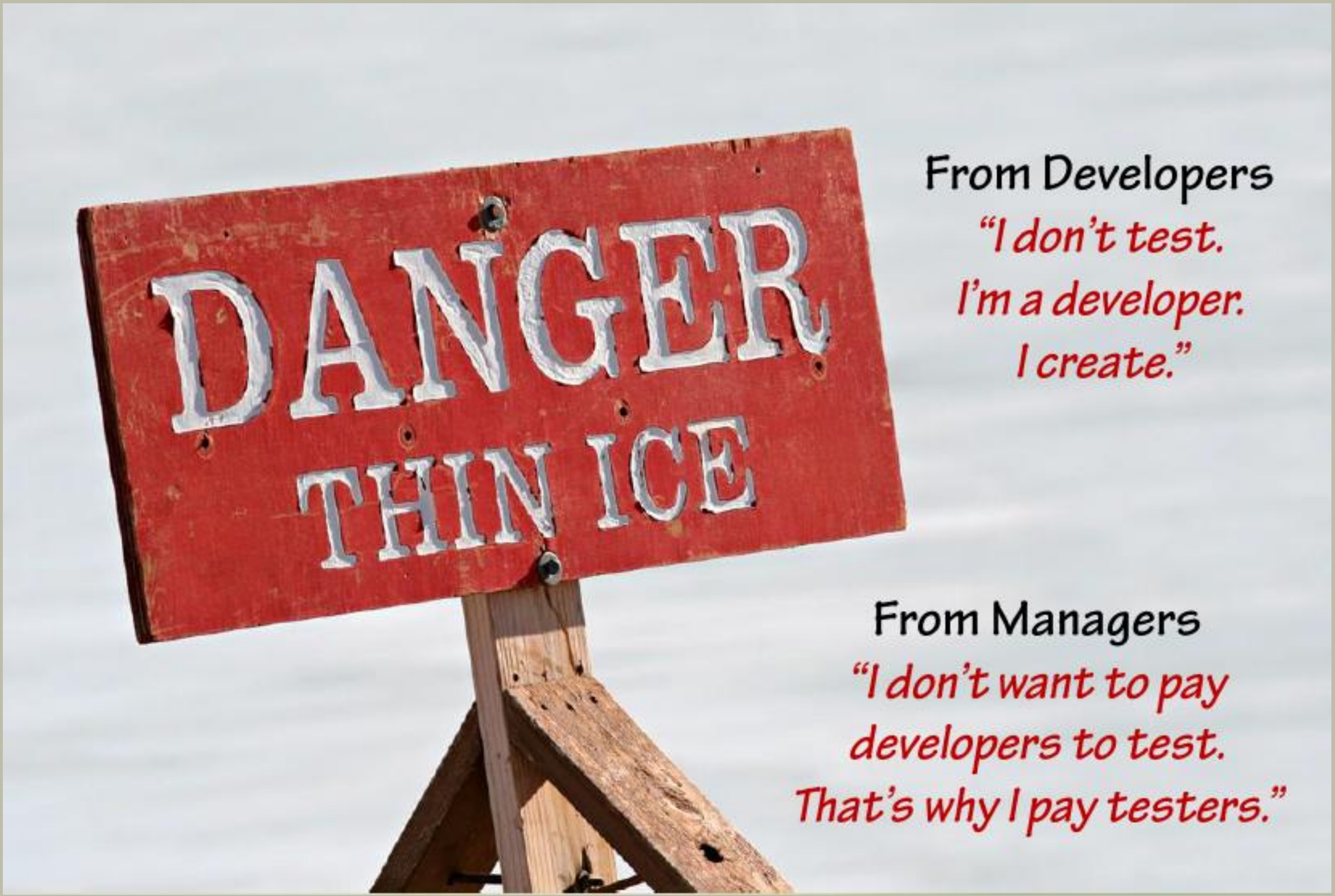
Test Driven Development

Test First Development in .NET

OdeToCode.com

In This Module

- What is it?
- What will it do for me?



From Developers

*"I don't test.
I'm a developer.
I create."*

From Managers

*"I don't want to pay
developers to test.
That's why I pay testers."*

Unit Tests

- Test a small but functional piece of code

```
public ActionResult Login(LoginModel model, string returnUrl)
{
   // …
}
```

It’s Something You’ve Already Done

- **Ever run in the debugger after the code compiles?**
- **Unit test framework**
  - Adds structure
  - Repeatability

