JAMES BACH ON

Explaining Testing to THEM

Helping non-testers understand and support your work

by James Bach

Don't you love it when a programmer or manager makes ignorant statements about testing? You don't? Well, I do. At least they're talking. My experience is that most of my non-tester colleagues, no matter how smart and talented they are in their own work, are pretty mixed-up about *my* line of work. But if they don't say anything, there's not a lot I can do about it. So, in a way, I feel better when I hear one of them say something like "You play with each feature and see if it works, right? What's the big deal?"

Because if they talk, maybe they'll listen. If they listen, maybe I can offer them a more useful view of testing.

Maybe you think your co-workers should already understand how testing works. I feel your pain. Now, get over it. You are the testing specialist. You *like* this stuff. Because you're good at it, the other project roles can focus better on what they do well. Take their confusion as evidence that you are needed.

In testing, it's *us* versus *them.* Well, not really, but you may feel that way sometimes. Good explanations help bring the team together. This is important, because the rest of the project team, including managers all the way up the chain, won't fully support your work unless they understand what you're trying to do.

It Starts with Intent and Attitude

I suggest that you approach every explaining situation with the same intent: to make your clients or co-workers more powerful and successful, to help them make more effective decisions, and to help them know how to get the most out of your work. Let that intent shine in your tone.

One way of granting power is to offer valuable information. If you explain the dynamics of a test process to a project team, then the team can use that information to "work the system" and get more of what they want from you. For instance, if the programmers understand the benefits of testability, they are more likely to design the product in ways that allow you to compress the testing schedule.

However, knowing about testing is not the only way you help your clients be more powerful. You also help them by *not knowing* and thus sharing the problem-solving process with them. In other words, an alternative to "I know what to do, listen while I explain it" is "I'm not sure what the best test strategy would be. Let's discuss our ideas and try something that seems reasonable." The latter is the attitude of a student of the craft, rather than an expert. A true student studies, wonders, listens, and works with other students to piece things together. The posture of a student conveys more of a sense of speculation

QUICK LOOK

- Good responses to common questions
- 9 basic principles of good hallway explanations

than a tone of lecture. Oddly enough, the more expert I become in the logic of testing, the more I believe that what I know is primarily speculation.

Being a student requires that you come to terms with this basic truth: you might be wrong. It's because you know you could be wrong that your voice doesn't sound icy and condescending when you speak. You sound confident, but not insufferable. Because you know you could be wrong, you keep the smirk out of your voice when you say things like "It would be great if I could write a program that would automatically test this product; I just don't know how to do that."

The Hallway Dialogue

Explaining testing is a big subject. Here, let's focus on one part of the challenge: the hallway explanation. That's the name I give to the kind of explaining opportunity that comes up in the natural course of a software project, such as in a workday hallway encounter.

I'm walking to my cube. Adam, the development manager, is going the opposite direction. He spots me.

ADAM: "Oh, James, we want to move the schedule in by three weeks. I know your schedule calls for eight full weeks of testing after code freeze. Can you do it in five? We may just not have the time to test as much as we'd like."

My first thoughts are that the guy isn't serious about quality. He's a jerk and I should slap him silly. Heck, I'll quit. I don't need this… These first thoughts are not the helpful ones. I let them scroll by. A millisecond later, a useful thought comes to mind: perhaps Adam thinks that the test schedule arises from factors that are fully within my control. If so, perhaps I can set him straight.

JAMES: "My schedule isn't under my control, Adam. Eight weeks is just an estimate based on the complexity of the product and the difficulties we think we'll have once it becomes testable. It may require more than eight weeks to test and fix. Or less. A lot depends on the state of the technology when we get it."

Notice how I try to provide a short menu of factors that influence the schedule, so we can productively work toward as short a test cycle as is reasonable. I'm hoping he will ask about these factors, in which case I'll find a whiteboard and make a longer list for him. These factors don't have to be exactly right, just good enough so we can talk about the whole situation that we face, rather than just the outcome preferred by this one manager.

