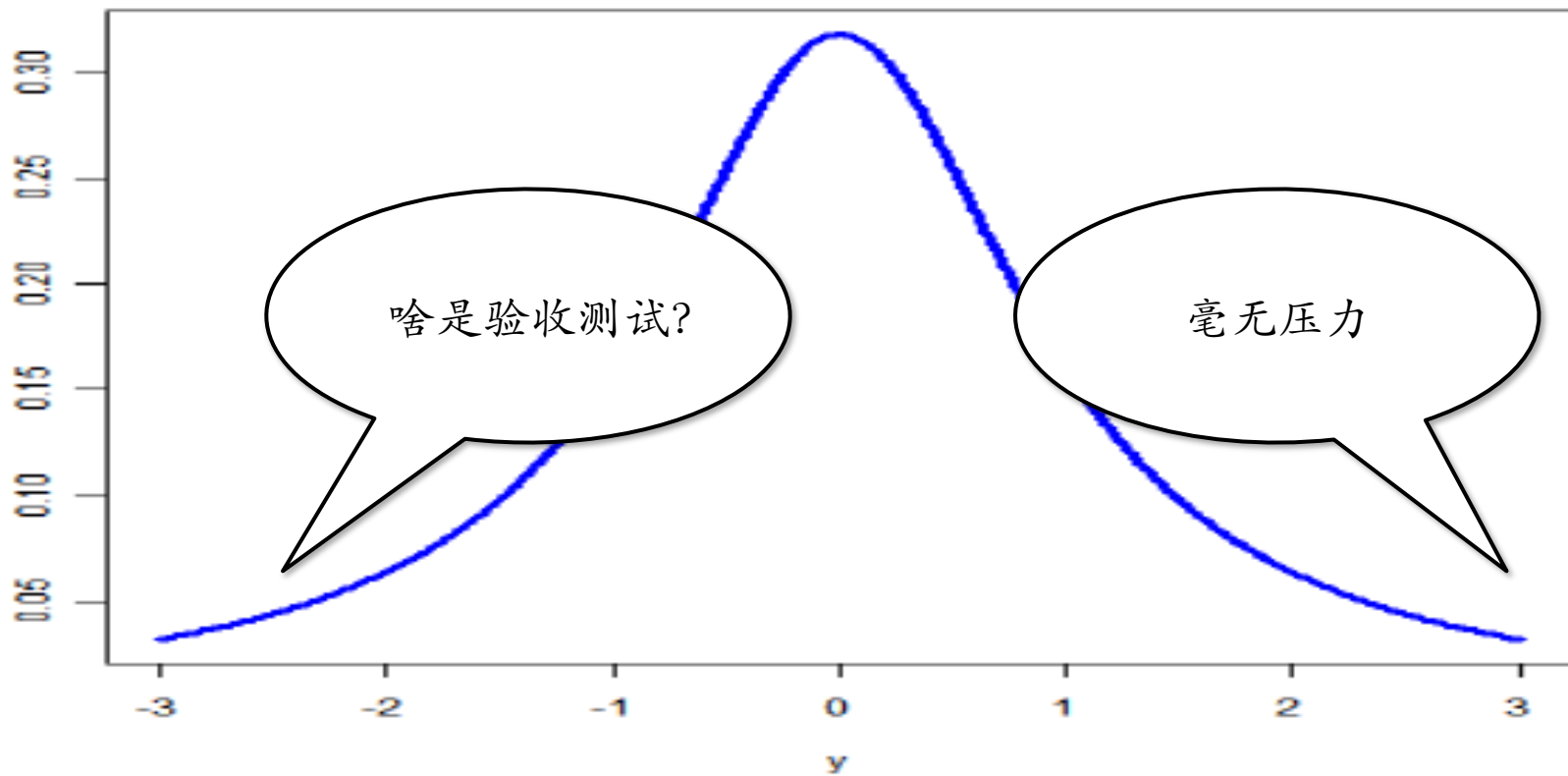


可维护自动化 验收测试

麦宇安 @ **PERFICIENT**

您在哪?



热身场景

以下场景，哪些具体可观测的结果会告诉你这个活动已经完成？

- 外出看电影
- 外出用膳
- 去购物

验收标准

是一组条件，只有当这些条件都满足了，一个故事才能被验收为完成

通常由客户或者PO提供

并非用来替代沟通

是沟通的产物之一

验收标准并非测试

编写验收标准

应该包括:

- Actor
- 动作: 描述一种行为
- 可观测的结果

加上前提条件验收标准可以描述为:

- 假设 [前提条件]
- 当 [Actor+行为]
- 那么 [可观测的结果]