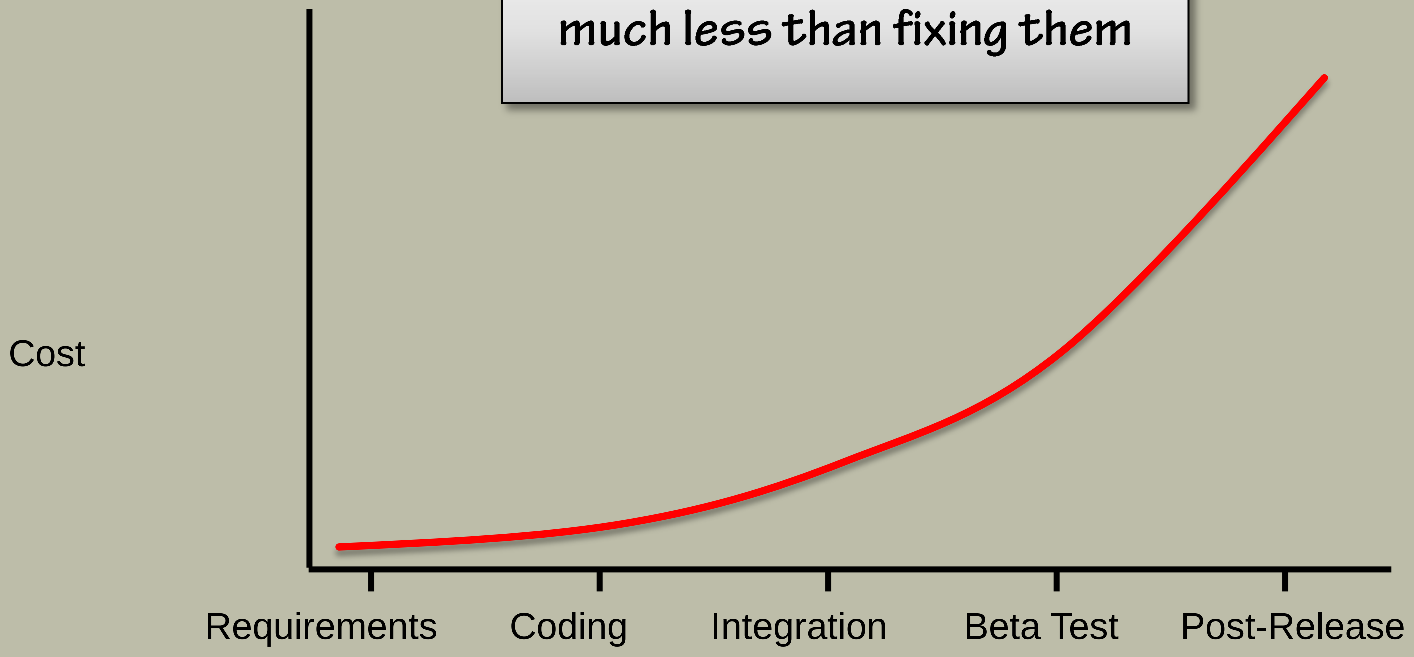
The Economics of Developer Testing

- **Realizing quality improvement through test driven development**
  - □ http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf
  - □ 40% – 90% decrease in defect density
  - □ 15% - 35% increase in development time

The cost of a released defect is usually far more than the cost of preventing it

Cost of Fixing a Defect

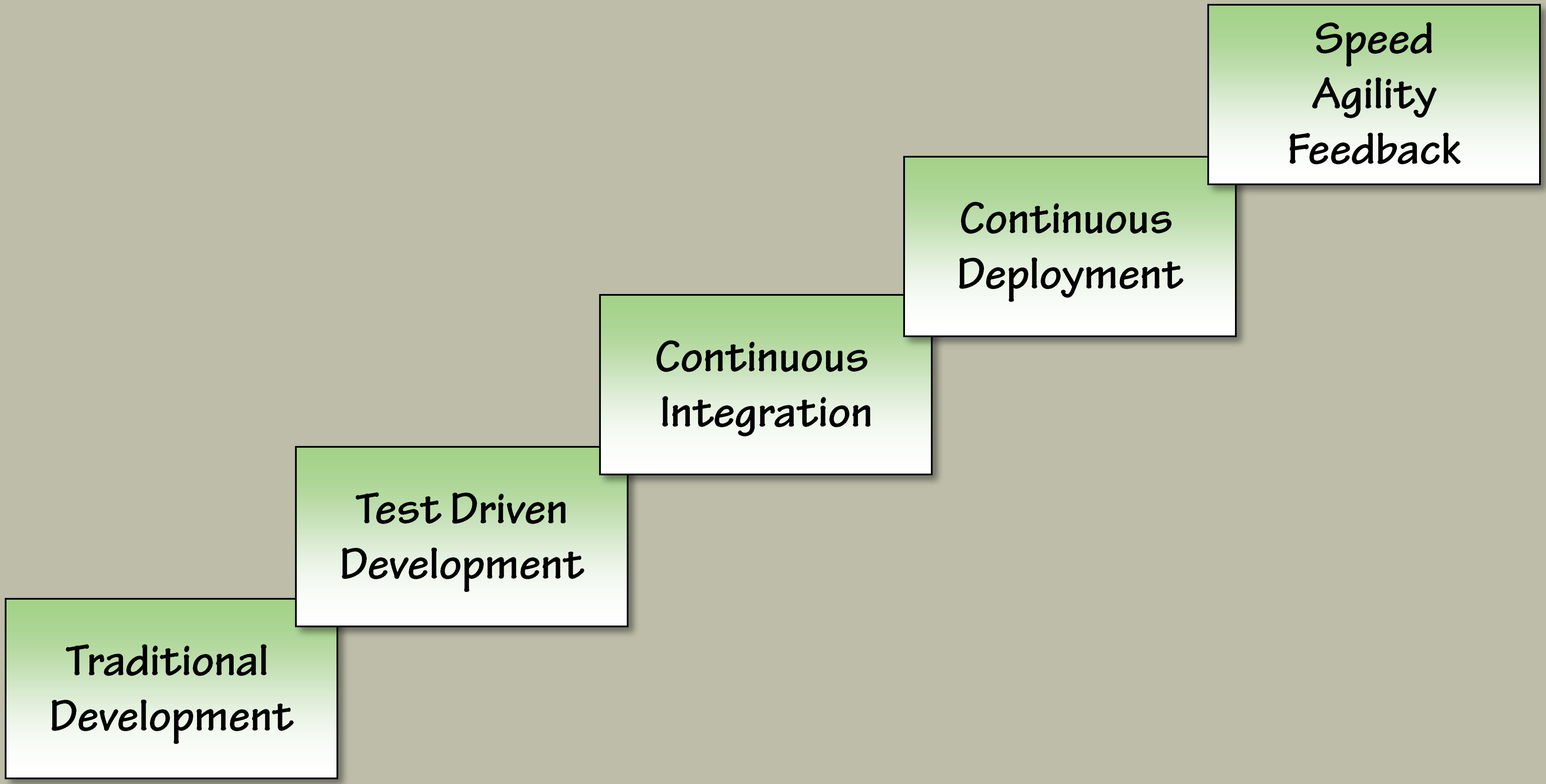
Avoiding defects costs much less than fixing them



Leap Years

- Microsoft Azure (2012) customers
- TomTom GPS failure (2012)
- PS3 network outage (2010)
- Scientific Atlanta (2004)
- Pontiac Grand Prix (2004)
- Norwegian State Railways (2000)
- Tiwai Pt Aluminum Smelter (1997)

