Test-Driven Development

Mark C. Paulk, Ph.D.

Mark.Paulk@utdallas.edu, *Mark.Paulk@ieee.org*
http://mark.paulk123.com/

Test-Driven Development in XP

"Test, then code" means a failed test case is an entry criterion for writing code.

- **aka Test-First Programming**

1) **write an automated test**
2) **write enough code to pass the test**
3) **refactor to improve readability and remove duplication**
4) **repeat**

The goal of TDD is clean code that works.

- **Ron Jeffries**

Test-First Programming

Writing a failing automated test before changing any code

- **Extreme Programming Primary Practice**
- ***K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, 2nd Edition, 2004.***

Addresses

- **scope creep**
- **coupling and cohesion**
  - **highly cohesive code is easy to test**
- **trust**
- **rhythm**
  - **test, code, refactor, …**

Martin's Three Laws of TDD

You may not write production code unless you've first written a failing unit test.

You may not write more of a unit test than is sufficient to fail.

You may not write more production code than is sufficient to make the failing unit test pass.

Following these laws perfectly doesn't always make sense… The goal isn't perfect adherence to the laws — it's to decrease the interval between writing tests and production code to a matter of minutes.

R.C. Martin, "Professionalism and Test-Driven Development," IEEE Software, May/June 2007.

Beck's Two Rules of TDD

Write new code only if an automated test has failed.

Eliminate duplication.

Consequences (*K. Beck, Test Driven Development, 2002*)

- **design organically with running code providing feedback between decisions**
- **write our own tests because we can't wait for someone else to**
- **environment must provide rapid response to small changes**
- **design must consist of many loosely coupled, highly cohesive components to make testing easy**

TDD Observations

TDD is functional testing, not white-box testing.

TDD demonstrates functionality by example.

TDD ties directly to the confirmation aspect of the user story’s card / conversation / confirmation.

Frequent testing (continuous integration, daily build and smoke test) provides timely feedback on usability, coupling, and sufficiency.

Importance of TDD

Test-driven development (TDD) is more important than both Scrum and XP.

- ***H. Kniberg, <u>Scrum and XP from the Trenches</u>, 2007.***

Test-first design helps the developers build better OO designs.

- ***J. Rasmusson, "Introducing XP Into Greenfield Projects: Lessons Learned," IEEE Software, May 2003.***

Corollaries of TDD

The unit tests that TDD causes us to write are much like the material in the back of a manual – each unit test is an isolated snippet of code that explains how some part of the system works.

TDD can’t force you to write clean code, but it can help eliminate the fear that cleaning your code will break it.

R.C. Martin, “Professionalism and Test-Driven Development,” IEEE Software, May/June 2007.

Assessing TDD

Reduces the gap between decision (design developed) and feedback (functionality and performance obtained by implementing that design).

- **fine-grained test-code cycle gives constant feedback to the developer**
- **identify and remove defects quickly**
- **determine the source of the defect easily**

Ensure testable requirements (user stories)

- **drives good understanding of requirements and up-front design**

Encourages writing testable code

Supports rich set of regression tests for continuous integration

Better unit testing, 50% reduction in defect density in testing

- ***E.M. Maximilien and L. Williams, "Assessing Test-Driven Development at IBM," ICSE, 2003.***

Testing Tools

X-unit test framework

- Kent Beck

- JUnit for Java
  - junit.org
  - JUnit tutorial: http://www.asjava.com/junit/junit-tutorials-and-example/
- CppUnit for C++
- httpUnit

FIT (Framework for Integrated Tests)

- Ward Cunningham

FitNesse = FIT + Ward's Wiki technology

- Robert Martin
- www.fitnesse.org

Questions and Answers