验收测试

验收标准
+ **例子(数据+场景)**

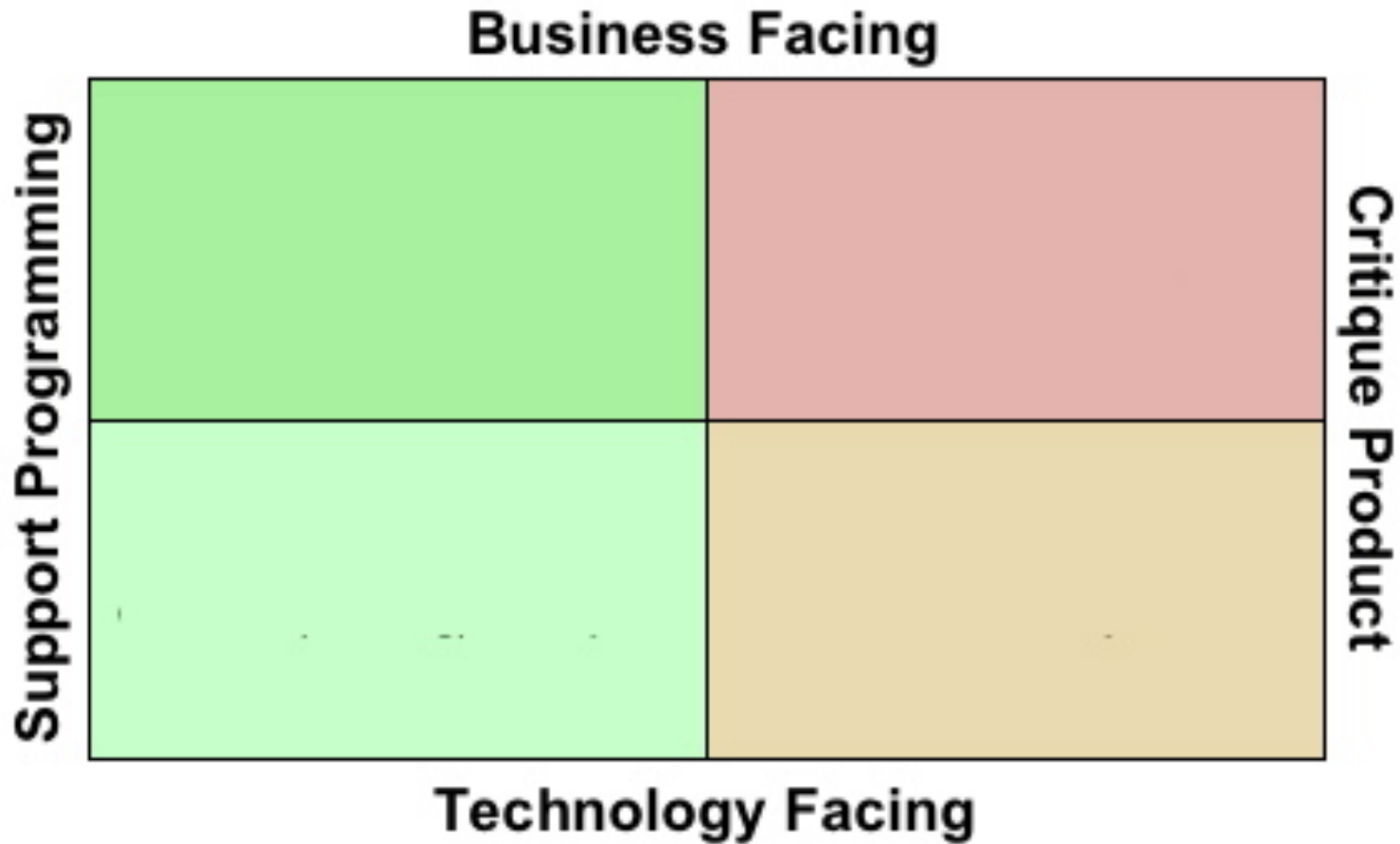
验收测试

**Are we
building the
right thing?**

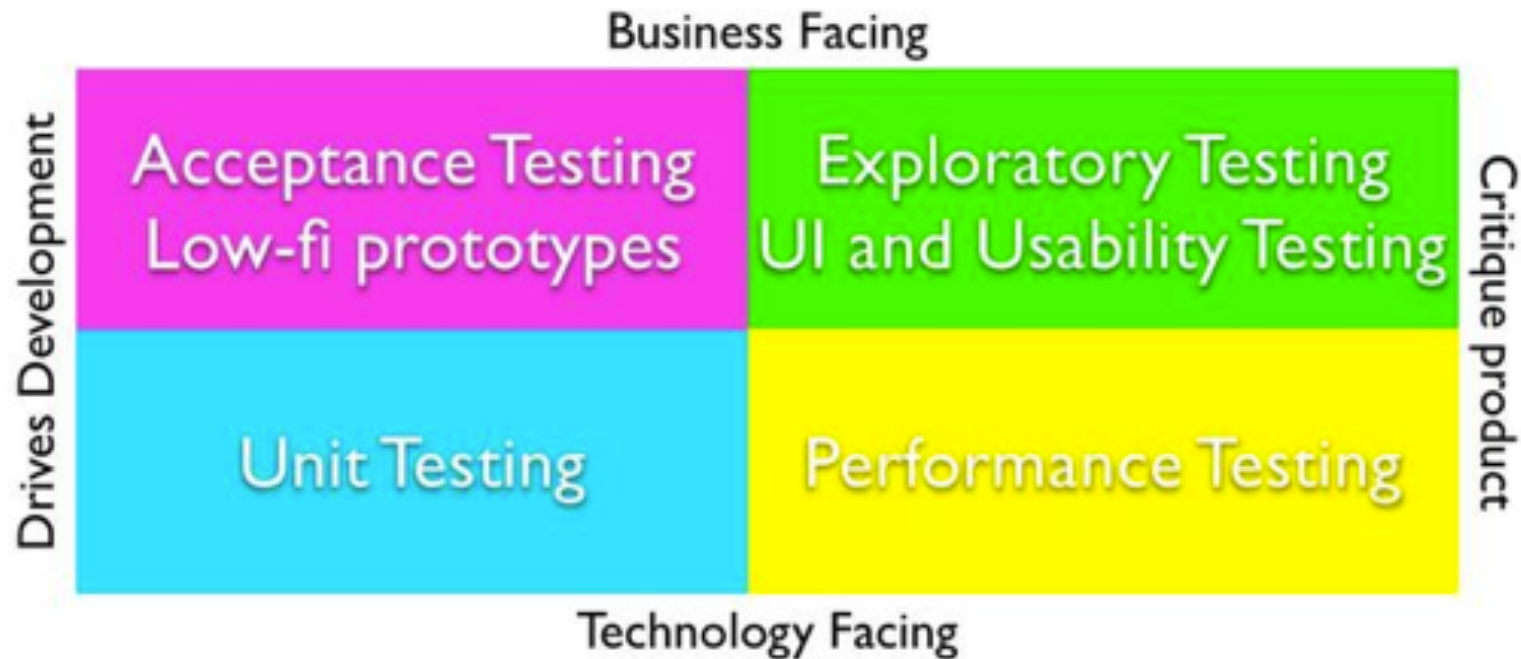
**Are we
building the
thing right?**



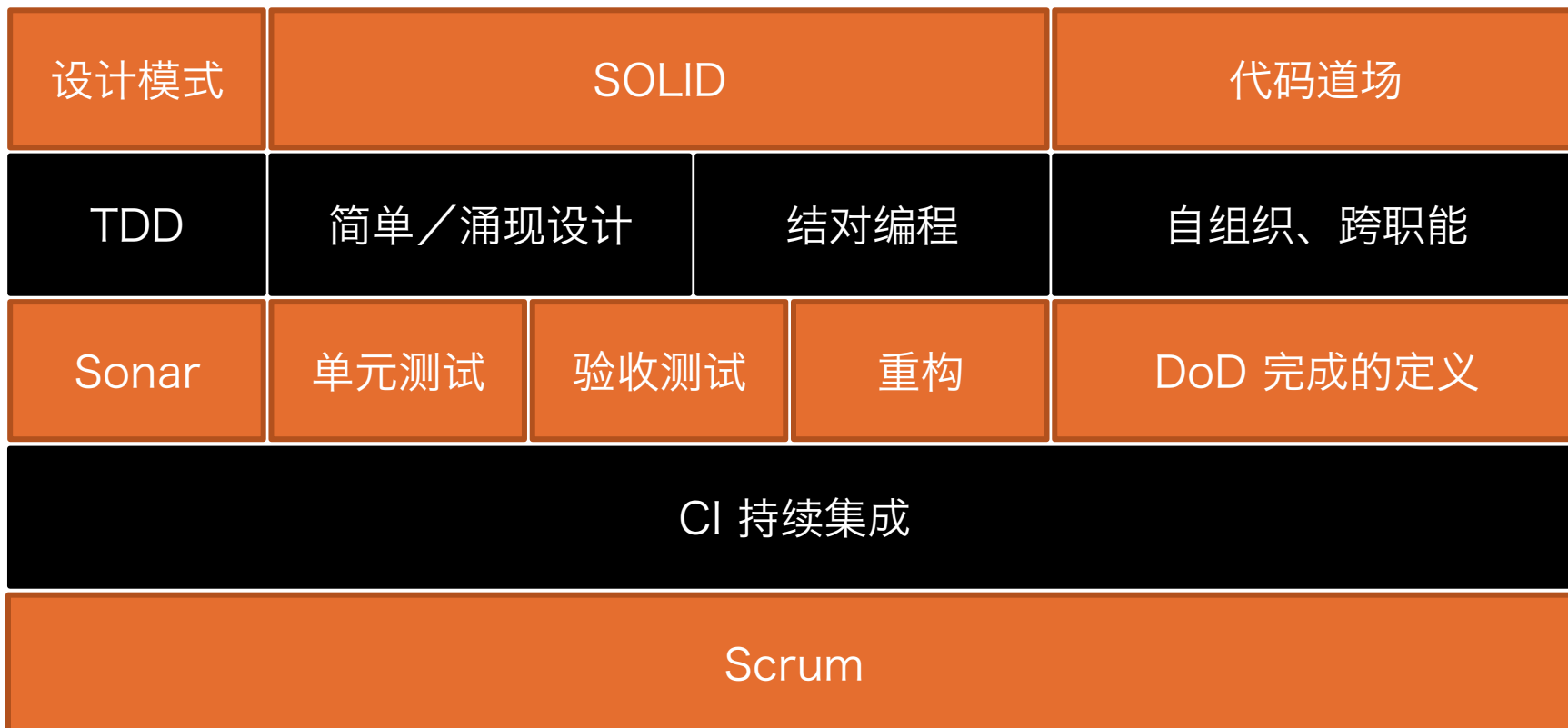
BRIAN MARICK



测试分类

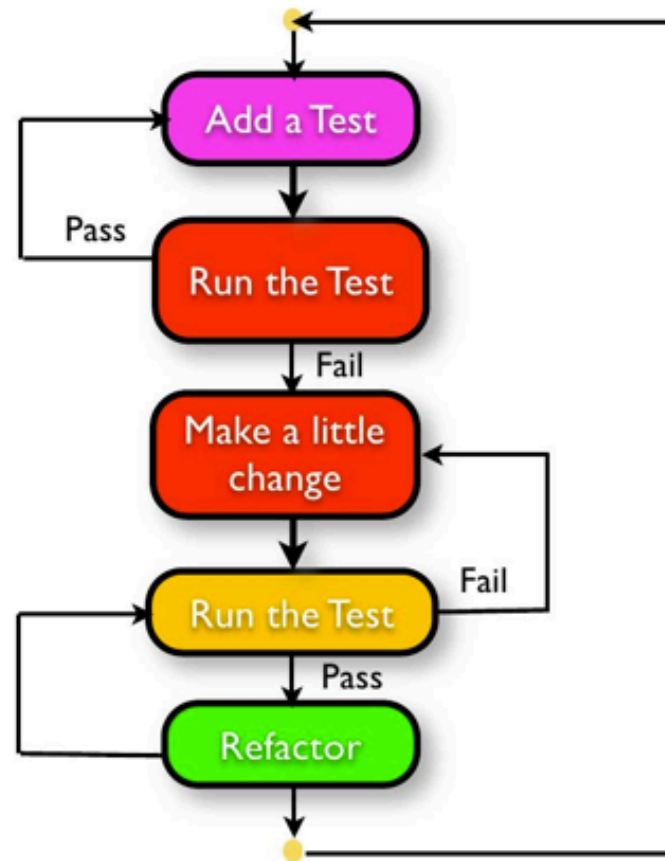


敏捷交付引擎

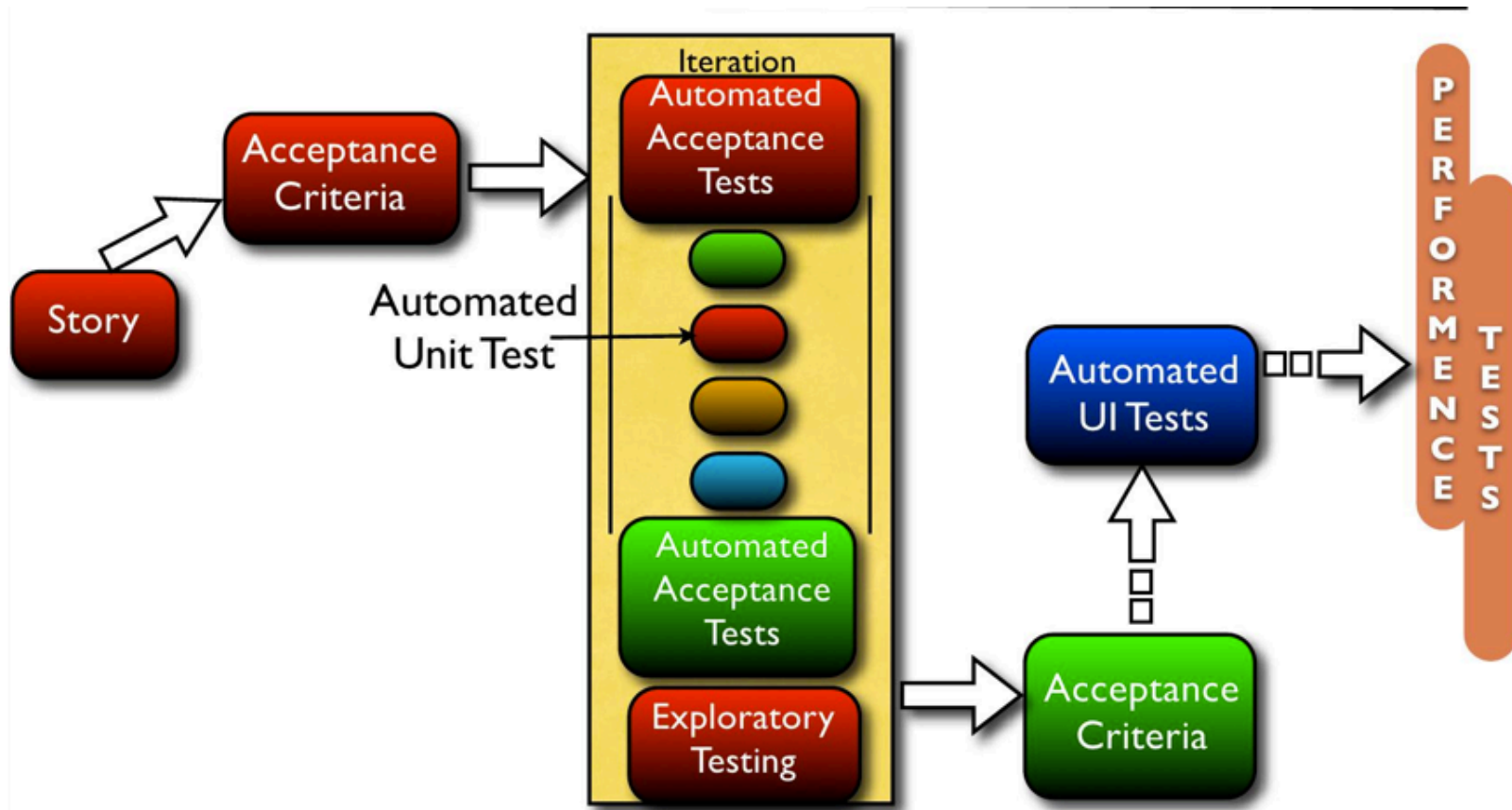


测试驱动

TDD Rhythm - Test, Code, Refactor



验收测试驱动



MIKE COHN测试金字塔