Stairway To Heaven



Traditional Development

Test Driven Development

Continuous Integration

Continuous Deployment

Speed
Agility
Feedback

Extreme Continuous Deployment

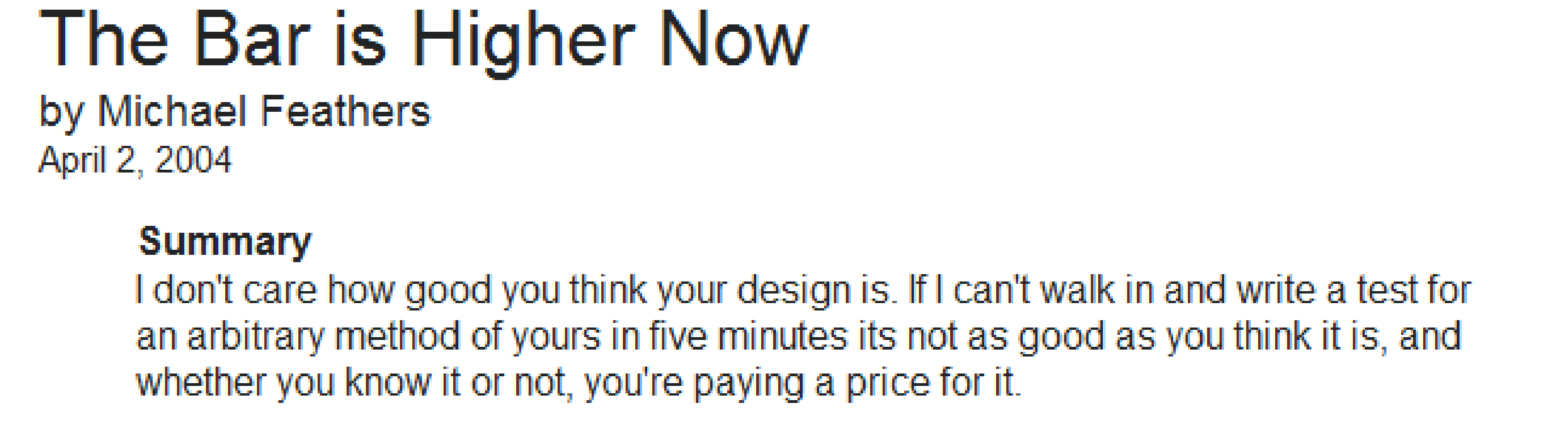
- **Amazon deploys functionality every 11.6 seconds**
  - 90% reduction in outage minutes
  - 0.0001% of deployments create an outage
- **http://code.flickr.com/**
- **http://www.etsy.com/**

TDD Makes Your Life Easier

- Easier to find bugs in software
- Easier to maintain and change software
- Easier to understand code
- Easier to develop new features

Thinking Like A Scientist

- Hypothesis
- Repeatable experiments
- Conclusions



The Bar is Higher Now

by Michael Feathers

April 2, 2004

Summary

I don't care how good you think your design is. If I can't walk in and write a test for an arbitrary method of yours in five minutes its not as good as you think it is, and whether you know it or not, you're paying a price for it.