ADAM: "You can't predict the schedule?"

Again my first thoughts seem unhelpful: Maybe I'm a fraud. Maybe I should be able to predict schedules. Maybe everyone can do this but me. If only I hadn't been sick that day in high school, I would know about scheduling… These insecurities are normal for those of us who work hard to be good at our jobs. I let these thoughts scroll by, too.

What would be an effective response here? It seems to me that Adam is looking for stark binary answers (and maybe psychic ones) in a situation that is too complex and shaded for that. As in my first response, I'll try to answer in a way that expands the dimensions of the issue so that he and I can talk more productively. I will also deploy a potent tool: an example.

JAMES: "I don't know how to *precisely* predict the schedule. This work can go fast or slow. It depends. On the Alaska project, bug 2642—remember that one?—took two weeks before we found a fully reproducible case. It turned out that the product was interacting with a popular anti-virus scanner. You know you were glad to have that one fixed before we shipped, but we couldn't have predicted such a thing in advance."

ADAM: "I realize it's hard to estimate. But is it *possible* to squeeze the work into five weeks?"

Now I'm wondering why Adam is fixated on this time frame. He doesn't seem to be listening to me. If I answer the question with an explanation, he'll just get annoyed. The trick with hallway explanations is knowing when to lecture and when to shut up. In this case, it's time to listen and empathize.

JAMES: "I see that bringing in the schedule is important to you. Help me understand the significance of five weeks. What's the deal, there?"

ADAM: "Well, during the early planning, some senior managers got their dates mixed up. All this time we've been thinking that the 'Release To Manufacturing' date was June 30. Now it turns out that's the revenue date, and RTM has to be at least three weeks earlier in order to get the product into the pipeline."

JAMES: "What if the product just isn't ready for RTM at that time?"

ADAM: "It has to be."

JAMES: "What if it isn't?"

My aim with these questions is to clarify the situation so we can deal with reality and desire separately. Then I can show that there is not one choice here, but many choices. I do this not only to be helpful; discovering and clarifying possibilities is also fundamental to the testing craft, and every conversation is an opportunity to demonstrate the way testers think.

ADAM: "The VP isn't going to like that."

JAMES: "Well, that may be. But as a tester, my job is to provide information that helps the organization make better decisions. I see more than one option here. When it comes down to the wire, the VP may prefer a late product to a bad product. Or maybe he'd rather that we strip some features."

ADAM: "Why can't we modify our strategy and get everything we want? You said yourself that testing might take less than eight weeks. I'm looking for ways to tighten up the whole process. Work with me here."

Now he sounds like he wants to engage. He has implicitly accepted the idea that there's more than one choice and uses that to suggest his pet plan as one of the choices. This

may sound like he's co-opting the argument, but the truth is that it has co-opted him. By accepting that there are choices, he is now obliged to consider the factors that affect our choices, which is what I need him to do if he's going to understand testing.

JAMES: "Okay, let's work on this. First, I hope you understand that the testing schedule is not under my sole control. When our quality standard is very high, we have to do more careful testing. If Development delivers a product that is highly unstable, some of our testing may be blocked. If Development delivers a product that's hard for us to learn, hard to diagnose, or hard to control, then testing will crawl along very slowly. If the bugs we do find are elusive and intermittent, our investigation and reporting will take much longer. If the changes to the product are not managed well enough, we may have to do extensive re-testing. And if the programmers take a long time to fix the bugs, they may not be ready to ship on time no matter what testing remains to be done. Do you see what I'm saying, Adam? If you want to pull the schedule in, then we have to look at what drives the schedule."

ADAM: "I see what you mean. What can I do to help? Would it help if the programmers helped you test? If we ran some of your test cases?"

JAMES: "We don't have defined test cases."

ADAM: "Oh really? But wouldn't testing be more organized if you created test cases in advance? Wouldn't things go more quickly that way?"

