I'm here today to talk about Mutation Testing, which is a technique that not enough developers have heard of.

But to explain what it is and why it's useful I'm going to tell a story...

Imagine your code is a town, Codeville.

Most of the time things are fine. But occasionally one of the townspeople commits a crime. We call this a bug.

Back in the old days when there was a bug, we'd come along and clean up afterwards and maybe try to stop it happening again.

But cleaning up after the event is not very good. It would be better to catch the criminal early before real damage is done.

So we started writing unit tests to act as a kind of police force making sure our citizens are law abiding.

That's great. But I remember when I started writing unit tests someone came out with the comment

"But how do you know if your tests are any good? Are you going write tests for your tests? And tests for those tests? And so on?"

It's a bit of a silly comment, but it does have a grain of truth in it.

We don't want to have to employ a second police force to watch the first one and so on.

But we'd like some reassurance that our police force is actually doing some good.

I think this was probably the point where someone thought up test coverage.

So we began checking that our police force was patrolling all the parts of our town.

This can be helpful, especially if it highlights that there are dangerous parts of town where the police never go.

But you might still have a nagging worry that your police officers are patrolling the whole town but not really paying any attention and just sitting around eating doughnuts. You know what government targets are like!

That corresponds to tests with good coverage that don't actually test very much. Can we do better?

The best way to check that the police are going to catch criminals in the act is to stage some fake crimes and see if they respond.

That's what mutation testing does. It injects bugs called mutations into your code and looks for test failures.

So it'll swap “if” statement logic around or swap an addition for a subtraction. If the tests spot the bugs then we can be fairly sure they are doing their job.

Not only is your police force patrolling the streets, but it'll actually respond to crimes.

Great! So how do I run mutation testing over my code? Unfortunately this story does not have a “Happily Ever After”.

If you run Java, there's a decent framework called PIT. If you don't, then the support for doing this is pretty thin.

PIT uses coverage analysis and bytecode manipulation to make it run quite quickly. It's being actively developed and I suggest you give it a try.

If your code is in another language I suggest you see what's available and lend a hand to make it better. The situation is a little like it was with unit test frameworks 15 years ago. i.e. There were hardly any to speak of.

Learn more:
http://pitest.org
http://accu.org/var/uploads/journals/overload108.pdf

Me:
Twitter: @nespera
Email: chrisr@we7.com

Photo credits:

mutant badges: http://www.flickr.com/photos/ntr23
donut neon: http://www.flickr.com/photos/krapow
oakland cops: http://www.flickr.com/photos/thomashawk
salisbury sign: http://www.flickr.com/photos/chough
crime scene: http://www.flickr.com/photos/freefoto
sad face: http://www.flickr.com/photos/kalexanderson