TDD Isn't All About Testing

- **Confidence**
  - Confidence in the code
  - Confidence to change the code

Less Time Debugging

- TDD for Embedded C

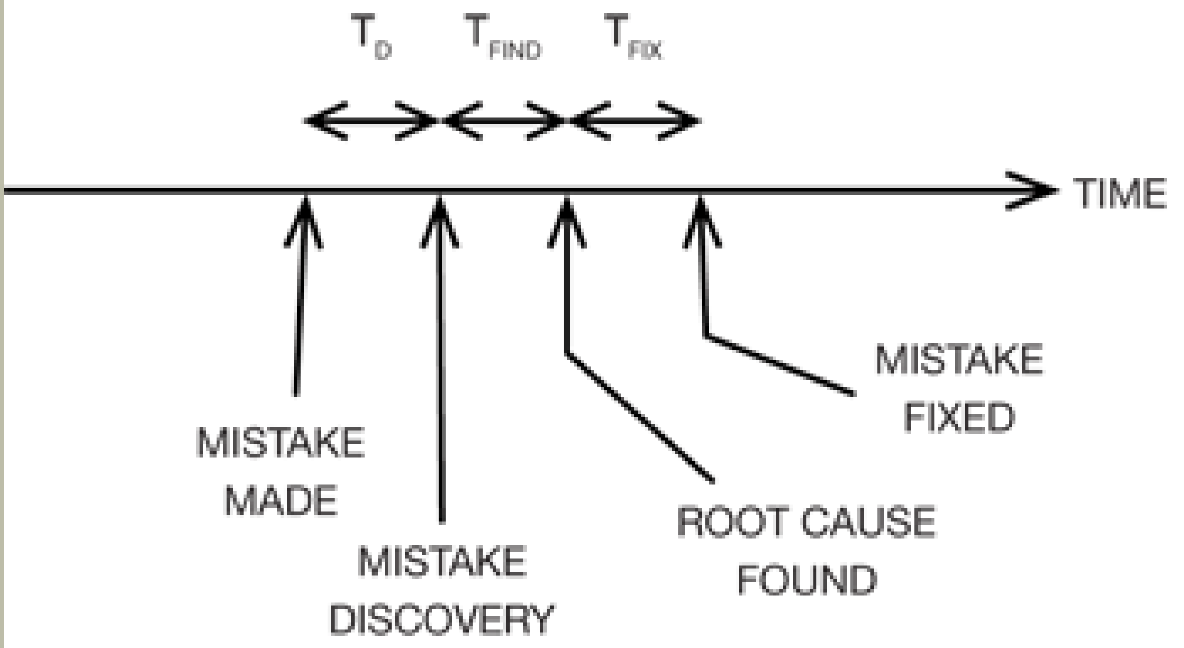
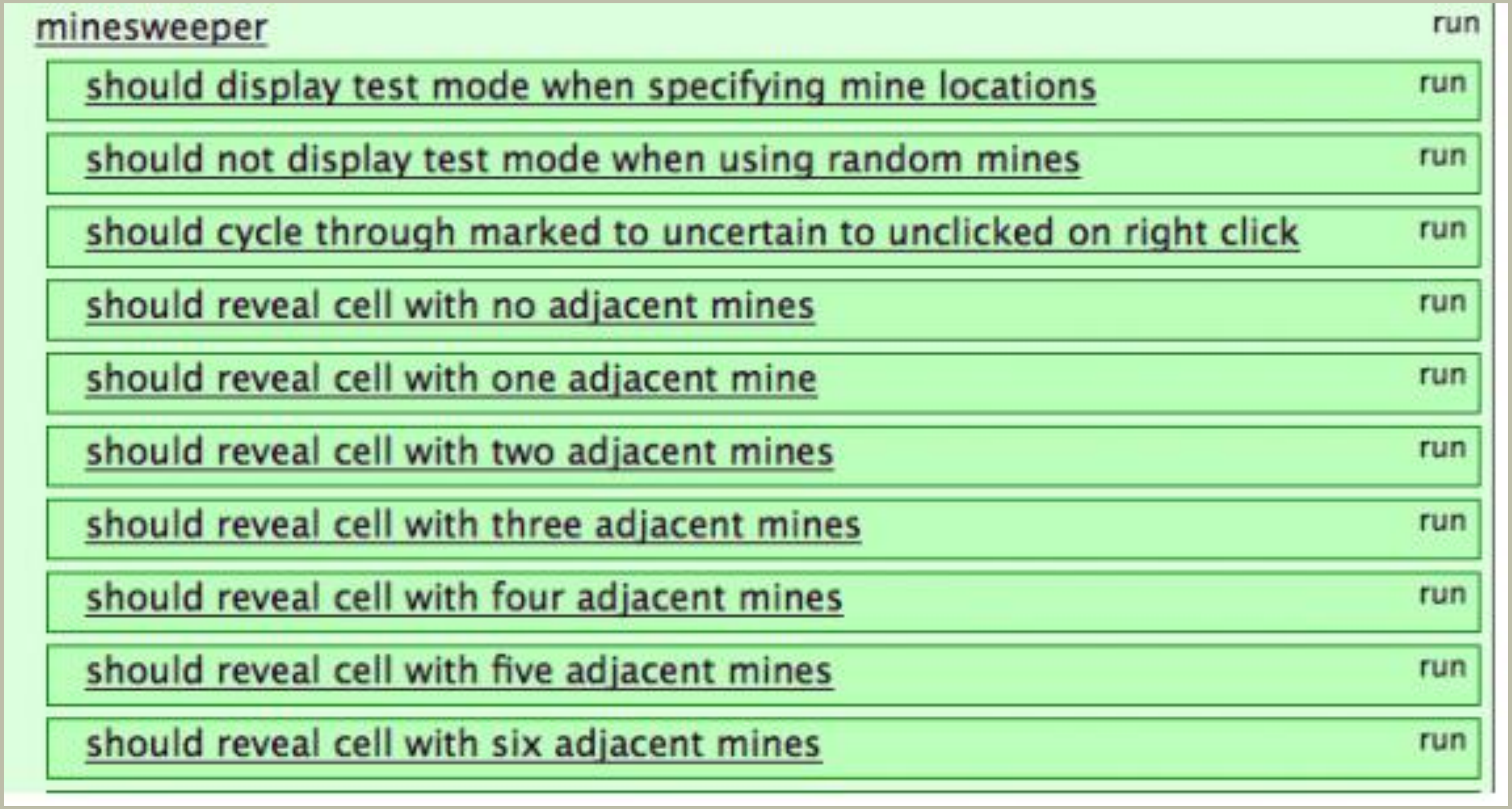


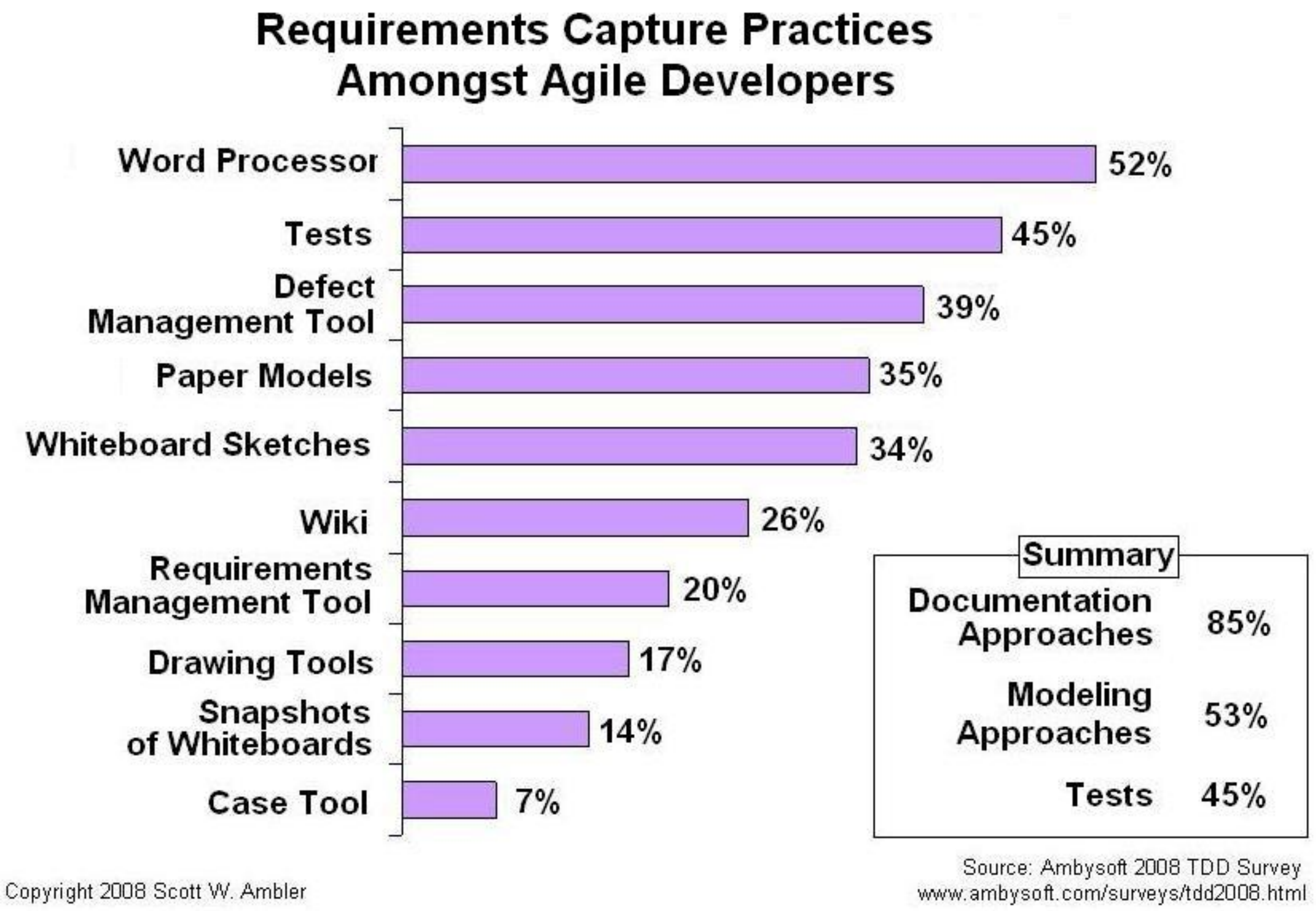
Figure 4 TDD has a profound effect on design and how you spend your time. In contrast to debug-later programming, the physics of TDD do not include the risk and uncertainty of tracking down bugs.

