Twitter Thread by GeePaw Hill

GeePaw Hill
@GeePawHill

Today, let's talk about microtest TDD's Judgment Premise: "We are absolutely and permanently reliant on individual humans using their individual judgment in TDD."

Folks, in these times, I gain respite in thinking & writing about geekery, and I hope you do, too. But there are more important stories, and this is just a break.

Stay safe. Stay strong. Stay angry. Stay kind.

Black lives matter.

Voting rights do, too.

The judgment premise emphasizes the human in test-driven development. There are no known non-humans practicing TDD, so it may seem a odd that we have to talk about this, and yet, we do.

As software geeks, we work with the most rigid deterministic systems conceivable, and we do much of that work in abstract mental space. To say that our target systems are machine-like is to say too little, really: they're more machine-like than any real-world machine.

The uber-mechanical material of our job can lead us to seeing *every* aspect of the job as if it were the same as that material, a phenomenon the French call "deformation professionelle". Put simply: working with uber-machines all day, we imagine uber-machines everywhere.

There have been and continue to be many many efforts to depict TDD as a kind of "machine for programming", an algorithm that runs deterministically and thereby makes software.

The ubiquitous TDD flow-charts are an example of this, and there are lots of others, too. (I'm not being mean here, btw, hell, I've made some of those flow-charts myself.)

But the little decision-diamonds on those flow charts, even the most seemingly accurate ones, don't begin to express the range and inherent complexity of human decision-making involved in successful TDD.

There are a fair number of Old Guard TDD'ers out there who write or talk or video their actual working process. When you see one of these folks at work, you start to notice: whatever they're doing it's not a mechanizable algorithm.

In my own Real Programming series, still in its early days, we're about five hours into building a rather simple desktop yahtzee game. And the range and reach of the judgment calls I've *already* made are just enormous.

I have decided -- hunch, experience, intuition, guesswork -- what to test. I have decided whether to test things I do test the hard way or the easy way. I have decided to create things just for testing. I have decided more or less arbitrarily when I have enough tests to proceed.

All this judgment-calling, and we're barely one full workday into coding, and it's a ridiculously tiny, simple, and useless application.

Compare that to the apps most of us work on, and magnify my judgment-quantity by many many orders of magnitude.

A for instance: one of our goals in writing tests at all is to gain confidence that our system works. But some of the greatest minds of the late 19th & early 20th c established beyond refutation that perfect confidence is not possible even in theory, let alone in practice.

We are left, then, to make judgments about how much confidence we need, where that need is most critical to us, how to best go about filling that need. And the judgments ripple out from there, all up and down and in and out of our software development process.

And every one of those judgments we make is non-computable. They are usually unparsed and unparseable mixtures of experience, insight, experiment, and intuition: all the stuff human minds do.

TDD'ers say "red-green-refactor" not to imply a judgment-less deterministic algorithm for rolling code, but to describe a broad schema -- a large-scale recurring pattern -- that can be seen in successfully rolling code using TDD.

That red-green-refactor leaves out a *lot*, and we know that, and we're okay with it. It still has value to us, for all that it's not describing a machine-like process.

The biggest things it leaves out? 1) Whether any given portion of our code will pay us back for TDD'ing it into existence, 2) How much advance work we do before writing a test, and 3) When have we achieved enough confidence to proceed.

All three of those decisions are incredibly nuanced, delicate, subtle judgment calls, and if we don't make them well, every day all the time, TDD just isn't going to function.

So. Here we are.

In TDD, we are permanently, inevitably, irremediably, and absolutely, dependent on individual humans using their individual human judgment.

And one more: "happily".

TDD is a human enterprise, not a mechanical one, and the Judgment Premise makes that explicit.

Thanks for hanging in. If you like my work, please subscribe. It's free, spam-free, comes in full text or audio, and it helps me. https://t.co/0iffwG5jrd

And change things, in the trade and out.

We can fix this. We're the only thing that can.

Black lives matter.