Unit Tests as Specifications

Effective Ways to Think About TDD

David Scott Bernstein

- Software developer since 1980
- Trained 8,000 developers since 1990
- Author of the book Beyond Legacy Code: Nine Practices to Extend the Life (and Value) of Your Software

Website: http://ToBeAgile.com
Twitter: @ToBeAgile
My Book – Beyond Legacy Code

Nine practices to design and build healthy code, plus some tips on dealing with legacy code.

- Discusses the value and reasoning behind the technical practices, so both managers and the team can get on the same page as to their value.
- It’s not a “How To” book, it’s a “Why To” book.

http://BeyondLegacyCode.com

to be agile

Nine Essential Practices

1. Say What, Why, and for Whom before How: With a Product Owner defining the next most important features to build, the need for upfront requirements goes away.

2. Build in Small Batches: Building incrementally increases feedback, helps simplify the construction of complex systems, and reduces risks.

3. Integrate Continuously: Sets up the infrastructure for incremental development.

4. Collaborate: Spiking, pairing, and swarming as a team to solve problems and radiate knowledge throughout an organization.

5. Create CLEAN Code: Share standards and practices for building software with code qualities that support testability.

6. Write the Test First: Drops the cost of building and maintaining software dramatically.


8. Implement the Design Last: Paying technical debt can pay back dividends in the short term as well as the long term.


to be agile
Test First Development

to be agile

Test First or Test Last?
How Can it be Faster?

What We Stop Doing
Test First

Slow Then Faster
Benefits of TDD

Why TDD Works
TDD Can Fail

C-ohesive
L-ooosely Coupled
E-ncapsulated
A-ssertive
N-on-redundant

CLEAN Code is Testable Code
Another Client Told Me

Fast Tests
TDD Does Not Replace QA

What is a Test?
What a Test Tests

It Becomes a Test
The Dual Role

Drive Development with Tests
TDD Metaphors

An Introduction
Three Steps in Test First

Red Bar, Green Bars
Write a Failing Test

Celebrate the Red Bar
How Many is Enough?

On the Three Steps
For Example

Java/C#

- Suppose I want to write an adder class
- I’d start by writing a failing test
  ```java
  Adder adder = new Adder();
  assertEquals("1+1=2", 2, adder.add(1,1), .1);
  ```
- I’d then stub it out so it compiles
  ```java
  public class Adder {
      public int add(p1, p2) {
          return 0;
      }
  }
  ```

to be agile
A Mad Dash

Java/C#

- What’s the fastest way to get to the green bar?
  - Return 2
    ```java
    public class Adder {
        public int add(p1, p2) {
            return 2;
        }
    }
    ```

Green Bar
Let’s Try Another

Java/C#

```java
assertEquals("2+2=4", 4, adder.add(2,2), .1);
```

- And what happens?

Red Bar
Now I Have a Choice

- I can add a conditional logic to my code
  - if ((p1 == 1) && (p2 == 1)) return 2;
  - if ((p1 == 2) && (p2 == 2)) return 4;
- Or I can just…
  - return p1+p2;
- Notice how doing the right thing is also doing the easiest thing
Test as Design

Uncle Bob’s Laws of TDD

3
What Makes a Good Test?

Characteristics of a Good Test
What is a Unit?

Test Semantics
Instead of Doing This...

```java
@Test
public void testConstructor() {
    User user = new User("Clark", "Kent", "user@example.com", "Superman", "kryptonite");
    assertEquals("Clark", user.firstName());
    assertEquals("Kent", user.lastName());
    assertEquals("user@example.com", user.eMail());
    assertEquals("Superman", user.userName());
    assertEquals("kryptonite", user.password());
}
```

to be agile
Do This

```java
@Test
public void testRetrievingParametersAfterConstruction() {
    private static final String firstName = "Clark";
    private static final String lastName = "Kent";
    private static final String eMail = "user@example.com";
    private static final String userName = "Superman";
    private static final String password = "kryptonite";

    User user = new User(firstName, lastName, eMail, userName, password);
    assertEquals(firstName, user.firstName());
    assertEquals(lastName, user.lastName());
    assertEquals(eMail, user.eMail());
    assertEquals(userName, user.userName());
    assertEquals(password, user.password());
}
```

to be agile

Generalizations

- Instrumentation can also specify generalizations:

```java
public void testAddition() {
    private int anyInt = 1;
    private int theResult = 2;

    Adder adder = new Adder();
    assertEquals(theResult, adder.add(anyInt, anyInt));
}
```

to be agile
Defining a Linear Range

- How many assertions do you need to specify a linear range?
  - 4... or 3
- For example, to validate value is within a range from MINIMUM_VALUE to MAXIMUM_VALUE:

  ```java
  assertEquals(value, MINIMUM_VALUE - 1); // exception
  assertEquals(value, MINIMUM_VALUE);     // valid
  assertEquals(value, MAXIMUM_VALUE);     // superfluous?
  assertEquals(value, MAXIMUM_VALUE + 1); // exception
  ```
Specifying Constants

- Using tests as specifications requires completeness
- “That which is not specified is specified to be false.”
- Any code change that could mutate behavior should have a test
- This includes having asserts for constants:

  ```java
  assertEquals(MINIMUM_VALUE, 1);
  assertEquals(MAXIMUM_VALUE, 10);
  ```

Two Kinds of Tests
Boundary Testing

Workflow Testing
Testable Code

Don’t “Test Until Bored”
TDD is Not Enough

Thank You!

Please fill out your feedback forms!

- We have just scratched the surface, to learn more:
  - Read my blog: http://ToBeAgile.com/blog
  - Sign up for my newsletter: http://ToBeAgile.com/signup
  - Follow me on Twitter (@ToBeAgile)
  - Read my book:
  - Attend my one of my Certified Scrum Developer trainings
    - See http://ToBeAgile.com/training for my public class schedule
    - Or contact me to arrange a private class for your organization
  - Visit http://ToBeAgile.com for more information
<table>
<thead>
<tr>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>to be agile</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>to be agile</td>
</tr>
</tbody>
</table>