Creating Documentation With Teeth


minesweeper

- should display test mode when specifying mine locations
- should not display test mode when using random mines
- should cycle through marked to uncertain to unclicked on right click
- should reveal cell with no adjacent mines
- should reveal cell with one adjacent mine
- should reveal cell with two adjacent mines
- should reveal cell with three adjacent mines
- should reveal cell with four adjacent mines
- should reveal cell with five adjacent mines
- should reveal cell with six adjacent mines

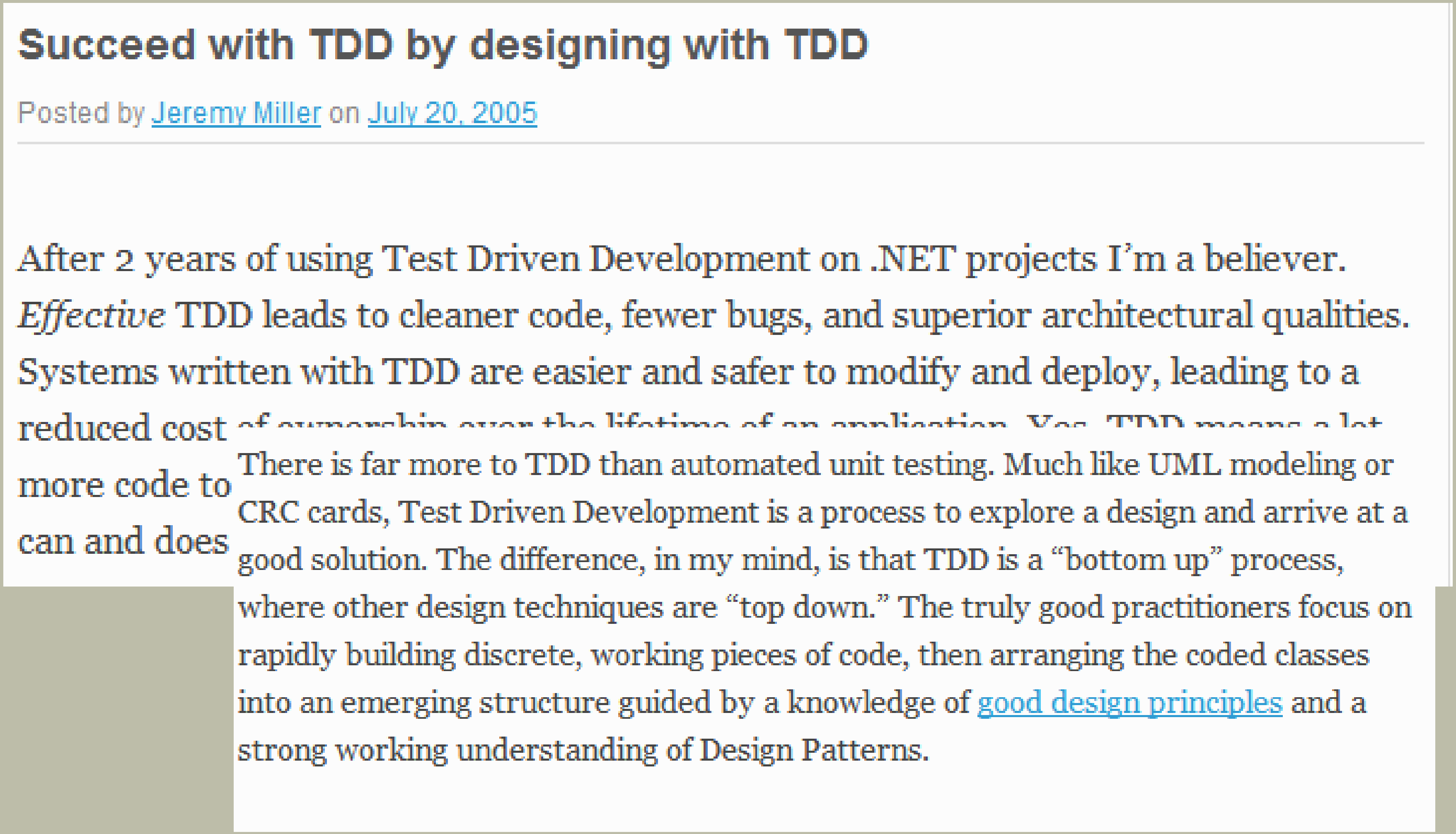
Capturing Requirements (Ambler 2008)



