

Testing Methods RFC

Source: <http://coding.derkeiler.com/Archive/General/comp.programming/2004-08/1058.html>

From: Jonathan Allen (X_at_X.X)

Date: 08/26/04

Date: Thu, 26 Aug 2004 14:16:58 GMT

We have found that our method of testing does not match traditional text-book methodologies. I decided to write a short white paper on it so that I could get your opinions. Does anyone else use this method? If so, what the heck is it called? Do you think it would work in other projects, or did we just luck out?

Jonathan Allen

Active Testing
Jonathan Louis Allen

Abstract

This document describes a process for integrating testing into the design and development phases of the software life-cycle. This is in lieu of or in addition to the standard practice of having a separate QA department. The advantages of this process are bugs are detected sooner and developers are more efficient.

Background

The most difficult challenge facing software developers continues to be quality assurance. As applications grow more complex, the task of ensuring a program works correctly becomes increasing difficult.

Timeliness of Bug Reports

When should a bug be reported? Ideally, it should be reported immediately after the code that caused the bug is written. There are two principal reasons why this is so important.

First, a project's design is the most flexible at the early stages of development. As time goes on, it becomes increasing more expensive to change the design. If a bug requires a major design change late in the game, it will not only impact the affected component, but also any component that relies upon it. If a bug can be caught before the dependant component are

built, then said components won't incur a refactoring cost.

The second reason has to do with human nature. As time passes, ones memory of a given topic degrades. If a bug is not discovered until the developer has moved on to other components, then time must be spent relearning how the component works. This is in addition to the needed to actually correct the issue.

The goal of Active Testing is to minimize the time between when a component is created and when bug reports are generated. In order to do this, testing must be integrated directly into the design and development phases of the software lifecycle. This minimizing of time to report is what differentiates it from the traditional models.

The Classic Life Cycle Model

Under the venerable Waterfall Model, testing occurs very late in the software life cycle. In fact, tests are not even written before coding is complete. Even under ideal circumstances, this does not allow much time for testing. When projects begin to fall behind schedule, time devoted to testing is further cut.

The Modern Approach

The modern approach to testing requires a separate quality assurance team. This QA team begins work at the same point the development team starts. This, in theory, allows for sufficient time to develop tests.

The problem with this model is the lack of communication between the development and QA teams. Once the requirements are set, the QA team disappears behind the "cinder-block wall". They do not enter the picture again until it is time to actually conduct the tests.

Both of the models have the same flaw. Bugs are not reported until very late in the software lifecycle. Even if there is time to conduct the tests, no time is set aside for actually fixing the issues.

Extreme Programming

Under the Extreme Programming model, tests are written immediately prior to the code itself. This methodology has advantages over the "wait until the end" method of testing, but it has its own set of flaws. First, the same person writing the code is writing the test. So any misconceptions or logic mistakes the developer may have will find its way into both the test and the code. Just as in publishing, the person who writes the book should not be the one who edits it. There is also a question of discipline. Despite their good intentions.

Programmers Hate Testing

It is a known fact that most programmers simply hate testing. They get enjoyment from challenges like designing and developing programs. Testing and documentation are considered chores. And like most chores, people will avoid them if they can.

Keep in mind that not all programmers feel this way. Some actually enjoy the testing process. For them, the challenge is finding (and correctly) flaws in the software. Normally these programmers either keep quiet or find themselves sequestered in QA teams. However, there is a better place for them.

Active Testers

An Active Tester is one that is actively part of the development team. They cannot be merely a member of the QA team that hangs out with developers; they must be a skilled developer in their own right. And they must work closely with the developers on a daily basis.

A Typical Day

The tester begins the day by checking out the most recent version of the source tree. An end-to-end test is conducted first. This is to determine if there are any major flaws that will impact the developers. Developers rely on one-another's code. If a component that their piece requires is broken, they may not be able to do their job. By watching out for major problems, the tester can reduce downtime for the entire team.

Once the end-to-end test is completed, the tester can focus on a specific component of the project. Normally, the tester determines which component to focus on based on these criteria.

- What is new and untested?

- What has changed significantly?

- What has not been tested recently?

Once a bug or flaw is detected, a repeatable test must be created. The type of test is situation-dependant. What is important is that it can be used to (a) demonstrate the flaw and (b) verify the flaw has been corrected. The bug report, including the test, should then be logged.

Handling Bug Reports

Unlike in traditional testing, the Active Tester's role does not stop once a bug report is submitted. Since the tester has access to the source code, the tester should immediately begin looking for the cause of the bug. If possible causes are found, they should be given to the developer directly. The tester can then assist the developer in determining the correct course of action. The possibilities actions include.

Simple Fix: The developer either corrects the flaw immediately, or assigns another developer to correct it. Since the tester is a developer and familiar with the bug, the tester may be the one assigned to make the correction.

More information is needed: The developer requests more information about the issue. This may include stack traces, the insertion of debug code, and variations of the test using different input. The tester gathers this information and reports back to the developer.

Complex Problem: A complex problem requires a major effort on the part of the developer and possibly a design change. At this point standard bug handling procedures such as PCCB go into effect.

Developer Requests

Developers may initiate tasks for the tester. These take priority over the general testing done on a daily basis. They fall into two categories. The first is a request to test a specific component. This is generally done when a complex test is required to satisfactorily show a component is working correctly. Often the developer will tell the tester what the specific concerns are so the tester knows what to look for.

The other request is assistance with optimization. The assumption here is that the tester has access to a lab with tools for benchmarking and analyzing code. When a bottleneck is located, the developer works on improving the code while the tester creates a reliable performance test. Once both tasks are complete, the tester runs before and after benchmarks and reports the results. If the developer determines that further optimization is needed, the cycle repeats.

Bug Tracking

The tester should keep a close eye on the big tracking database. Any valid report should be re-tested on a regular basis. For open reports, this is to determine if they were corrected and the database was simply not updated. One does not want developers trying to fix bugs that were already corrected. For closed reports, this is to ensure that old problems have not resurfaced.

The Design Phase

During the design phase, there is no code to test. So what does the tester do? The tester assists in writing the design documents. In order to assist in correcting bugs, the tester must be familiar with details of the system. The best way to gain this familiarity is to participate in the design process.

While the developers are looking for ways to build the system, the tester should be looking for ways to break it. In effect the tester is playing the devil's advocate, challenging any potential flaws in the design. Keep in mind that the tester does not make the ultimate decision; that has to be left to the developers. The tester's veto power over a design comes once the code is written and tested.

Conclusion

comp.programming: Testing Methods RFC

The Active Tester role does not replace a formal QA team. The tester is a resource of the development team. By assisting in early identification of bugs, the tester helps the developers work more efficiently. Formal testing must still be conducted at the end of the software lifecycle. However, far fewer bugs should arise from the formal testing.