

Blog: Why We Do Scenario Testing

From Michael Bolton at Developsense.

Original post at <http://www.developsense.com/blog/2010/05/why-we-do-scenario-testing/>

Last night I booked a hotel room using a Web-based discount travel service. The service's particular shtick is that, in exchange for a heavy discount, you don't get to know the name of the airline, hotel, or car company until you pay for the reservation. (Apparently the vendors are loath to admit that they're offering these huge discounts—until they've received the cash; then they're okay with the secret getting out.) When you're booking a hotel, the service reveals the general location and the amenities. I made a choice that looked reasonable to me, and charged it to my credit card.

I had screwed up. When I got the confirmation, I noticed that I had booked for one night, when I should have booked for two. I wanted to extend my stay, but when I went back to the Web page, I couldn't be sure that I was booking the same hotel. The names of the hotels are hidden, and I knew that the rates might change from night to night. One can obtain clues by looking at the amenities and the general location of the hotels, but I wanted to be sure. So instead of booking online, I called the travel service's 1-800 number.

Jim answered the phone sympathetically. It turns out that not even the employees of the service can see the hotel name before a booking is made. However, this was a familiar problem to him, so it seemed, and he told me that he'd match the hotel by location and amenities, back out the first credit-card transaction for one night, and charge me for a new transaction of two nights. He managed to book the same hotel. So far so good.

I went to the hotel and checked in. The woman behind the counter asked for identification and a credit card for extras, and then she asked me, "How many keys will you be needing tonight, sir?" "Just one", I said. She put a single key card into the electronic key programming machine, and handed the card to me. I took the elevator to room 761, which had a comfortable bed and desk with a window behind it, including a nice view. I went up to my room, unpacked some of my things, and decided to go for a dip in the hot tub. When I came back upstairs, I changed into dry clothes, took out my laptop, plugged it in, and sat down at the desk.

The floor was shaking. I mean, it was really vibrating. Some big motor—an air-conditioning compressor? a water pump?—had turned the office chair into a massager. I stood up, and it seemed that half of the room, including the bed was shaking. I tried to do a little work, but the vibration was enormously distracting. I called down to the front desk.

Peter answered the phone sympathetically. "I'll send someone right up to check it out," he said. Fair enough, but this problem was unlikely to go away any time soon, and until it did, I wanted another room. "No worries," said Peter. "I'll start the process now, and send someone up to check out the problem. Then you can come downstairs to exchange your

key.” (“Why not send the new key up with the person coming upstairs?” I thought, but I didn’t say anything.) “I’ll need a few minutes to tidy up,” I said. “Very well, sir,” said Peter. I repacked my bags. A few minutes later, the phone rang, and Peter asked if I was ready for the staff member to arrive. Yes.

After a short time, someone knocked on the door. He had a pair of new keys (two, not one), which he passed to me. He appeared skeptical at first, but I sat him down in the desk chair. “Oh, now I feel it,” he said. “Stand over here, next to the bed” I said. He got up, moved over, and felt the shaking. “Wow,” he said. We chatted for a few more moments, speculating on where the shaking was coming from. He left to investigate, and I decamped to my new room, 1021, on another floor on the other side of the building. So far so good.

This morning on my way to the shower, I noticed that a piece of paper had been slipped under the door. It was the checkout statement for my stay, noting my arrival and departure date and the various charges had been made to my credit card, including state sales tax, county tax, and a service fee for Internet use. I noticed that the checkout date and time was this morning, but I’m not supposed to be leaving until tomorrow morning. I called the front desk.

Zhong-li answered the phone sympathetically. I explained the situation, noting that I had booked through a travel service twice, once for one night and then later for two, and that the first booking should have been backed out (but maybe the service hadn’t done that), plus I had changed rooms the night before, so maybe it was an issue with the service but maybe it was an issue with the hotel’s own system too. Or maybe it was only the hotel. “No problem,” he said. “We can extend your stay for another night. But you’ll have to come downstairs at some point today so that we can re-author your room keys.”

So here’s the thing: how many variables can you see here? How many interconnected systems? How many different hardware platforms are involved? What protocols do they use to communicate? To create, read, update, and delete? What are the overall transactions here? What are the atomic elements of each one? How does each transaction influence others? How is each influenced by others? What are the chances that everything is going to work right, and that I will neither under nor overpay? What are the chances that the travel service will overpay (or underpay) the hotel for my stay, even if my credit card shows the appropriate entries and reversals?

It’s not even a terribly complicated story, but look at how many subtleties there are to the scenario. Have you ever seen a user story that has the richness and complexity of even this relatively simple little story? And yet, if we pay attention, aren’t there lots of stories like mine every day? Does my story, long as it is, include everything that we’d need to program or test the scenario? Does the card below include everything?

Next question: if you want to create automated acceptance tests, do you want a scenario like this to be static, using record and playback to lock in on checking specific values in specific fields? Are we really going to get value from the story if we use the same data and the same outputs over and over again? This approach will be hard enough to

program, but it will tend to be very brittle, resistant to change and variation. It will tend to miss details in the scenario that we would only learn about through repeated human interaction with the product.

Or would you prefer to have a flexible framework that allows you to explore and vary the scenario, designing and acting upon new test ideas, and observing the flow of each piece of data through each interconnected system? Might you be able to do this by exploiting testing tools that you've developed for the lower levels of the system and assembling them into progressively more powerful suites? This second approach will likely be even harder to program, although you might be able to take advantage of lower-level test APIs, probes, and data generators that you and the programmers have developed as you've gone along. This approach, though, will tend to be far more powerful and robust to change, to learning, and to incorporating new and varied test ideas. Think well, and choose wisely.

In either case, unless you have people exploring and interacting with the product and the story directly, I guarantee you will miss important points in the story and you'll miss important problems in the product. Your tools, as helpful as they are, won't ever pause and say, "What if...?" or "I wonder..." or "That's funny..." You'll need people to exercise skill, judgment, imagination, and interaction with the system, not in a linear set of prescribed steps but in a thoughtful, inventive, risk-focused, and variable set of interactions.

In either case, you'll also have a choice as to how to account for what you're doing. It's one scenario, but is it only one test? Is it dozens of tests? Thousands? If you use the second framework and induce variation, what does that do for your test count? Or would it be better to report your work in an entirely different way, reporting on risks and test ideas and test activities, rather than try to quantify a complex intellectual interaction by using meaningless, quantitatively invalid units like "test cases" or "test steps"?

It's been a while since I've posted this, but it's time to do it again. This passage comes from a book on programming and on testing, written by Herbert Leeds and Gerald M. Weinberg (Jerry wrote this passage, he says). It's understandable that people haven't got the point yet, since the book is relatively new: it came out only 49 years ago (in 1961). The emphases are mine.

"One of the lessons to be learned ... is that the sheer number of tests performed is of little significance in itself. Too often, the series of tests simply proves how good the computer is at doing the same things with different numbers. As in many instances, we are probably misled here by our experiences with people, whose inherent reliability on repetitive work is at best variable. With a computer program, however, the greater problem is to prove adaptability, something which is not trivial in human functions either. Consequently we must be sure that each test does some work not done by previous tests. To do this, we must struggle to develop a suspicious nature as well as a lively imagination."

Amen.

This entry was posted on Saturday, May 1st, 2010 at 11:29 am and is filed under [Acceptance Tests](#), [Coverage](#), [Scenario Testing](#), [Systems Thinking](#), [Testing Story](#). You can follow any responses to this entry through the [RSS 2.0](#) feed. You can [leave a response](#), or [trackback](#) from your own site.