extra eye.

Good test engineers employ a 6th sense to "*know*" where to look for and find problems; this skill borders on an art form. This extra sense is only developed by those engineers willing to invest years pursuing testing as a serious career discipline. Some of the best bugs are found by looking to the *left or right* of the current testing activity—at the just the right time!—to catch some of the nastier problems a system is hiding. These types of problems are unlikely to be detected by the casual test *volunteer*.