Requirements Capture Practices Amongst Agile Developers

Source: Ambysoft 2008 TDD Survey
www.ambysoft.com/surveys/tdd2008.html

TDD Is About Design



Succeed with TDD by designing with TDD

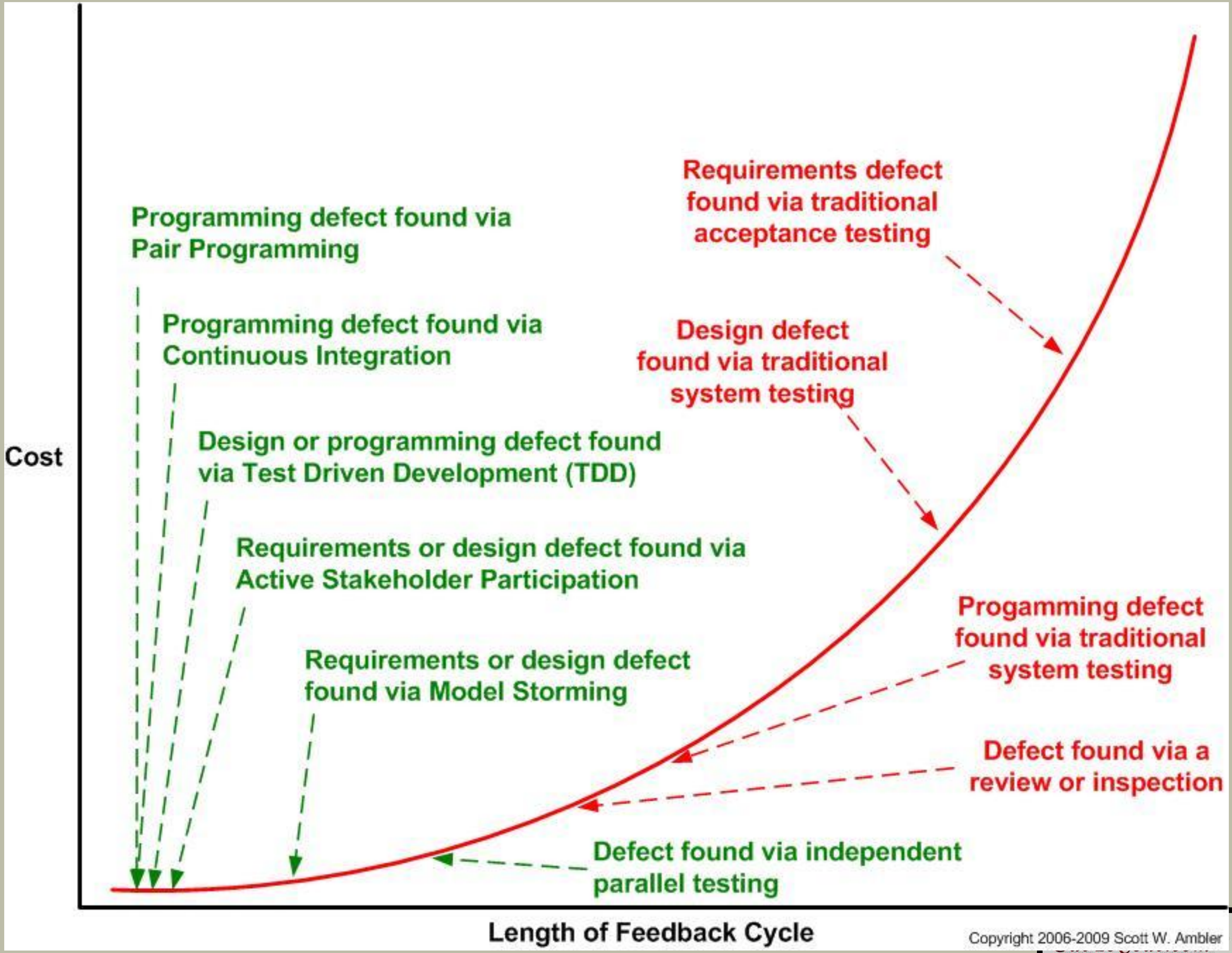
Posted by Jeremy Miller on July 20, 2005

After 2 years of using Test Driven Development on .NET projects I'm a believer. *Effective* TDD leads to cleaner code, fewer bugs, and superior architectural qualities. Systems written with TDD are easier and safer to modify and deploy, leading to a reduced cost of ownership over the lifetime of an application. Yes, TDD means a lot

more code to

can and does

There is far more to TDD than automated unit testing. Much like UML modeling or CRC cards, Test Driven Development is a process to explore a design and arrive at a good solution. The difference, in my mind, is that TDD is a "bottom up" process, where other design techniques are "top down." The truly good practitioners focus on rapidly building discrete, working pieces of code, then arranging the coded classes into an emerging structure guided by a knowledge of good design principles and a strong working understanding of Design Patterns.



Copyright 2006-2009 Scott W. Ambler

Testing Spectrum

- **Unit testing**
  - Testing an isolatable ‘unit’ of code, usually a class
- **Integration testing**
  - Testing a module of code (e.g. a package)
- **Application testing**
  - Testing the code as the user would see it (black box)

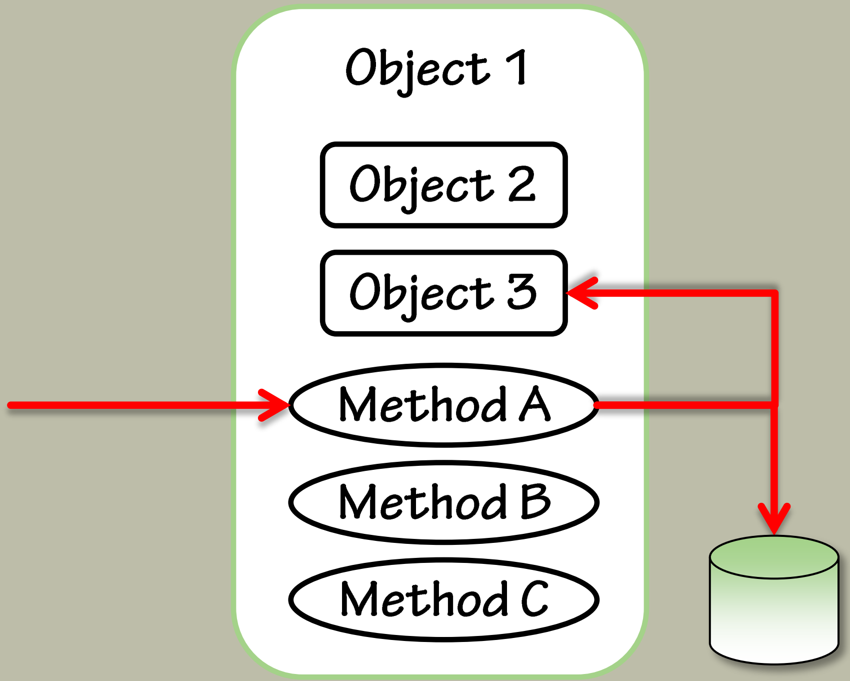
Unit Testing

- Programmer tests of individual units of code are fit for use
- A unit is a small testable part of an application
- Ideally, each test is independent
- Done in almost all programming languages
- Can be done with or without a unit testing framework