Uh-oh. Now I have to deal with beliefs about testing coming from a non-tester. This triggers another set of unhelpful defensive thoughts: Your specifications are a mess, yet you expect us to write museum-quality test case documents? Gimme a break… Unless I'm really tired or I've just heard that I'm being targeted for a tax audit, I usually let those thoughts go, too. Instead, I try to form the most sympathetic interpretation of his words: He's right; sometimes it's possible to define good test cases in advance, and it's no crime to hope that testers could always do that, regardless of the circumstances.

JAMES: "Yeah, you'd think so. Sometimes that's exactly the right thing to do. In our case, though, I don't know how to make that work. There's too much uncertainty. We could create test cases, but they would be bad ones. People who define tests substantially in advance of having a product to test are either doing good work based on a very stable, well-defined technology, or they're doing bad work and bluffing you. Our challenge is to create the specific tests as soon as we can do a good job creating them, and no sooner."

The trick with hallway explanations is knowing when to lecture and when to shut up.

ADAM: "Then how do you control the testing? Do you have a test plan?"

It's a common misconception that a process can only by controlled by documented plans. This is my opportunity to introduce a different way of thinking.

JAMES: "For the most part we use an exploratory, risk-based test methodology. If by 'test plan' you mean a set of ideas that guide the test process, yes, I have that. It's not written down except as notes in my notebook, but I can share it with you if you want to hear it. Actually, I was hoping you could tell me if we're on the right track with our plan. We control the testing by frequently reporting our test status and adjusting our test strategy based on the joint sense, among the managers, of the product risks. In other words, as a client of our test process, I want you to participate in controlling how we test."

Words like "exploratory" and "risk-based" are buzzword bait. I'm hoping to provoke him into asking more about how we test. But this can be a risky maneuver. If I lay it on too thick, I will give the impression of a squid inking the waters.

ADAM: "What's an exploratory test methodology?"

He took the bait! He might suspect that I'm bluffing. Good for him. He's paying attention. Now I want to wrap up the explanation and end with a strong suggestion for what he can do that will get him what he wants.

JAMES: "It's like playing 'twenty questions' with the product. We test in a series of bursts. We simultaneously learn about the product, design tests, execute them, and report bugs. Our test coverage is based on a product model that we improve as we go. We also use a list of test heuristics—and I can show them to you right now if you want. It's the fastest way I know to test a new and unfamiliar product. Do you want to make it go even faster? Then let's start getting the testers up to speed on what this product is and how it works. Let's schedule an hour-long product briefing for the testers, with another hour of Q&A. Then we can start brainstorming more specific test strategies for use when you deliver the code to us in two weeks. How about that?"

ADAM: "When we will know how long the entire test process will take?"

JAMES: "Tell you what, Adam. Let's sit down right now and go over all the factors in detail: the risks, the various testing tasks, the development tasks—all that. Maybe we'll find ways to streamline the test process, but we should also develop some other options in case—despite our fondest hopes—the testing and fixing process drags on."

At this point it almost doesn't matter what he says. This is my last and best offer. I want him to get what he wants, and the price for that is to slog through the details. If he declines the meeting I propose, everything as I explained is still just as true. Ultimately, if the organization elects to force the testing to be "com-

pleted" in five weeks, I'll do what I can and faithfully report my status. We might be in good enough shape by then, or they might ship the product without knowing much about it.

This is a typical hallway explanation. An explanation in the wild. It may occur in a project meeting instead of a hallway, but the principle's the same. Note that it doesn't matter whether you agree or disagree with me about the role or utility of any of my ideas about testing. Insert your own ideas. What I'm illustrating here is the give and take of this kind of explanation, and the forces that shape it. The point is, we rarely get more than a few sentences to explain our work. That's why having some examples ready (like the sad story of bug 2642) or an analogy (such as the "twenty questions" game) is important. That's why it helps to practice reeling factors that influence testing off the top of your head.

