TDR & TDD
Test-Driven {Requirements & Development}
By Mike Rieser (email:mike@mrieser.com)

“People have told me that they see Scrum as the management approach to agile development and XP as the engineering practices that make it effective, both bonded together by complimentary practices and goals.”

— Ken Schwaber

ENGINEERING PRACTICES
Scrum for Process, Extreme Programming (XP) for Practices

“Accordingly, whenever I have a situation in which software is being developed, Scrum is being used, and the engineering practices are substandard, I also call for XP to be implemented.”

– Ken Schwaber

Scrum’s Influence on Extreme Programming (XP)

From: Kent Beck To: Jeff Sutherland <jsutherland>
Reply: 70761.1216@compuserve.com
Date: Mon, 15 May 1995 18:01:15 -0400 (EDT)
Subj: HBR paper

Is there a good place to get reprints of the SCRUM paper from HBR? I’ve written patterns for something very similar and I want to make sure I steal as many ideas as possible.

Kent
MATURING PRACTICES

Discussion

Teams new to Scrum typically start by putting everything in progress and treating the iteration like a mini-waterfall.

How have you seen teams mature in the way they approach and accomplish their work?
Skill Maturity Models

Agile teams mature in their skills and practices:
- **Dreyfus Model**: Novice to Expert.
- **Martial Arts**: Shu-Ha-Ri.
- **Meilir Page-Jones**: Innocent to Researcher.

**Novice Agile**

Treat an iteration like a *mini-waterfall*. All stories put in flight at once. QA gets squeezed at the end having to test everything the night before the demo.
QA Plight
The Tester is at the end of the process and may have questions.

Advanced Beginner
Team starts completing stories earlier, but still approaches each story in a traditional fashion.
Steven Covey tip: “Begin with the end in mind.”
Let’s isolate a User Story and think about it.

“Begin with the end in mind.”
What if we began with QA Testing?
QA Tests and Requirements at the start.  
This wraps a lot of things around. Notice where the BA and QA end up?

Separate out the Dev tasks.
Isolate each area.

Rearrange the Dev tasks. Code moves earlier.
We’ll redirect the flow.

Analysis feeds Tests. From Tests we go straight to Code.
Let’s name each of these key areas.

Test-Driven Requirements + Test-Driven Development
Mature team with mature practices
Practicing Test-Driven Requirements with Test-Driven Development

“An example would be handy right about now.”
– Brian Marick

TEST-DRIVEN REQUIREMENTS
Test Tables are best done collaboratively!

Software Requirements as a Table

“...any software program can be thought of as a table that maps program inputs to their corresponding correct outputs.

This characterization of programming assumes a finite number of finite inputs, which is fair for practical purposes”

July 2001 issue of ACM Software Engineering Notes, J. P. Lewis
## World of Warcraft Character Movement

<table>
<thead>
<tr>
<th>Character Name</th>
<th>Character Location</th>
<th>Object Location</th>
<th>Move</th>
<th>Location?</th>
<th># Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operandi</td>
<td>(0, 0, 0)</td>
<td>-</td>
<td>Forward</td>
<td>(1, 0, 0)</td>
<td>Unhindered movement</td>
</tr>
<tr>
<td>Operandi</td>
<td>(0, 0, 0)</td>
<td>(-1, 0, 0)</td>
<td>Backward</td>
<td>(0, 0, 0)</td>
<td>Blocked movement</td>
</tr>
</tbody>
</table>

## ATM Withdraw Cash Test Table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1,000</td>
<td>$5,000</td>
<td>$100</td>
<td>OK</td>
<td>$900</td>
<td>$4,900</td>
<td>Happy</td>
</tr>
<tr>
<td></td>
<td>$1,000</td>
<td>$1,000</td>
<td>$1,000</td>
<td>OK</td>
<td>0</td>
<td>0</td>
<td>Boundary</td>
</tr>
<tr>
<td><strong>$100</strong></td>
<td>$5,000</td>
<td><strong>$200</strong></td>
<td>ERROR</td>
<td></td>
<td>$100</td>
<td>$5,000</td>
<td>Insufficient ATM cash</td>
</tr>
<tr>
<td>$1,000</td>
<td>$100</td>
<td>$200</td>
<td>ERROR</td>
<td>$1,000</td>
<td>$100</td>
<td>Insufficient acct balance</td>
<td></td>
</tr>
<tr>
<td>$1,000</td>
<td>-$100</td>
<td>$200</td>
<td>ERROR</td>
<td>$1,000</td>
<td>-$135</td>
<td>Overdraft fee!</td>
<td></td>
</tr>
</tbody>
</table>
Gherkin – Given-When-Then