界面
测试

若干
Selenium

验收测
试

至少每个故事一个
RSpec、SpecFlow

单元测试

至少每个类一个
xUnit

验收测试驱动开发

验收测试是系统期望行为和功能的可执行规格说明

- 用问题领域的语言来表达
- 自动化
- 回归测试

ATDD与TDD

TDD非常由价值，但你需要更多

ATDD侧重于完整的特性和功能

ATDD：宏观

TDD：微观

完成的定义

每个故事必须有至少一个验收测试

一个故事不能算完成，除非它的验收测试通过了

验收测试担当的角色

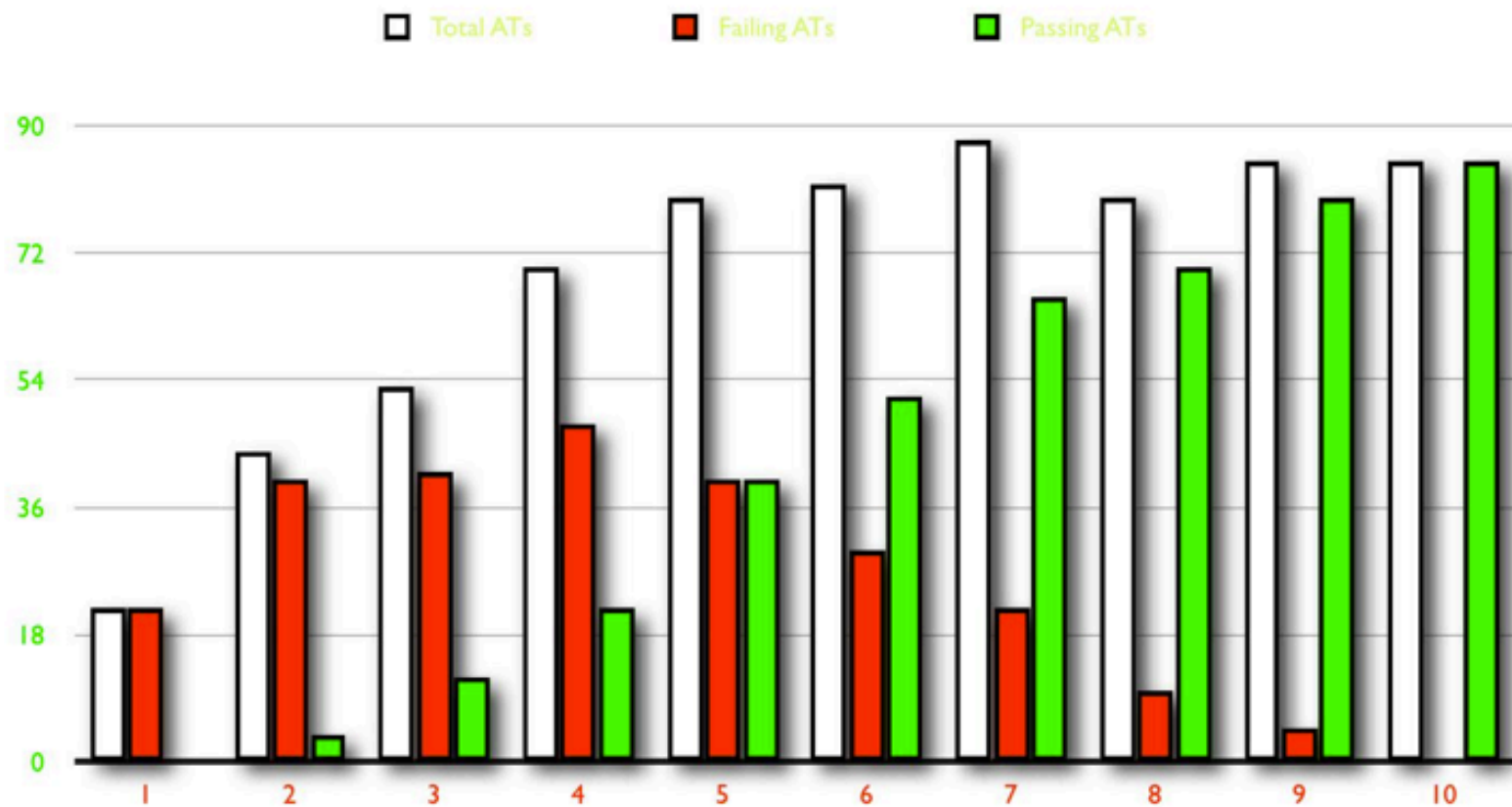
完成的标准

作为协作工具

其中一种反馈

测量进度

验收测试的数据



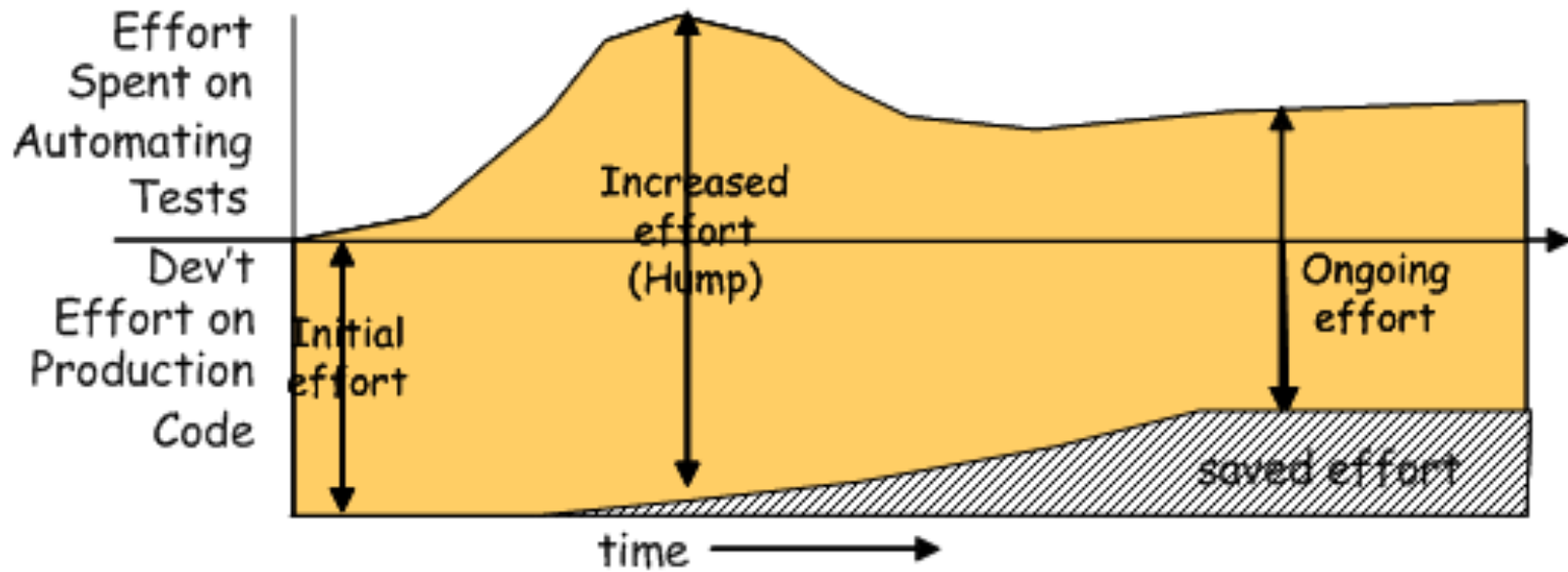
可执行规格说明

+

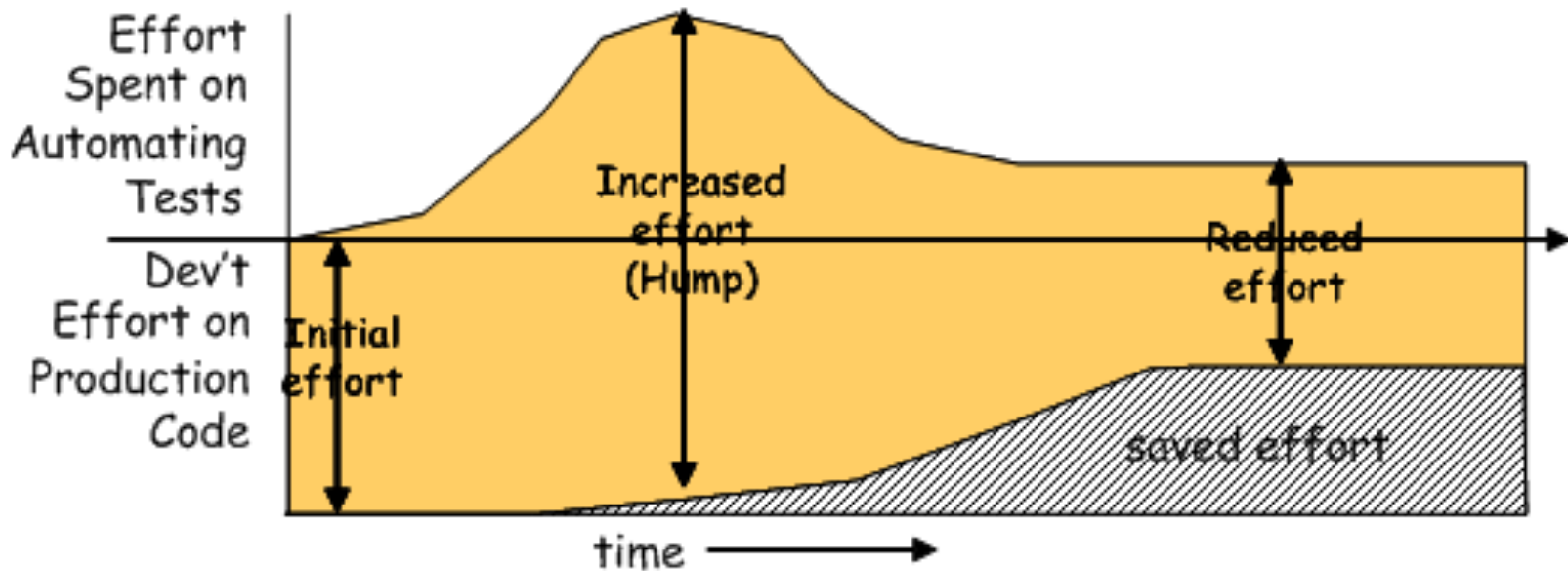
完成的标准
自动化

可执行的规格说明

低回报率的项目



高回报率的项目



<http://xunitpatterns.com/Goals%20of%20Test%20Automation.html>

自动化测试的目标

测试应帮助我们提高质量

- 测试作为规格说明
- 预防缺陷
- 帮助迅速定位缺陷，减少Debug

测试应帮助我们理解被测系统(SUT)

- 测试作为文档

测试应减少（而非引入）风险

- 测试作为安全网
- 测试案例不能对产品造成影响，不能修改SUT

测试应容易运行

- 完全自动化测试
- 能自我检查，无需人工检查每个测试用例的结果

- 具有重复性，具有自我清理能力

测试应容易编写和维护

- 简单测试
- 清晰表达目的
- 用例测试单一独立的功能

系统演化的过程中测试应需要尽可能少的维护成本

- 健壮测试

谁来写验收测试?