There's no substitute for a lot of practice, but there are several classes out there that can help you build your dialogue skills. Two that helped me be a much more effective explainer were Gerald Weinberg's "Problem Solving Leadership Workshop" and the "Change Shop," which covered the whole spectrum of doing technical work in teams.

Principles of Hallway Explanations

There are nine basic principles of good hallway explanations:

1. Speak from practice. Be real. Own what you say. Avoid using someone else's explanations if you don't feel you fully understand them, because skeptical listeners will cross-examine you. Make it an ongoing project to improve your understanding of the testing craft—by watching how your ideas change when you put them into practice, and letting your ideas be influenced by how other people practice.

2. Be with your audience. Connect with them. Put yourself in their shoes. What does an executive vice president care about? What is a technical support manager's day like? What are a project manager's top three fears? Gear what you say to the ambitions, knowledge, and concerns of your listener.

3. Check your terms. Many times my explanations have been foiled by unexpected differences in terminology. Terms such as testing, test case, test plan, regression test, and bug are what I call *danger words*, because there are many different definitions of these terms floating around. If you find yourself in a debate, consider briefly defining your terms as you go.

4. Show respect. Treat everyone as if they are smart people who care about quality as much as you do. Honor objections and questions. I find that I can sometimes turn an otherwise hostile audience around by interpreting objections as well-intentioned, insightful concerns. It doesn't matter whether they *really are* well intentioned or insightful. I treat them that way because to do otherwise guarantees failure.

5. Provoke interesting questions. You can't productively explain something to a person who has no curiosity. Provoke your audience a little so they ask questions. One way to do this is to sow a few obvious ambiguities, jargon, or contradictions into your explanation. If your audience asks about them, say "Good eye. I'm glad you asked about that. That's an important issue." If they don't ask, then say "You've probably noticed that there's a contradiction here. You would be right to question it." Either way, go on to explain the issue better.

6. Explain things that matter. What operational difference do your ideas make? What do you want to have happen? How will we behave differently if we all accept your view of testing? Keep your eye on the mission behind your explanation.

7. Be quick. Get to the point. Use short sentences. Then maybe you'll be allowed to finish your explanation, instead of being cut off before the punch line.

8. Show how everyone is involved. I do my best to construe the situation as a team reality: we're all in this together. We all contribute. We all suffer when things go wrong. If you find it difficult to construe the situation as one in which everyone plays a part, then stop and think about whether the subject is one you can successfully explain. If the situation involves only you, then you'll bore your audience. If it involves only them, you may sound like you're butting in where it's not your business.

9. Be prepared. Develop a variety of standard explanations and analogies in advance and record them somehow. Ideas for explaining testing occur to me most often when I'm actually doing testing, so I keep a notebook around just to capture those ideas. Now I have a pile of notebooks brimming with fragments of explanations.

What If They Don't Believe You?

First, don't worry too much. In the movie *Harvey*, Jimmy Stewart uttered the famous line: "In this world you must be oh-so smart, or oh-so pleasant. For years I was smart. I recommend pleasant." Sometimes you won't persuade. Your explanations will fail. When that occurs, you do have another resource: reality. So take deep breaths, be pleasant, and keep your eyes open. Events will unfold. The project will struggle, or it won't. Whatever happens, pay attention. Take notes about the decisions that are made and how events unfold.

In the end, the team will have a shared experience. People take lessons so much better from their own experience than from stories about strangers or treatises of abstract logic. And those shared experiences provide the basis for new and compelling ideas, the next time you have to explain testing to *them.* STQE

James Bach is the founder of Satisfice, Inc., a test training and consulting company. A pioneer in the emerging disciplines of Good Enough quality and exploratory testing, James specializes in expert testing under chaotic conditions. He can be reached at james@satisfice.com.

STQE magazine is produced by STQE Publishing, a division of Software Quality Engineering.