Chapter 2

Why XP Teams Need Testers

Much of the published material on Extreme Programming is aimed at programmers, customers, and managers. Some purists may argue that a tester role is unnecessary in XP projects: customers can write the acceptance tests and programmers can automate them. This can work, and certainly some successful XP projects don't have testers.

We believe, however, that more XP teams can be successful by doing a better job of defining, automating, and running acceptance tests when someone is focused on that role and that this focus helps in other areas as well. If you don't like to think of someone in the tester "role" on an XP project (because the only true roles defined in XP are programmer and customer), think of having a programmer with a "tester focus."

We'll illustrate this point with a true story about a remodeling project Lisa recently experienced, but first let's define what we mean by the term "tester."

Definition of Tester

By the term **tester,** we mean a person who not only develops and runs tests but has quality assurance and development skills as well. Testers spend a portion of their time actually testing, but they also help customers write stories, and they define acceptance tests and take the lead in automating them.

We're combining what would traditionally be described as both quality *assurance* and quality *control* activities. The classical distinction between these two is that quality assurance (QA) aims to avoid the introduction of defects in the first place, and quality control (QC) aims to detect defects that have already been introduced. QA is process oriented; QC is product oriented.

While this distinction is useful for manufacturing, it's less so for software, and our experience has been that either activity alone is generally useless. In any case, XP tries not to overly classify people, because to do so implies that they alone are responsible for coding, testing, analysis, and so on, when XP is all about integrating these activities.

If you're used to thinking of a "test developer," "quality analyst," or "software quality engineer," please mentally substitute your own term for ours. We're using "tester" mostly because it's short, but also because it's an instantly recognized term among software teams. Usually followed by "Oh—*them*."

The Tester's Contribution, Illustrated

Almost all software teams include testers in some form. We want to talk a little about why this is so and then move on to how this plays out in XP. Consider the following story about Lisa's experience with a construction contractor:

> *When I decided to put an addition on my house, I wanted the basement extended as well. I signed a contract with my contractor that specified many details. I'm a pretty detail-oriented person, a quality assurance professional, and I thought I read this contract carefully.*
>
> *When the contractor built the new basement and cut a hole for the door into the old basement, he didn't install a door to close it off. When I asked why, he pointed out that I hadn't said I wanted one. I could access the new basement—it was functional—but had no way to close it off. Since the door was not included, I'd either have to do without or pay extra.*
>
> *Naturally, I had assumed the new basement would have a door I could open and shut. But since I hadn't specified it, the contractor hadn't included the cost of the door or the labor to install it in the contract. My contractor is terrific (call me if you live in Denver and need remodeling), but I couldn't expect him to just give me a free door.*
>
> *How nice it would have been if someone had looked at the contract and then said, "No door is specified here—don't you want one?" Then I could have decided whether to spend the money, and it wouldn't have been a less than pleasant surprise later.*

This remodeling project unfolded right before Lisa's eyes, so the omission of the door became apparent long before the project was complete. But with a software project, this wouldn't necessarily happen. In fact, for traditional software development, the "missing door" would probably not surface until the house was finished and the contractor had ridden off into the sunset.

The use of Extreme Programming would improve on this outcome, because the lack of a door would become obvious at the end of the iteration (short development cycle, usually two weeks), and the customer could choose to have it included in the next iteration. In fact, with XP, you could do even better: ask questions the first day of the iteration that would uncover the hidden assumption of the missing door.

As Lisa points out, what we really want as customers is for a builder to understand that when we specify a doorway in the requirements, we also expect a door to go in the doorway. This seems so blindly obvious to us that when a builder omits it, we might assume he's either stupid or dishonest.

