Fire Tower Tests & The GRIM Test.

08 May 2020

Sometimes finding out why something is broken is a long and painful process. You might have to trawl through a tonne of data, logs or equipment. Filtering out what looks OK from what looks, suspect.

These laborious investigative tasks are in the back of our mind when we're asked to do a code review, test some new code or a pull request.

What people often forget is that finding out if something is broken is completely different from finding out why it's broken. Being quick and efficient at finding broken stuff often takes a different approach to the task of clarifying the causes. A failure to test efficiently is therefore often a failure in imagination, a difficulty in creating these new techniques.

Take fire towers, for example, large forests in places like Canada and the US used to have large networks of Fire towers. These steel structures literally towered over the neighbouring landscape, providing a perfect vantage point for observers to spot and locate fires.

The towers made no attempt to diagnose the cause, monitor each tree, enforce bans on open fires or dictate how people used the forest. They just made it really easy to spot and locate smoke.

> They’ve taken the worst aspects of old scripted manual tests and shoehorned them into code. …they made the movie just like the book, and it’s painful to watch.

They did not catch every fire, but when they did see smoke people knew for sure that something was burning. That’s something the local loggers and townspeople wanted to know about.

As software developers, we often get lost in the woods and try to check every tree for every kind of problem. Many teams take pride in the fact they can churn out large volumes of regression or integration tests for all their new features.

For some, normal has come to be the writing out hundreds of (formerly manually executed) test cases in their chosen programming language. Automated, job-done, next ticket! They’ve taken the worst aspects of old scripted manual tests and shoehorned them into code. in doing so they made the movie just like the book, and it’s painful to watch.

Another approach is to use software ‘fire tower’, relatively simple tests that can see that something is busted, at a glance. Like my Cribbage example in my last post (http://www.investigatingsoftware.co.uk/2020/04/dealing-with-bugs-using-impossible-tests.html), where it’s quicker and simpler to scan over a large number of inputs checking for the wrong result than it is to exhaustively check each possible correct combination.

Another example is the GRIM test. Developed by academics Heathers & Brown (https://peerj.com/preprints/2064v1/), it’s a quick and easy way to spot if some summary statistics are incorrect. The following video explains how it works:

In summary, it’s a useful way to check if a mean and sample size is consistent. It will highlight if they are definitely wrong in some way, but it can’t tell you if it’s correct. What’s clever is that you don’t have to see the raw data. If you have a mean of 100 integers, you don’t need to know those 100 numbers to know if there might be a problem. Like the fire tower, it won’t catch every fire, but if it spots a problem then there’s definitely a bug somewhere in there.

This could make your tests much quicker or allow them to work on existing data produced by other tests. For example, extracting data via a GUI can be slow. So we often have to simplify the tests to make them quick.

If instead of reading every raw data value to find the mean, you could check the average by using the sample size and the GRIM test. While you would still need to check that the results are generally correct, being able to check quickly if big data sets are broken could save you and your tests considerable time.

I’ve created a python package you can use (https://pypi.org/project/grim/) to incorporate the GRIM test into your Python test code. It’s capable of handling the full range of decimal rounding

methods available in Python 3, and can even return a summary of which types of rounding deliver a consistent result.