Testing a Unit

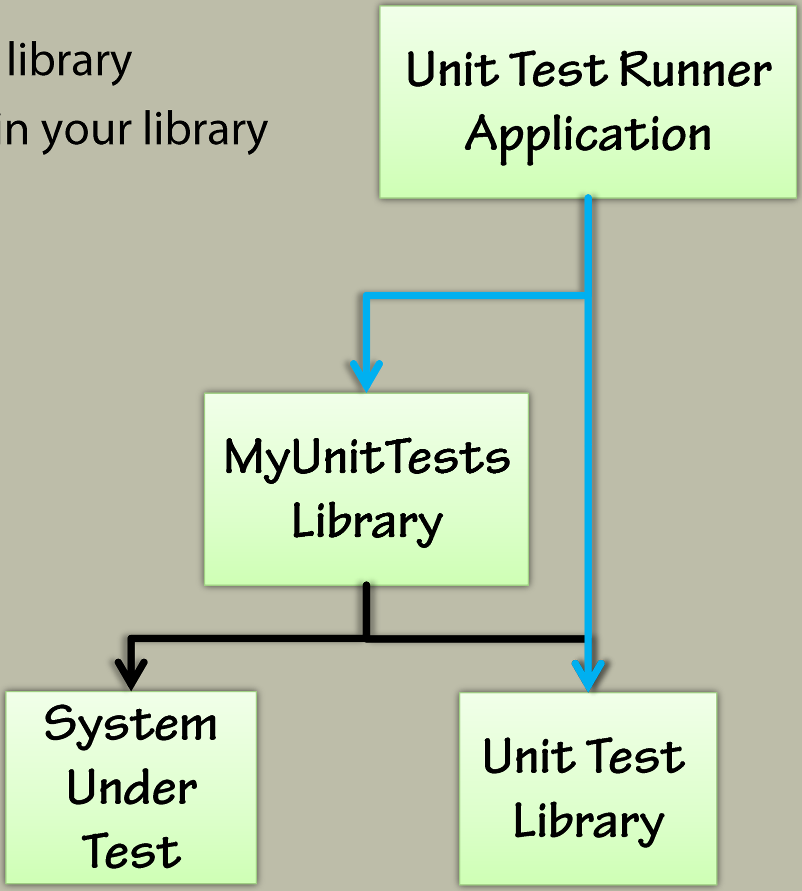
- **Unit Test Frameworks can be used to create**
  - Unit Tests
  - Integration Tests
  - An app to exercise your app

