Test-Driven Development Exposed

Growing Complex Software One Test at a Time

RBCS

TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Introduction

- Software engineering has seen its share of controversies, fads, and techniques recently
- One technique that has stirred controversy and been called a fad is test-driven development (TDD)
- While it shares a lot of letters, it's not the same as ATDD or BDD
- TDD is not where testers tell the developers what to do
- TDD might not really even be testing, at least not as testers typically mean that word
- While it came from Agile, it's not specific to Agile
- Let's look more closely at this oft-discussed, sometimes-misunderstood technique…

Test-driven Development

- A development technique (test-first)
  - Add test for small piece of code
  - Run the test (which should fail)
  - Write code until test passes
  - Refactor code
  - Repeat process (previous tests and added tests)
- Used in Agile and sequential lifecycles
- Fix code bugs on introduction
- TDD provides executable design specifications, as automated tests, in continuous integration

Acceptance Test-driven Development

- Define acceptance criteria and tests early in development
- Collaborate so every stakeholder understands behavior
- Process:
  - Define tests for intended behavior
  - Create automated acceptance tests
  - Program intended behavior
  - Run automated acceptance tests
- Create reusable regression tests for continuous integration
- Test data and service layers, too
- Test system/acceptance level in appropriate environments
  - Identify/quickly resolve defects
  - Verify feature behavior
  - Measure acceptance criteria
  - Deliver to external testing teams

Behavior-driven Development

- Behavior-driven development is a black-box approach
- Focus on expected behavior:
  - Developer creates a test for the class under development
  - Tests should make sense to stakeholders (including testers)
  - Clarify where defect lies (code, user story, or test)
- Behavior-driven development defines tests:
  - Given some context
  - When some event occurs
  - Then the system should produce some outcome(s)
- Helps to define test cases for developer based on tester/stakeholder collaboration

TDD Process

- If first starting, set up unit test framework
- Design the module by defining the steps (and thus tests) making up the task
- For each step
  - Create one new unit test
  - Run the new unit test to confirm it fails
  - Write only code needed to make the new test pass
  - Refactor the code until you're happy with it
  - Ensure that all unit tests continue to pass
  - Move on to the next unit test
- Check code and automated tests into continuous integration framework

What Some People Mean by TDD

- When some people say they use TDD, they might mean:
  - "We have automated unit tests"
  - "We create unit tests after writing code"
  - "We have a continuous integration framework"
  - "We create unit tests unless the code is too tightly coupled" (excuse alert!)
  - "We'd like to use TDD, but there aren't any frameworks for our language" (excuse alert!)
  - "We don't really know what TDD is, but our developers say it's cool, so I'm sure we do it" (BS alert!)
- Often, pure TDD is not done
- I think the first three items are where the benefit is

Not Testing as We Know It, Jim

- The process leaves many open questions:
  - What are the test bases?
  - What are the test oracles?
  - Are any black-box and other techniques used?
  - Are positive *and* negative tests written?
  - Is any level of code coverage required?
  - Is static analysis used?
  - Is an effective code review required?
- Without good answers to these questions, TDD tests are executable design specifications, but not tests

What Can Testers Do in TDD?

- By involving testers, organizations can make TDD truly testing
- Skills required of testers:
  - Code literacy, in system language(s)
  - Knowledge of unit test tools
  - Ability to interpret static analysis results
  - Understanding of code coverage
  - Knowledge of the right test bases and oracles to use
  - Ability to organize and participate in code reviews
- The more skill in each area, the more the tester can contribute
- Testers should aim to complement (not duplicate) coverage in the tests they create

An Example of TDD in Action

- Building an e-commerce application
- One function needed: processing credit and debit card payments
- An online payment processing service will be used
- A programmer will use TDD to create a module to access this service
- Let's see the steps…

Creating the Module

- Write test, write code, refactor:
  1. Verify valid input parameters (probably multiple steps here)
  2. Establish a connection to the web service
  3. Transmit the parameters
  4. Handle successful transactions
  5. Handle unsuccessful transactions (probably multiple steps here)
- The tests *will* provide executable specifications for the code as it grows
- The tests *might* also truly test the code as it is written

TDD Controversy

- Boosters of TDD (i.e., Beck, Fowler, Martin) claim big things for TDD
  - "Re-factor code with zero regression risk"
  - "Clarify thinking before coding"
- Does TDD live up to the claims?
- I asked Beck to comment, but got no response
- I also ask TDD gadfly Jim Coplien to summarize his concerns…

Coplien's Take

- TDD is not a testing technique, it's a design technique
- It has the weaknesses of unit testing (see articles)
  - The author bias is a problem
  - It tests individual objects, not interfaces or emergent system behaviors
- As a design technique, it encourages the wrong perspective
  - Objects should reflect usage (top-down)
  - TDD encourages procedural design (bottom-up)
- TDD can create as much test code as delivered code
- Automated TDD tests give:
  - An ever-growing inventory of tests
  - A sense of false confidence
  - Pure verification; no validation
- Empirical research does not support Beck's claims for improved coupling and cohesion or for reduced bug density
- So, TDD has little or no benefit with very high cost

Conclusions

- TDD is part of the alphabet soup of agile techniques, along with ATDD and BDD
- TDD can mean different things to different people
- TDD is different from most other testing, and may not really be testing at all
- Technically skilled testers can contribute to TDD if it's used in their teams
- Controversies exist around TDD, which might lead to changes in the future

To Contact RBCS

For over twenty years, RBCS has delivered consulting, outsourcing and training services to clients, helping them with software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Phone:	+1 (830) 438-4830
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com
Twitter:	@RBCS, @LaikaTestDog
Facebook:	RBCS-Inc.