Walkthroughs, Code Reviews, and Inspections

Our objective with Inspections is to reduce the Cost of Quality by finding and removing defects earlier and at a lower cost. While some testing will always be necessary, we can reduce the costs of test by reducing the volume of defects propagated to test.

—Ron Radice (2002)

When you catch bugs early, you also get fewer compound bugs. Compound bugs are two separate bugs that interact: you trip going downstairs, and when you reach for the handrail it comes off in your hand.

—Paul Graham (2001)

Here’s a shocker: your main quality objective in software development is to get a working program to your user that meets all their requirements and has no defects. That’s right: your code should be perfect. It meets all the user’s requirements and it has no errors in it when you deliver it. Impossible, you cry? Can’t be done? Well, software quality assurance is all about trying to get as close to perfection as you can – albeit within time and budget. (You knew there was a catch, didn’t you?)

Software quality is usually discussed from two different perspectives, the user’s and the developer’s. From the user’s perspective, quality has a number of characteristics – things that your program must do in order to be accepted by the user – among which are:¹

- Correctness: The software has to work, period.
- Usability: It has to be easy to learn and easy to use.

- **Reliability:** It has to stay up and be available when you need it.
- **Security:** The software has to prevent unauthorized access and it has to protect your data.
- **Adaptability:** It should be easy to add new features.

From the developer’s perspective, things are a bit different. The developer wants to see the following:

- **Maintainability:** It has to be easy to make changes to the software.
- **Portability:** It has to be easy to move the software to a different platform.
- **Readability:** Many developers won’t admit this, but you do need to be able to read the code.
- **Understandability:** The code needs to be designed in such a way that a new developer can understand how it all hangs together.
- **Testability:** Well, at least the testers think that your code should be easy to test. Code that is created in a modular fashion, with short functions that do only one thing, is much easier to understand and test than code that is all just one big main() function.

Software Quality Assurance (SQA) has three legs to it:

- **Testing:** Finding the errors that surface while your program is executing, also known as *dynamic analysis*.
- **Debugging:** Getting all the obvious errors out of your code, the ones that are found by testing it.
- **Reviews:** Finding the errors that are inherently in your code as it sits there, also known as *static analysis*.

Many developers – and managers – think that you can test your way to quality. You can’t. As we saw in the last chapter, tests are limited. You often can’t explore every code path, you can’t test every possible data combination, and often your tests themselves are flawed. Tests can only get you so far. As Edsger Dijkstra famously said, “…program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence.”

Reviewing your code – reading it and looking for errors on the page – provides another mechanism for making sure that you’ve implemented the user’s requirements and the resulting design correctly. In fact, most development organizations that use a plan-driven methodology will not only review code, they’ll also review the requirements document, the architecture, the design specification, the test plan, the tests themselves, and the user documentation. In short, all the *work products* produced by the software development organization. Organizations that use an agile development methodology don’t necessarily have all the documents mentioned above, but they do have requirements, user stories, user documentation, and especially code to review. In this chapter we’ll focus on reviewing your code.

---