PO

开发人员

一个典型的用户故事

作为管理员，我希望用户创建的帐号必须使用安全的密码，以免他人使用猜密码程序破解其帐号。

讨论

至少7个字符，
必须包含字母、
符号、数字。

而如果用户提供的
密码不安全则
显示错误信息。

怎样才算“安
全”的密码？



捕捉为具体的例子

密码	有效
p@ssw0rds	是
p@s5	否
passw0rd	否
p@ssword	否
@#\$%1234	否

捕捉为具体的期望

假设 用户正在创建帐号

当 他使用了不安全的密码

那么 他会看到这个消息：“密码至少包含7个字符，必须包含字母、数字和符号”。

ATDD好处#1

需求的含糊之处能及早得到澄清

BUG紧急会议，离交付还有两天

这是个bug

不是bug

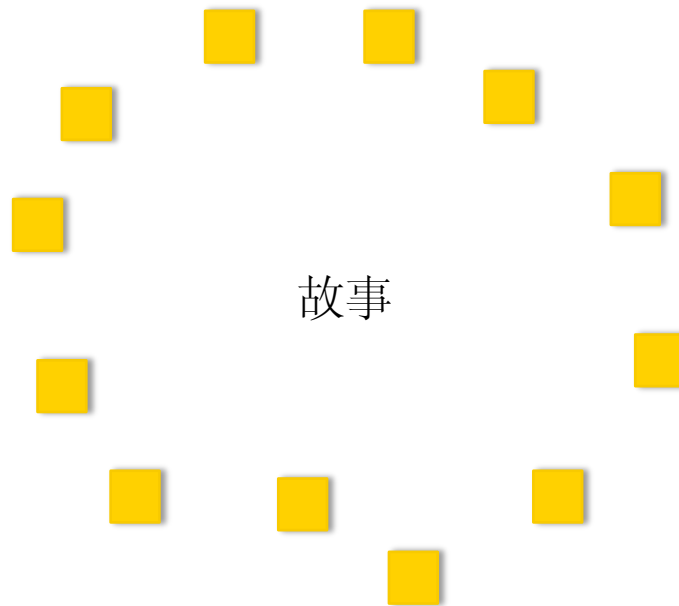
是的

不是的

不管是不是bug，
如果我们作出修改，
就没法如期交付了



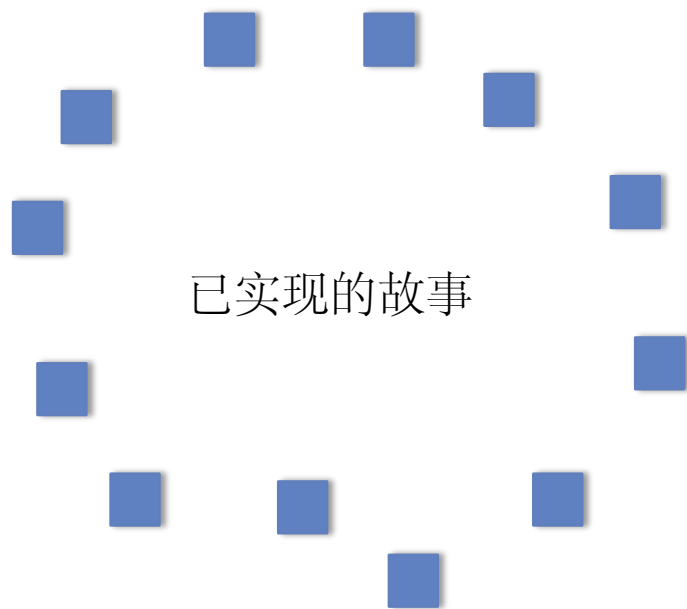
验收测试划定了故事范围



ATDD好处#2

使进度可视化

我们到了吗？



测试和代码的版本控制

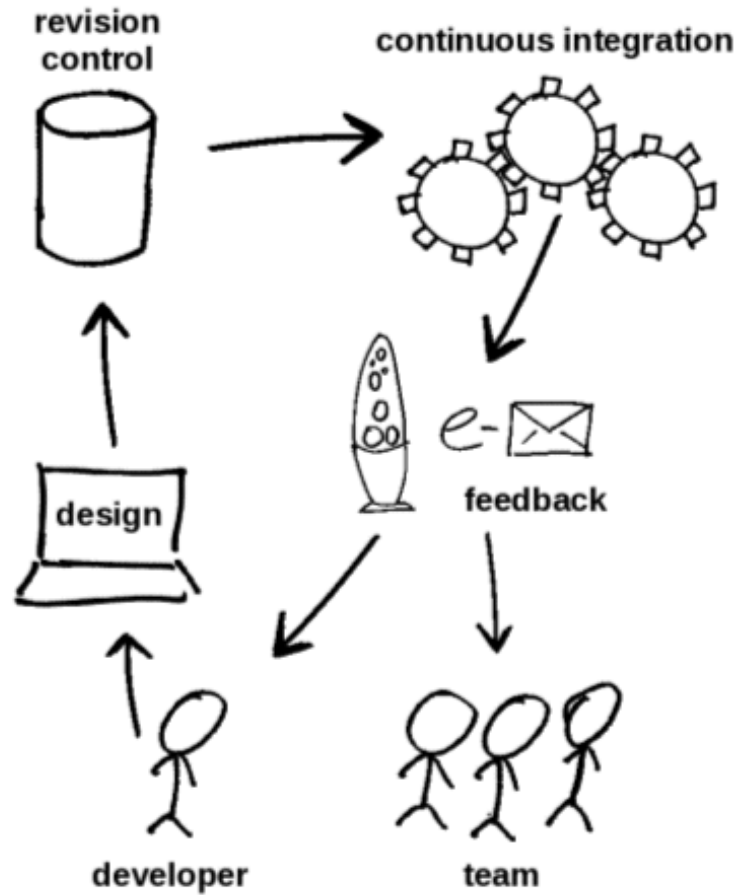
规格说明

测试固件

单元测试

产品代码

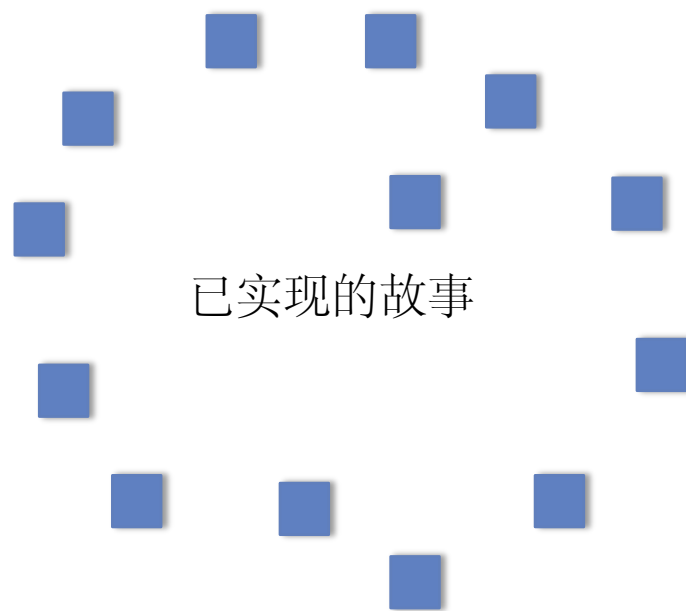
测试是持续集成的一步



ATDD好处#3

减少缺陷

对缺陷零容忍



编写可维护的 验收测试

登录测试

在浏览器中打开登录页面

输入用户名” 张三”

先建立一个可测试的环境

登录测试

往系统中加入一些用户

往用户名框填入一个值

测试需要是例子

使用具体的例子

指定具体的行为

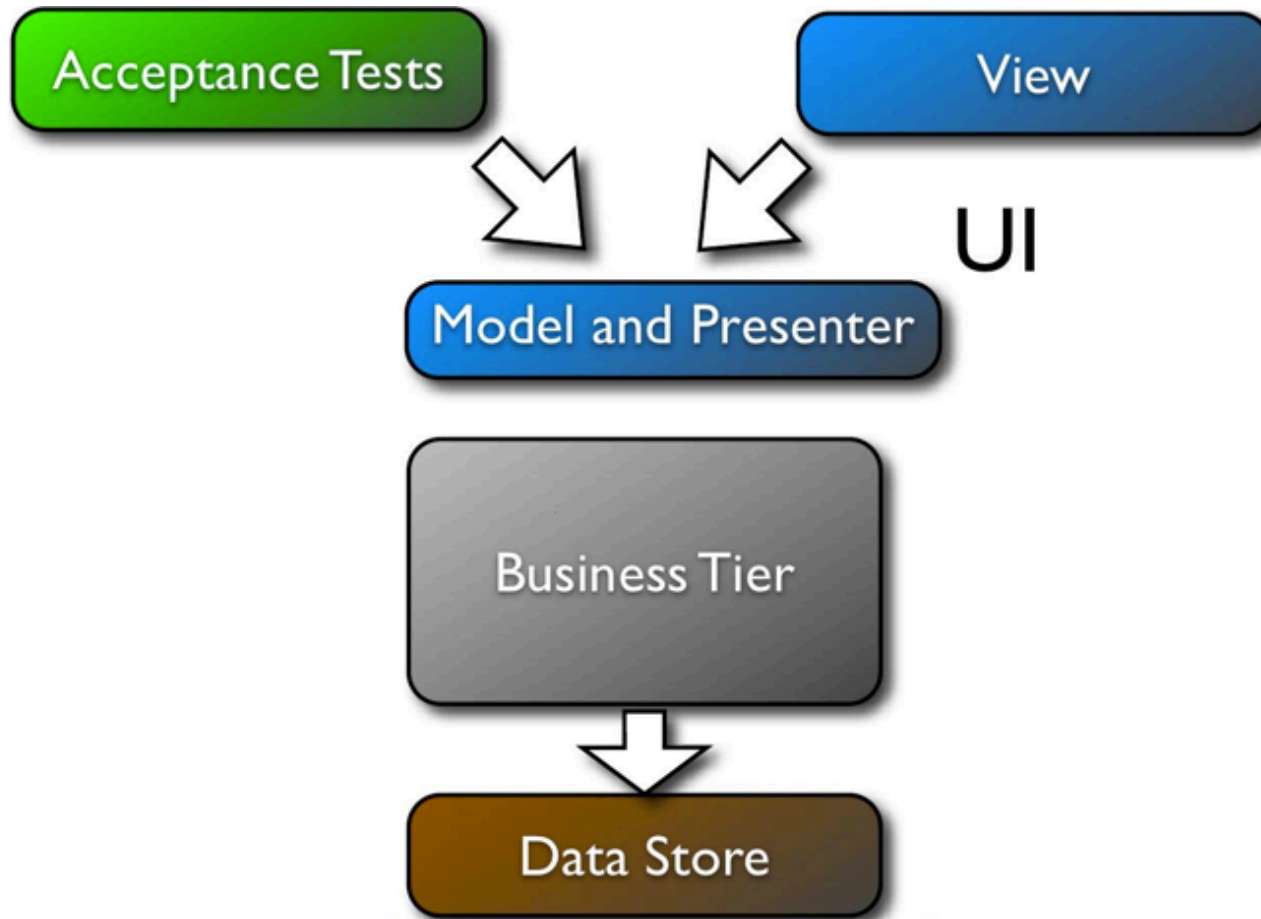
不允许含糊不清

登录测试

往User表插入纪录(“张三” “密码”)