Unit vs. Integration Test

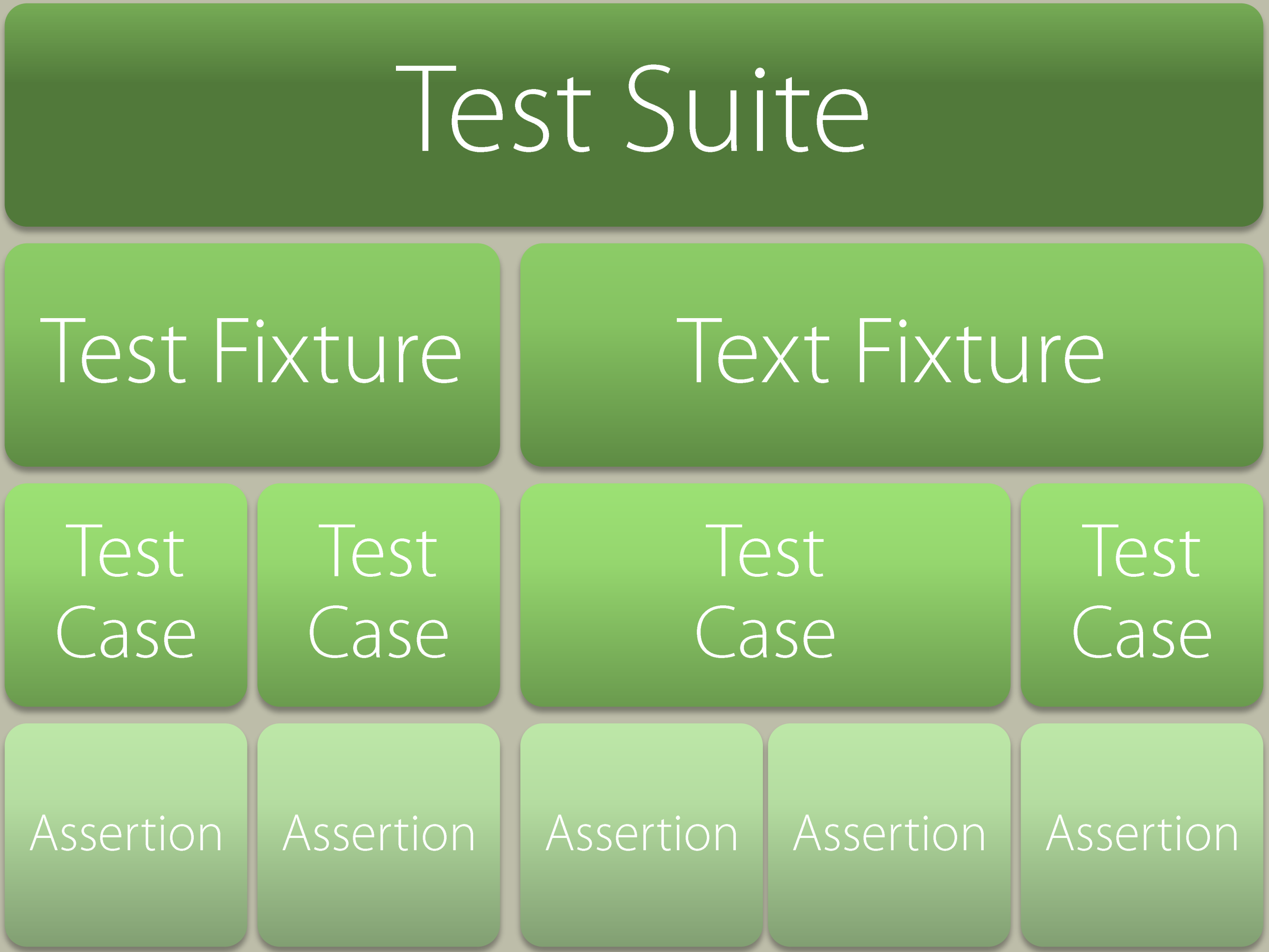


Unit Testing Frameworks

- Usually consists of:
  - A software library included by your test library
  - A runner application that invokes tests in your library
- Available for almost any language or runtime
- For .NET
  - MS Test
  - NUnit
  - MbUnit
  - Many others



xUnit Architecture



Test Suite

Test Fixture

Text Fixture

Test Case

Test Case

Test Case

Test Case

Assertion

Assertion

Assertion

Assertion

Assertion

MSTest Implements xUnit Architecture

xUnit Architecture	NUnit Implementation
Test Suite	Namespaces or Visual Studio Projects
Test Fixture	.NET Class
Test Case	Method
Assertion	Assert.AreEqual(0,1);

> TDD helps you pay attention to the right issues at the right time so you can make your designs cleaner, you can refine your designs as you learn.
>
> - Kent Beck

Test-Driven

Test-Driven Development

1. Create a new test

2. Run the test to see it fail

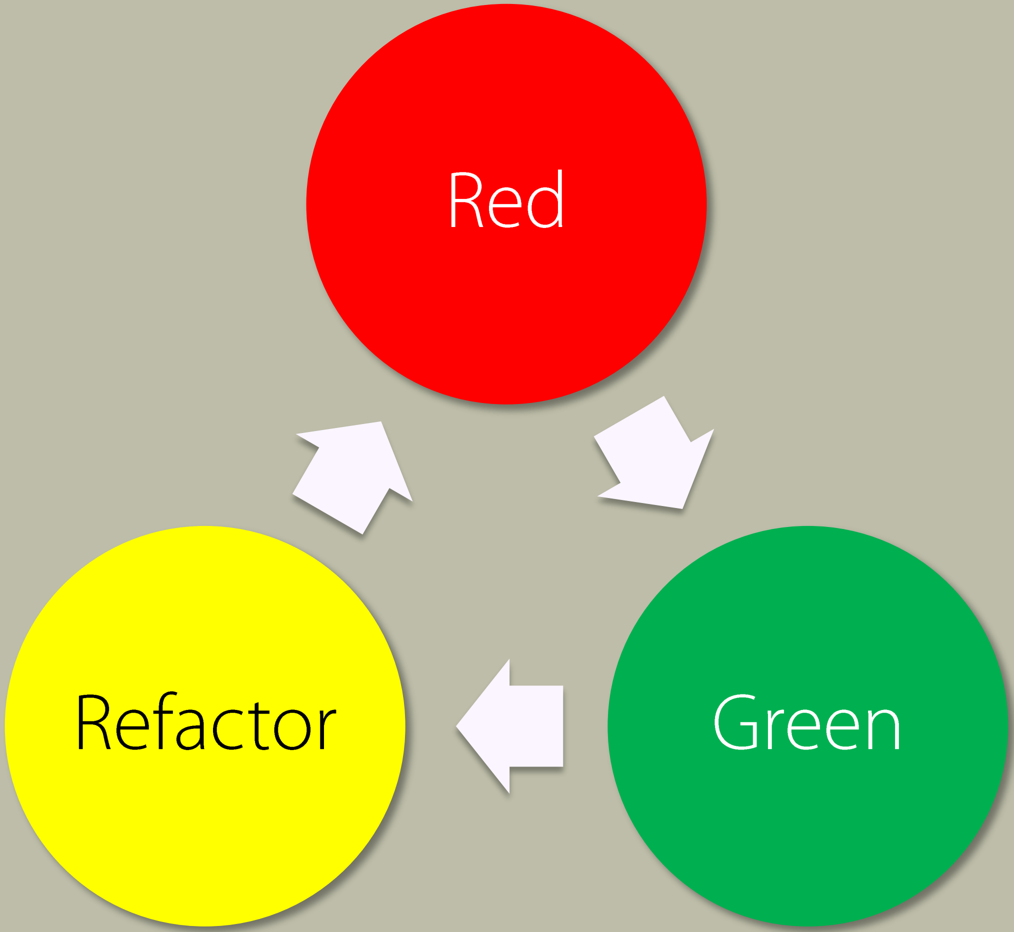
3. Make the test pass
   By adding functionality to the SUT

4. Refactor while keeping tests passing
   - Clean as we cook

5. Repeat

Creating tests before creating functionality in the SUT

The Mantra



The Result of TDD

- Keep our SUT working at all times
- Drive out design as it is needed
- Ensure I didn’t just break something
- Create a regression harness as a by-product
- Reduced defect density
- Typically creates more flexible, modular, and extensible code

Refactoring

From This

```
public int ConstantMult(int y)
{
       int x;
       x = 3;
       int p = x*y;
       return p;
}
```

Changing the internal implementation of code without changing its external behavior.

To This

```
public int MultiplyByFixedRate(int numberToMultiply)
{
       return numberToMultiply * 3;
}
```

Common Refactorings

- Rename
- Extract Super Class
- Pull Up Member
- Extract Interface
- Introduce Field

We Need Good Refactoring Tools

- **Or we won’t do it**
  - Because its too hard

- **And we must**
  - To improve our design
  - For code to be readable and maintainable
  - To express our intent

Summary

- Writing Unit Tests
- Unit Testing Frameworks
- Anatomy of a Test Fixture
- Test-Driven
- Basic Refactorings