The problem is that, as customers, our view of the proposed doorway is, first, something you can go through, and second, something you can close off. We get to the idea of a door in the doorway pretty quickly. But the contractor is thinking of the many other attributes of the proposed doorway, such as the size (30, 32, 34, 36 inches?), the shape (square or arched?), the moldings (profile, material, finish), the threshold and transition between floor materials, the framing required (especially in a bearing wall), the electrical and plumbing runs, location of light switches, and so on. The aspects the contractor is naturally going to pay the most attention to are the ones that affect construction, not the ones that affect the livability of the result. That's because the contractor "lives in" the remodeling process itself: this month our remodeling project, next month another, and so on.

This type of thing happens a lot in software development projects. As developers, we tend to focus on items that are important or interesting to the development process. How often have you been involved in projects that got the really hard part right but missed the easy stuff—which, unfortunately, turned out to be the very thing that mattered most to the customer?

This is where a tester can make the biggest contribution to a software project: by thinking about the system from the viewpoint of those who have to live with the solution but with the grasp of details, possibilities, and constraints of those building it.

In XP, for example, it would be someone in the role of tester who, during the Planning Game or while writing acceptance tests, would think of the obvious things one would do with a doorway and how to test if they're working.

This person would ask the customer, "How about a test for whether the room can be closed off?"

If the customer wants the door, the tester makes sure this was included in programmers' estimates. If not, the estimates are changed, and the customer may have to pick stories again for the iteration. The tester helps the customer write a test to make sure the door works as desired; then the team executes the test and reports its result. Before the end of the iteration, the customer knows not only that she needs to ask for the door but also that it's there and working properly.

In addition to providing this double-vision view, incorporating both programmer and customer sensibilities, the tester also has an outlook that differs from the programmer's in terms of the assumptions they make about correctness.

Programmers naturally assume that, in general, "things work." After all, even the simplest method or function involves the execution of millions of low-level microcode instructions or hardware circuits, and each one has to work flawlessly every single time or that piece of code could never work. As each method of an object is implemented and the objects are assembled into packages and applications and the applications into systems, which are then maintained, extended, and evolved, each piece in each level has to perform correctly every time for the whole system to work. Since programmers achieve this almost impossible-seeming level of reliability most of the time, that shapes their experience and world view.

Testers, on the other hand, are all descendants of Murphy and assume, in general, that things "don't work right." This is because testers live in a world where, no matter how many things are working right, some things always aren't. This is the same world, by the way, we all inhabit as customers. Our new electronically controlled, fuel-injected road machine may be a marvel of modern engineering, but if the cup holders are too small for a can of pop, that's what we notice. And even though 50 billion microcode instructions in our system may work flawlessly every second, if highlighted text is invisible because it's the same color as the background, that's what the customer will notice.

Shun the Dark Side

Just as we had a caveat for testers on the potential of XP to solve quality assurance and testing problems, we also have one for XP teams on the potential of testers to help XP projects. The tester role we advocate in this book is not the traditional quality assurance/testing role. It follows a much more integrated,

team-oriented approach and requires adapting one's skills and experience to function in an XP environment.

Just as you wouldn't want to be in the tester role on a team that claims to be XP but doesn't follow the practices (see "Wolves in Sheep's Clothing" in Chapter 1), you also wouldn't want to have a tester on your XP team who doesn't understand and buy into the methodology. A tester who views programmers as adversaries or doesn't believe in test automation or expects to spend more time planning than doing is likely to cling to techniques honed on traditional software projects. This will, in the best case, fail to help and, in the worst case, undermine the rest of the team's adherence to XP practices. Find a tester who understands and is committed to XP.

Summary

- We define a "tester" as someone who not only tests but who has QA and development skills.
- Our use of the term "tester" involves both QA and QC activities, and we drop the distinction between the two and encompass both in the integrated testing activities of XP.
- A tale of remodeling in which the contractor did not appreciate how the presence of a doorway in the plans should imply a door illustrates how customers' assumptions get missed in the development process.
- A tester makes a contribution by viewing the system from the standpoint of the customer but with the grasp of details, possibilities, and constraints of the developer, avoiding the "missing door" syndrome.
- A tester provides a skeptical outlook on the correctness of the system that helps keep the team from overlooking things that will make the customer unhappy.