Test-Driven Development: Good or Bad?

Juni 03, 2014 Huib Schoots

> *The theme of this Agile Record is "test-driven development". TDD is a method of learning while writing clean, maintainable, highquality code. A topic I hear and read a lot about, but do not really know from experience. I have not encountered test-driven development in the projects I have been in. That might say something about the projects I have seen. Or maybe it is not used that much? As a test expert with a programming and test automation background, unit testing in general and TDD in particular interest me a lot!*

Some think of TDD as a programmer's practice, so it does not concern testers. But doesn't that sound a bit strange? Development is a team sport. My experience is that we tend to think in roles, but I would rather speak of skills. A perfect team has all the skills necessary to do the job. These skills are used to apply the methods and techniques that solve our problems in order to solve our clients' problems… If this is true, then TDD *does* concern me! When used, it is part of what a team does. That makes it part of my context and part of the testing we as a team do! I like to think that people with excellent testing skills need to help the people with coding skills do great unit testing.

Testing?

I like to make the distinction between testing and checking as described by James Bach and Michael Bolton in their blog post "Testing and Checking Refined" [1]. It helps me see that what is commonly called 'testing' has several aspects to it: "evaluating the product by learning through experimentation which includes to some degree: questioning, study, modeling, observation and inference". James and Michael call this 'testing'. What they call checking is "the process of making evaluations by applying algorithmic decision rules to specific observations of a product". They say that only checks can be automated! Please read Michael's post "On Testing and Checking Refined" [2] to understand why they wrote the post and why we need both testing and checking.

As said, I read about it a lot and, while trying to learn more about TDD, I found two opinions that kind of contradict each other and that I want to share.

Test-Driven Development is dead, long live testing

The first opinion is a rather angrily written blog entitled "TDD is dead, long live testing" [3] by David Heinemeier Hansson, the creator of Ruby on Rails, who tried TDD and who obviously does not like it. However, he does a poor job of explaining why he hates it and calls it 'fundamentalism'. I use this as an example of a context where TDD did not work out too well.

He thinks our industry suffers from a lack of automated regression testing and that test-first just showed this. He thinks of TDD as design dogma and he advocates regression testing with less emphasis on unit tests and more emphasis on (slow) system tests. I feel he is just frustrated that TDD did not work for him.

Test-Driven Development helps teams to get better

My developer/tester friend Markus Gärtner talks about TDD in this interview [4] in the "Disruptive Testing" series. He says several very interesting things: TDD helps to deliver better unit testing, but will not relieve teams from other testing. It also helps teams get better.

Markus also says that there is a huge difference once you experience the benefits, rather than hearing or reading about it. I think it makes a big difference when teams work together on unit testing and it will make teams better if they work on testing (including TDD) together as well.

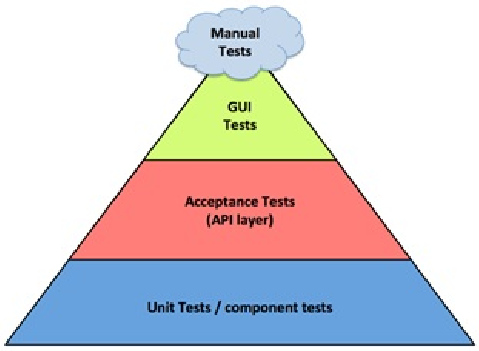
Reducing risks

We tend to think in test levels (or agile quadrants) too much. I like to think of testing in terms of collecting information and reducing risks. We should ask the question: “What do we need to do to reduce a (product) risk?” Therefore, we need an approach that concerns everything the team does. I think the team needs to know what everybody else in the team is doing.

For example, would it not be a good thing to have developers focus on check automation helped by testers to make their testing great? They pair, or a tester can review the unit test to see what is checked. If necessary the tester can add checks to get the coverage the team needs. Knowing what is checked will help the testing during the project! This could be the best of both worlds and everybody does what he or she likes to do: testers like solving puzzles and having an overview of what gets checked and tested, developers create the code to run the checks to create great code and do not have to worry about their checks being good enough. Together they create the best possible testing and checking! I guess TDD fits into this perfectly.