Given an Account with a $5,000 balance
   And the ATM with $1,000 cash

When the Patron withdraws $100 cash

Then the Account balance is $4,900
   And the ATM has $900 cash

Common Tools

• Fit (or Slim) and FitNesse (Java or C#)
• SpecFlow (C#)
• Jbehave (Java)
• Cucumber (Ruby)
• Concordian
• Nominations?
Discussion

- Anyone have stories of Test-Driven Requirements sparking collaboration on teams?
- When’s the best time to have the team create the tests?
Software Development Process Dependencies

Waterfall
- Upfront Design
  - Upfront Requirements

Agile
- Evolutionary Design
  - Refactoring
    - Automated Developer Tests

Test First
Test After

Test-After Approach

Design 😊 → Code 😞 → Unit Test 😞

Traditional Way – Test After
- 1 day to think and design
- 2 days to write the code
- Oh yeah, 2 days to write the unit tests
Test-First Approach

TDD Way – Test First
• Takes about the same time as just design and code did before
• Unit Tests seemingly for free – how?

What is TDD?
• Developer Practice
• Test-First Programming
• Incremental Design
• **One** test at a time
• TDD does unit testing but unit testing is not necessarily TDD
What TDD is Not!

- Writing all the tests before the code
- Not QA Testing
- Not Integration Testing

What is Red?

A failing test is like a hunting license to change production code.

- **Enhancements** – add a test to add new functionality.
- **Defects** – reproduce the bug as a failing test.

Only introduce as much of a test as you can get to green in say, 5 mins.
What is **Green**?

Get to **Green** quickly!

**Obvious Implementation** – if you know what to do, just do it!

**Fake It!** – when you don’t use a constant!

What is **Refactor**?

- Work incrementally.
- Be able to show **Green** after each change.
- Remove duplication.
- **Clean as you Go!**
Refactor – what do you mean?

```java
if ( ... ) {
    ...
    i = i + 1;
} else {
    ...
    i = i + 1;
}
```

```java
if ( ... ) {
    ...
} else {
    ...
}
i = i + 1;
```

What is Integrate??

- Integrate as frequently as possible.
- May not be complete, but if done-enough, share it.
- (I typically avoid checking in Fakes.)
Test-First vs Test-After

**Test-First**
- The tests get written.
- Testable code is generally flexible code.
- Lower coupling
- Higher cohesion
- Allows Refactoring
- Confidence

**Test-After**
- Sometimes the unit tests get “skipped.”
- Tests that start from a wizard are generally pretty poor.

What Are Unit Tests?

- Tests which have a specific focus which call a method (or function).
- Verify that the requirements are being met by the code when tested in isolation.
- “Like a compiler” for the business requirements.
What Unit Tests are NOT!

WARNING
Do Not Hit the Database

DANGER
Do Not Hit the File System

Do Not Have a Complicated Environment

Benefits of TDD?

• Test first is more fun that test after
• Unit Tests actually get written
• Produces testable designs
• Think outside-in, first about how to use, then about how to do it
• Good unit tests that support Refactoring
• Requirements Clarification & Understanding
• Confidence (no fear)
• Solutions that fit like a hand in a glove
• Version 1.0 code that you are happy with
• Reduces Defects
TDD – “Rediscovered” by Kent Beck

“To the unknown author of the book which I read as a weird 12-year old that suggested you type in the expected output tape from a real input tape, then code until the actual results matched the expected result, thank you, thank you, thank you.”

– Kent Beck

(Test-Driven Development By Example)

Questions?

THE END