Writing a Test Plan

Establish your hypotheses, methodologies, and expected results.

Dear KV,

We're getting ready for a project release at work, and since we're a small startup, all the developers have been asked to test the code of one of the other developers. We did this by lottery, each of us drawing a name from a hat (we were not allowed to draw our own name). It was an odd way to select testers, but it seems no worse than the processes I've seen at larger companies. The problem for me isn't that I have to write tests, but that I also have to write a test plan, one of the requirements imposed by our CEO, who is also the VP of engineering, aka my boss. I've never written an actual test plan, just collections of tests. Of course, I test my own code, but because I wrote the code, I know what I'm testing, and it has always been a straightforward process. Should I just write the tests and then list them in the plan? Somehow that doesn't seem to be what my boss is looking for.

A Man Without a Plan

Dear Planless,

Ah, a test plan, which can be an incredibly useful document or a massive time sink and distraction. Most good test plans start out as one-page documents, because what you must avoid is setting out to test everything--all at once. Instead of just trying to poke at various things that you think you need to test, you need to have a plan of attack as to what and how to test your colleague's code.

A good test plan is a lot like the lab reports some of us had to write for high school science classes. You won't use the word *hypothesis*, but each test is basically testing one. The plan should start with an outline so that you know you're covering the basics and the main thrust of the code. In place of a hypothesis, you have a statement about what you expect the code to do: "Given input X, we expect to see output Y."

Of course, it's not enough to have just a hypothesis; you have to say how you're going to prove or disprove the hypothesis. What is your test method? Do not answer this with, "Run the code," because if you do, both your management and KV will be perfectly justified in hanging you out of the office window by your thumbs. I bet you didn't read your whole employment contract, did you? Go ask HR, thumb hanging is in there. I'll wait.

Now that you know you have to do more than "run the code," let's look at some more useful valid test methods. Describing the test inputs you intend to use is a good start. You don't need to

list every possible input, but you should describe the range or shape of what the inputs might be. For a networked system, you might describe the types of messages you'll use in your test: "We will send packets of between 64 and 1,500 bytes, with most messages being power-of-two size bytes and containing random bit patterns in their payload sections." That's the test input, but you also need to describe the test output. Again, taking a networked system as an example, you might say, "A correct test result is one where all messages are forwarded without any messages being dropped, lost, or corrupted."

If your test has special setup requirements, such as a particular configuration of software or hardware, these must also be included in the plan, probably under their own section marked "Configuration." At the present time, you're the one writing the plan and the tests and probably executing them, but in the future, it may not be you running the tests. All the assumptions that are in your head while writing the test plan must be sought out, cornered, beaten into submission, and then written down. A test plan that leaves out an important but obvious (to you, anyway) requirement is going to be a source of maddening frustration for the next person who tries to use it.

Two more items to note in the test plan are the framework you're using and where it will store its results. Unlike a lab report, your test plan doesn't need to contain the results of running the test, and, in fact, I would expect that the results would be stored somewhere by the test framework that you're using.

If you can think of each of your tests as an experiment with a hypothesis, a test methodology, and a test result, it should all fall into place rather than falling through the cracks.

George V. Neville-Neil, Communications of the ACM, February, 2019