



White paper

Did they ever test this, even once?

The importance of testing everything in detail

www.stdtime.com
Scoutwest, Inc.

Did they ever test this, even once?

Have you ever tried a feature of a product and immediately found that it crashed or misbehaved badly? Your immediate reaction is, “did they ever test this?” Surely, if the developer tested the feature, he would have immediately seen the grossly errant behavior and fixed it before releasing it, right? My personal belief on this subject has evolved to a rather distrustful one: if it hasn’t been tested, it probably doesn’t work.

There are a lot of reasons for this. The biggest reason is that many development companies do not have QA departments, or anyone who carries the responsibility to objectively test the products. The same engineers who develop the products perform the testing. That means they have a hard time being objective. They miss defects and usability issues because they are too “close” to the product, and they just don’t want to take the time to make things work perfectly. Testing is left to customers who contact the developer when the product doesn’t work as advertised. Developers who can’t afford a QA department must still force themselves into an objective testing mode.

Sometimes there are just too many new features to juggle. There is just not adequate time and mental capability to test everything well. Something is bound to slip through the net. That’s one of the primary reasons I recommend using a micro-release strategy discussed elsewhere in this book.

A QA team without a punch-list of features is flying in the fog. You can’t expect to get 100% feature coverage without a list of them to test. A simple checklist of features is a simple place to start. A more exhaustive QA plan would be even more helpful. I once worked on a project where the company brought in temporary QA help. The temps were unfamiliar with the product, didn’t have a checklist to work from, and relied upon the developers for guidance. Big mistake. The QA team never achieved good feature coverage because they had a steep

product learning curve. In all fairness, it's good to have fresh eyes to look at a product objectively, but new eyes don't know the depth of each feature, and can't test deeply enough to guarantee quality.

Old features sometimes get "broken" by new ones. Sometimes when developing new features it's hard to anticipate critical overlap with older ones. If you don't test the older features along with the new ones, a customer might find some pretty nasty unforeseen results. The customer might come to the conclusion that you never once tested the broken feature, when in fact, you tested it well, but not after implementing new ones. Sometimes life just isn't fair, and this happens.

Very frequently products are rushed to market, resulting in poor quality. I suppose there are a few legitimate business decisions for this, but the habit always bothers me. I use two high-visibility software products (which will go unnamed) that were rushed to market. They can't be trusted to do anything right, and it bothers me anytime I use them. Perhaps it's better to have "bothered" customers than none at all.

Software development is particularly tricky as it relates to unforeseen circumstances and software configurations. Customers may experience crashes or wrong behavior because of their particular software configuration. I've seen a thousand bizarre configurations that affect software programs. There's always a logical reason for conflicts, but not always easy to foresee when developing new features. Only experience and defensive programming can guard you in the area.

Finally, I sometimes like to engage in a "test it until it breaks" session. I learned this from a hardened QA soldier. He claimed that every feature will break, given enough time. Nothing got past him. He would beat upon a certain feature unmercifully until he found the enemy's weakness – no quarter! We all assumed that his approach would only result in finding small "polish bugs", but not so. Often he'd find clearly noticeable critical bugs this way.

About Us

Scoutwest, Inc. develops and publishes project management and time tracking products for consulting, manufacturing, government, and general business applications.

Thousands of small to large businesses, in dozens of countries worldwide, trust their mission critical business processes to Scoutwest products. Standard Time® and Standard Issue® work together to offer well-rounded project management solutions.

We specialize in packaged software for timesheets, project management, time tracking, defect tracking, and issue tracking. Standard Time is a web-based timesheet that also runs on Windows, Palm OS, and Pocket PC. It can be used for client billing and task management. Standard Issue is used for bug tracking and general issue tracking.

Please visit these web sites for more information.

www.stdtime.com
www.sdtissue.com