打开链接<http://localhost/mysite>

如何避免实现细节



可能的方案

假设 用户(“张三”, “123”)存在于系统

当 用户使用用户名“张三”和密码“abc”登录

那么 他会登录失败

当 用户使用用户名“张三”和密码“123”登录

那么 他会登录成功

反模式：测试人员**AT**

开发人员自己替自己写验收测试

反模式：单元测试**AT**

不要在单元测试进行AT

- 单元测试是和实现相关的
- 验收测试是实现无关的

反模式：测试后行

在完成产品代码后才进行验收测试

反模式：实现相关的**AT**

让测试数据依赖于实现细节或者数据结构

规格的坏味道

规格经常需要变更

规格由一连串步骤组成

规格需要大量的固件代码

规格中的例子有类似的结构

SPECFLOW保龄球积分

ScoreCalculation.feature

Feature: Score Calculation

In order to know my performance

As a player

I want the system to calculate my total score

Scenario: Gutter Game

Given a new bowling game

When all of my balls are landing in the gutter

Then my total score should be 0

Scenario: All Strikes

Given a new bowling game

When all of my rolls are strikes

Then my total score should be 300

看着它失败

Unit Test Sessions - Session #1

Session #1 X

Tests failed: 2, passed: 0, ignored: 0

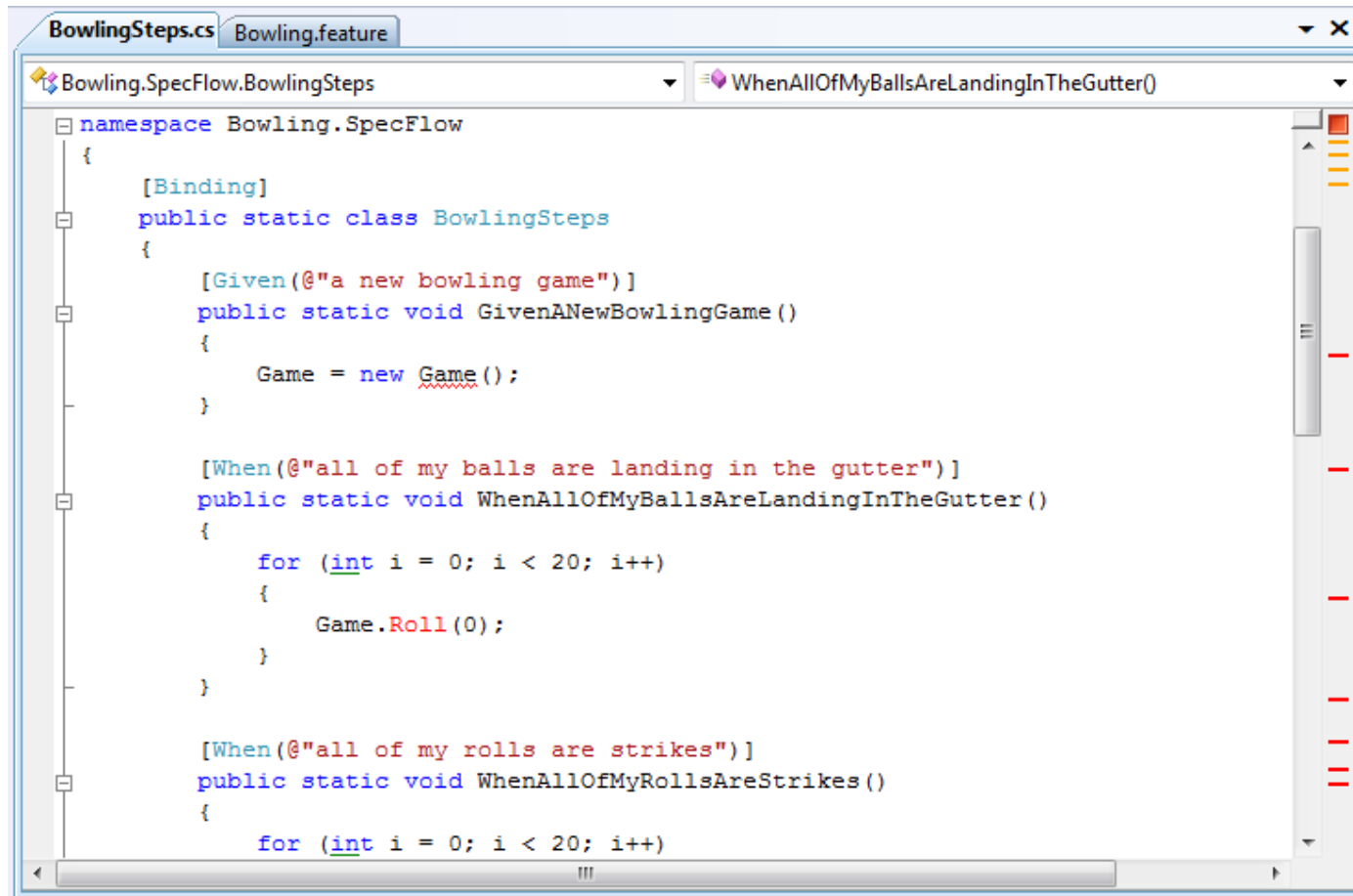
- <Bowling.specFlow> (2 tests)
 - { } Bowling.specFlow (2 tests)
 - CoreCalculationFixture (2 tests)
 - GutterGame
 - AllStrikes

CoreCalculationFixture.GutterGame : Failed

Given a new bowling game
-> No matching step definition found for the step. Use the following code to create one:
[Binding]
public static class StepDefinitons
{
 [Given(@"a new bowling game")]
 public static void GivenANewBowlingGame()
 {
 ScenarioContext.Current.Pending();
 }
}

When all of my balls are landing in the gutter
-> No matching step definition found for the step. Use the following code to create one:

实现步骤的定义

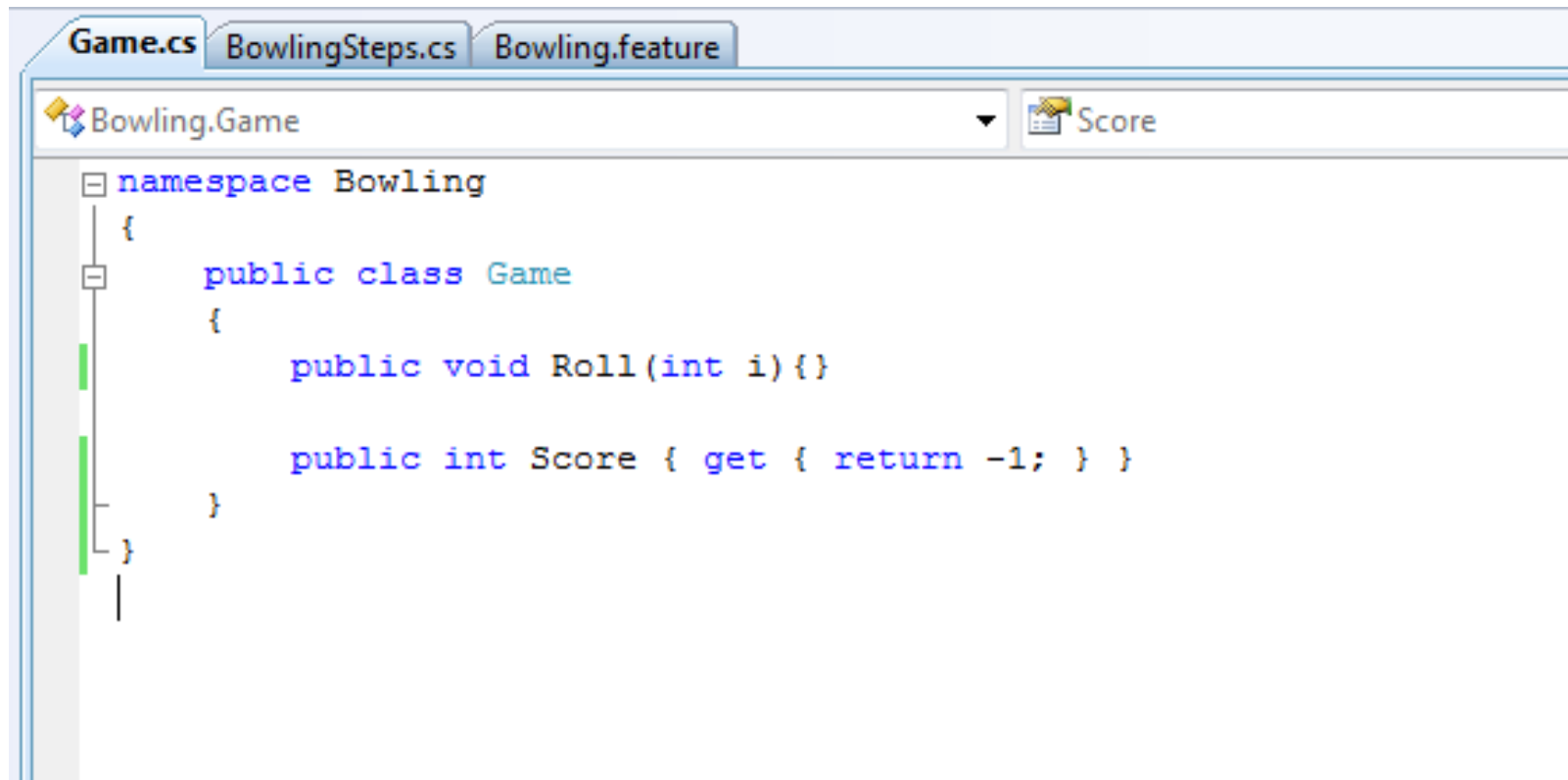


```
namespace Bowling.SpecFlow
{
    [Binding]
    public static class BowlingSteps
    {
        [Given(@"a new bowling game")]
        public static void GivenANewBowlingGame ()
        {
            Game = new Game ();
        }

        [When(@"all of my balls are landing in the gutter")]
        public static void WhenAllOfMyBallsAreLandingInTheGutter ()
        {
            for (int i = 0; i < 20; i++)
            {
                Game.Roll(0);
            }
        }

        [When(@"all of my rolls are strikes")]
        public static void WhenAllOfMyRollsAreStrikes ()
        {
            for (int i = 0; i < 20; i++)
```

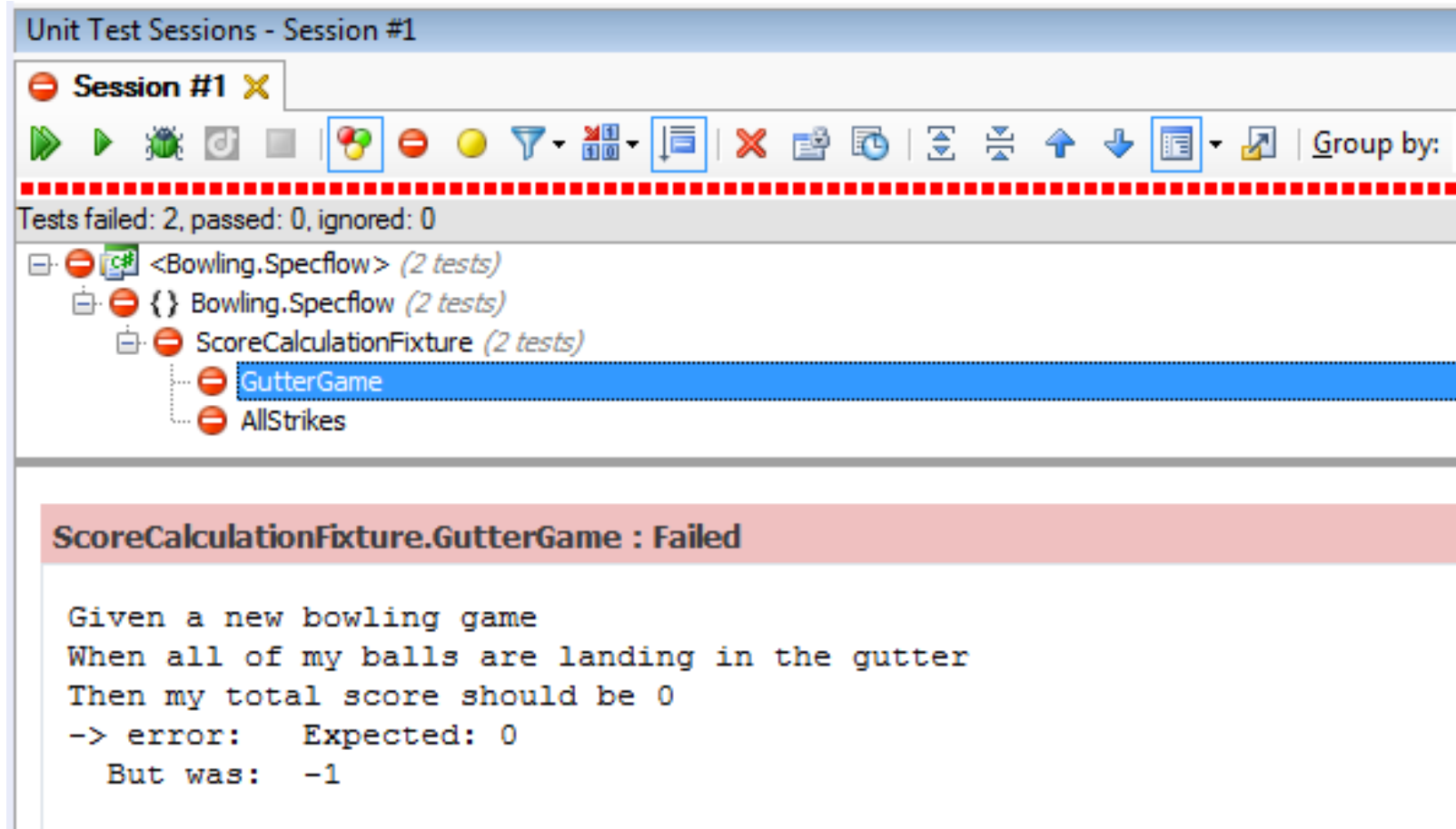
编写领域骨架



```
Game.cs BowlingSteps.cs Bowling.feature
Bowling.Game
namespace Bowling
{
    public class Game
    {
        public void Roll(int i) {}

        public int Score { get { return -1; } }
    }
}
```