Testing pyramid

I like to talk about automation in testing using the “testing pyramid”. I try to use the pyramid as a heuristic. The story I tell is that we should automate checks as much as possible when there is a good reason to do so. That reason is often speed and ease. I wrote about “keeping up” in Agile Record no. 14 [5]. There can be many different reasons why you might want this as long as it solves the problem. Automating your checks will create more time for testing.



I have seen many teams struggle with test automation trying to automate stuff at a GUI level. While there are no or very few automated checks at a lower level, they put a lot of effort into checking at the GUI level. Checking at the GUI level is hard to do, since objects defining the GUI change all the time, it is often very slow compared to unit checks, and it needs much more resource to check enough – think of environments, data, etc. The pyramid reminds us that it might be more valuable to have unit checks rather than GUI checks. Since it is a heuristic, it may not be true in some situations! The testing part in this is the learning, the thinking about how to test the product: doing manual testing, manual checking, and automated checking.

Funny names?

To me, TDD has several other interesting aspects to it: we call it testing but it is actually checking. As said before, I think using testing and checking is very useful when talking about testing in general, and in a discussion about ‘test automation’ in particular. Think of it like this: unit testing with TDD should be thinking and learning about the product using the automated checks. I know very few developers who like testing. They often think of it as a necessary evil. So why is it called TEST-driven development? TDD is not about the check created, it is about learning about the design. It is about understanding what needs to be built. It is primarily a specification technique that enables understanding and reduces thinking errors. Secondarily it helps finding coding errors.

Critical thinking and problem solving

Automated checking, unit testing, TDD: they do not replace thinking. Every team has its own problems to be faced, and different solutions to provide. I do not believe that one technique solves all problems. Especially not if this technique is used in the same way everywhere, like a recipe. The value of any practice depends on its context!

Uncle Bob wrote a response on the "TDD is dead" post called "Monogamous TDD" [6] – and it makes sense to me. But how can I really and deeply understand whether what he says makes sense?

I think I have to talk to some experts who really tried this and go deep into their context, the problems they were facing, and how TDD solved these. Writing this column has certainly helped me think about it critically.

Finally

I guess I have more thinking to do. Markus Gärtner replied to me on Skype when I asked him about the blog post: "Everything can be a problem if you are faced with incompetence. While learning Test-driven development, I also went through a phase where I wrote terrible tests. I have learned from that, and am now able to write better tests that don't become a nightmare in the long run."

After writing my column Martin Fowler, Kent Beck and David Heinemeier Hansson started doing hangouts to discuss TDD is dead. You can have a look on the following video:

References:

- [1] http://www.satisfice.com/blog/archives/856
- [2] http://www.developsense.com/blog/2013/03/testing-and-checking-redefined/
- [3] http://david.heinemeierhansson.com/2014/tdd-is-dead-long-live-testing.html
- [4] http://www.thoughtworks.com/insights/blog/disruptive-testing-part-3-markus-gartner
- [5] Agile Record #14
- [6] http://blog.8thlight.com/uncle-bob/2014/04/25/MonogamousTDD.html

Huib Schoots

Huib Schoots is a tester, consultant, and people lover. He shares his passion for testing through coaching, training, and giving presentations on a variety of test subjects. With fifteen years of experience in IT and software testing, Huib is experienced in different testing roles....

Read more about Huib Schoots

AGILE TESTING DAYS
EUROPE´S GREATEST AGILE SOFTWARE TESTING EVENT!
November 09–12, 2015, Potsdam, Germany
TICKETS LIMITED
"Embracing Agile for a Competitive Edge - Establish leadership by delivering early, rapid & iterative application releases"

Newsletter

Stay up-to-date with the best articles written by your peers!

SUBSCRIBE NOW

Newsletter

Stay up-to-date with the best articles written by your peers!

SUBSCRIBE

Recent Arcticles

By the way…
Oktober 23, 2014

Managing Technical Debt with Agile
September 25, 2014

Scrumban – Being Differently Agile