看着它失败



Unit Test Sessions - Session #1

Session #1 X

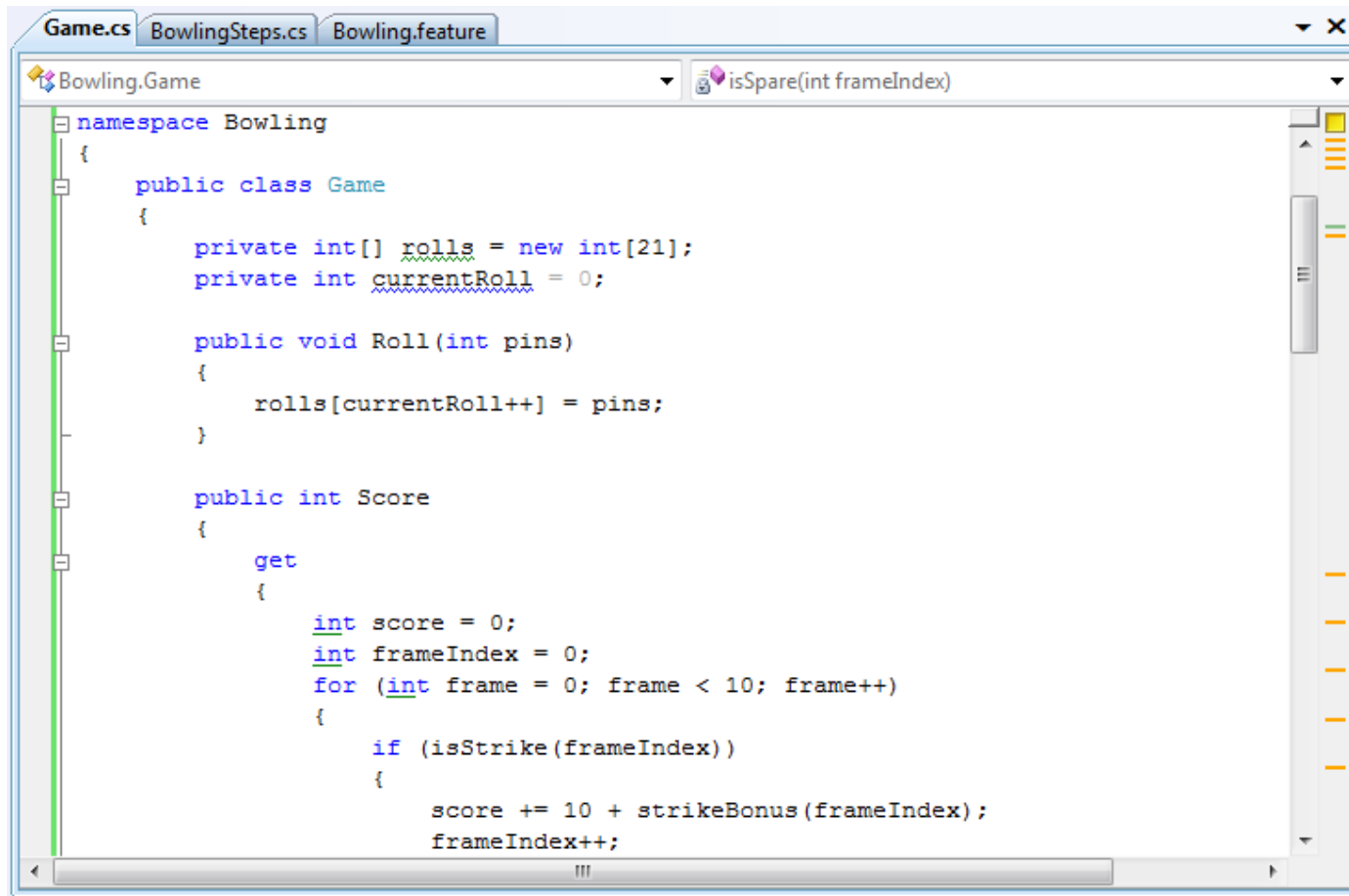
Tests failed: 2, passed: 0, ignored: 0

- <Bowling.Specflow> (2 tests)
 - { } Bowling.Specflow (2 tests)
 - ScoreCalculationFixture (2 tests)
 - GutterGame**
 - AllStrikes

ScoreCalculationFixture.GutterGame : Failed

```
Given a new bowling game
When all of my balls are landing in the gutter
Then my total score should be 0
-> error:   Expected: 0
           But was:  -1
```

实现领域功能

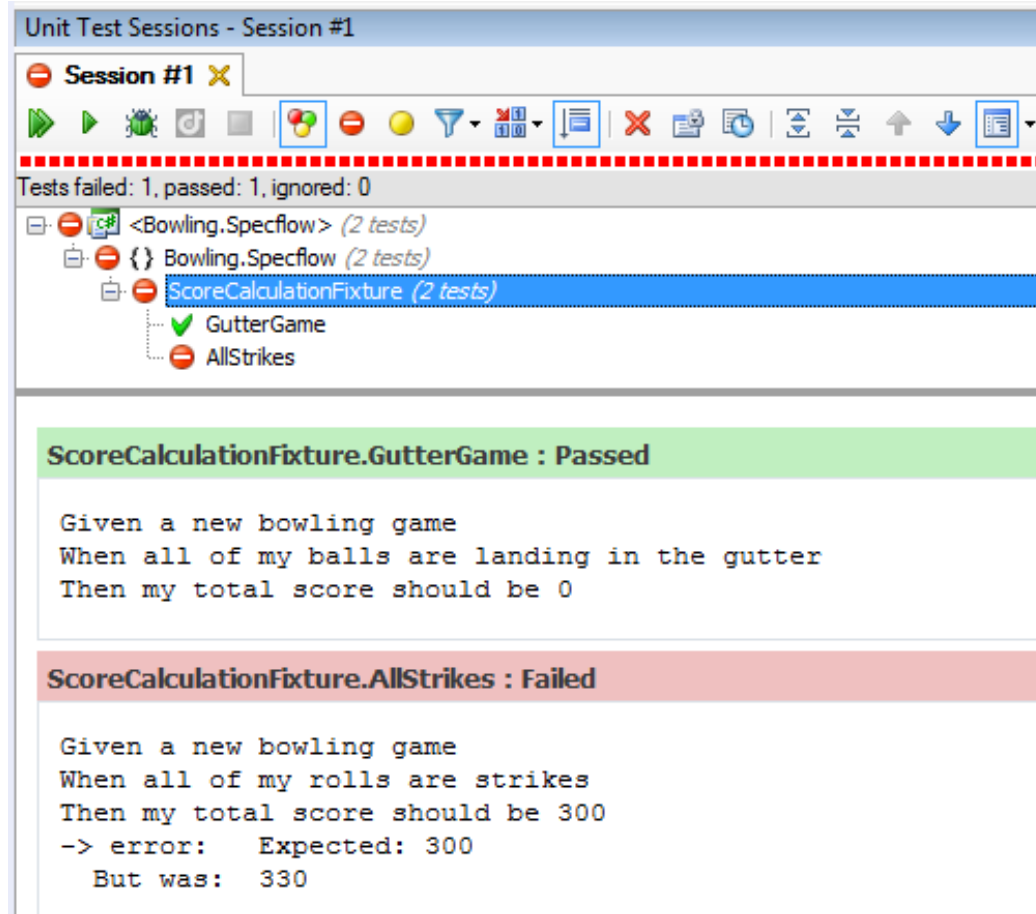


```
Game.cs BowlingSteps.cs Bowling.feature
Bowling.Game
isSpare(int frameIndex)
namespace Bowling
{
    public class Game
    {
        private int[] rolls = new int[21];
        private int currentRoll = 0;

        public void Roll(int pins)
        {
            rolls[currentRoll++] = pins;
        }

        public int Score
        {
            get
            {
                int score = 0;
                int frameIndex = 0;
                for (int frame = 0; frame < 10; frame++)
                {
                    if (isStrike(frameIndex))
                    {
                        score += 10 + strikeBonus(frameIndex);
                        frameIndex++;
                    }
                }
            }
        }
    }
}
```

重复这个循环



Unit Test Sessions - Session #1

Session #1

Tests failed: 1, passed: 1, ignored: 0

- <Bowling.Specflow> (2 tests)
 - Bowling.Specflow (2 tests)
 - ScoreCalculationFixture (2 tests)
 - GutterGame ✓ Passed
 - AllStrikes ✗ Failed

ScoreCalculationFixture.GutterGame : Passed

```
Given a new bowling game
When all of my balls are landing in the gutter
Then my total score should be 0
```

ScoreCalculationFixture.AllStrikes : Failed

```
Given a new bowling game
When all of my rolls are strikes
Then my total score should be 300
-> error:   Expected: 300
          But was:  330
```


直到功能完全实现

Unit Test Sessions - Session #1

✓ Session #1 ✕

Tests failed: 0, passed: 2, ignored: 0

- ✓ <Bowling.Specflow> (2 tests)
 - ✓ {} Bowling.Specflow (2 tests)
 - ✓ ScoreCalculationFixture (2 tests)
 - ✓ GutterGame
 - ✓ AllStrikes

ScoreCalculationFixture.GutterGame : Passed

```
Given a new bowling game
When all of my balls are landing in the gutter
Then my total score should be 0
```

ScoreCalculationFixture.AllStrikes : Passed

```
Given a new bowling game
When all of my rolls are strikes
Then my total score should be 300
```

最佳实践

编写高层次的规格

规格应该相对稳定

给独立的行为编写规格

用Given-When-Then的格式来设计规格

FITNESSE版本

eg.bowling.fixtures.FinalScore																
10	10	10	10	10	10	10	10	10	10	10	10					300

Now how about a slightly flawed game:

eg.bowling.fixtures.FinalScore																
10	10	4	10	10	10	10	10	10	10	10	10					262

And then finally, the sort of game you might ordinarily see. Well, I see it a

eg.bowling.fixtures.FinalScore																
5	7	9	10	4	3	0	8	10	6	7	9					91

FIXTURE的实现

```
public class FinalScore extends TableFixture {
    private BowlingGame game;
    protected void doStaticTable(int rows) {
        game = new BowlingGame();
        doRolls();
        doScore();
    }

    private void doRolls() {
        for(int i = 0; i < 21; i++) {
            if(!blank(0, i)) {
                int pins = getInt(0, i);
                game.roll(pins);
            }
        }
    }
}
```

```
private void doScore() {
    int expected = getInt(0, 21);
    int actual = game.score(10);
    if(actual == expected)
        right(0, 21);
    else
        wrong(0, 21, "" + actual);
}
```

进一步了解验收测试

<http://specflow.org/>

<http://cukes.info/>

<http://fitnesse.org/>

Q&A

麦宇安

<http://weibo.com/yuanmai>

ronald.mai@perficient.com