



Acceptance Testing in Plain  
English with Concordion.NET

**Gojko Adzic**  
<http://gojko.net>  
[gojko@gojko.net](mailto:gojko@gojko.net)  
[@gojkoadzic](#)

# Unit tests aren't really readable

```
@Test
public void testPlayersGetNewStatus() throws GameException {
    expectGameStart();
    Player p1 = new Player(BigDecimal.valueOf(17),BigDecimal.valueOf(17),"");
    Player p2 = new Player(BigDecimal.valueOf(18),BigDecimal.valueOf(18),"");
    EasyMock.expect(gameRules.getPlayers(gameStatus1)).andReturn(new Player[]{p1, p2}
    //ExecutionContext context = new ExecutionContext(GAME_ID, gameStatus1, v, "A", Tabl
    EasyMock.expect(gameRules.getStatusForPlayer(EasyMock.eq(p1), (ExecutionContext) Easy
    EasyMock.expect(gameRules.getStatusForPlayer(EasyMock.eq(p2), (ExecutionContext) Easy
    EasyMock.expect(gameRules.execute(new ExecutionContext(GAME_ID, gameStatus, v, "a", t
        (GameWalletImpl)EasyMock.anyObject()), EasyMock.eq(command))).andReturn(new E
    dd.notifyPlayer(EasyMock.eq(t.getId()),
        EasyMock.eq(BigDecimal.valueOf(17)),
        EasyMock.eq(new GameStatusDocument(1L, gameStatus1, "12345"))));
    EasyMock.expectLastCall();
    dd.notifyPlayer(EasyMock.eq(t.getId()),
        EasyMock.eq(BigDecimal.valueOf(18)),
        EasyMock.eq(new GameStatusDocument(1L, gameStatus, "12345"))));
    EasyMock.expectLastCall();
    EasyMock.replay(a, dd, gameStatus, gameStatus1, gameRules);
    executeCommand();
    EasyMock.verify(dd);
}
```

# Here's what I thought...

- When the order is in a “pending” state, we first check the account, and if approved move it to “confirmed” state
- When the order is manually confirmed, it moves from the “pending” to “confirmed” state even if the account does not have enough funds
- When the order is in a “pending” state for two days, we send an alert

# Here is what they saw:

ghorgh [the] [order] 'oH Daq [a] ["pending"] [state]  
maH wa'DIch [check] [the] [account] 'ej chugh  
[approved] vIH 'oH Daq ["confirmed"] [state]  
ghorgh [the] [order] 'oH [manually] [confirmed]  
'oH vIHtaH vo' [the] ["pending"] Daq  
["confirmed"] [state] 'ach chugh [the] [account]  
ta'taH ghobe' ghaj yap [funds] ghorgh [the]  
[order] 'oH Daq [a] ["pending"] [state] vaD cha'  
jajmey maH ngeH [an] [alert]

# 5 key things about Concordion.NET

- Acceptance testing tool
- Works on HTML documents
- Special HTML attributes mark inputs and expected outputs
- MBUnit-like fixtures link HTML and domain code
- Integrated with Gallio

# A simple test

```
<html
  xmlns:concordion="http://www.concordion.org/2007/concordion">
<head>
<link href="../../concordion.css" rel="stylesheet" type="text/css" />
</head>
<body>
<h1>String concatenator</h1>
<p>String concatenator joins two strings and adds a blank in
  between.</p>
<h2>Examples</h2>
<p>When <b concordion:set="#first">Hello</b> and <b
  concordion:set="#second">World</b> are joined, the result is <b
  concordion:assertEquals="Concatenate(#first, #second)">Hello
  World</b>.
</p>
</body>
</html>
```

# Fixture

```
[ConcordionTest]
public class ConcatenationTest
{
    public String Concatenate(String first, String second)
    {
        // this should go to some business code
        return first + " " + second;
    }
}
```

# A bit more structure

```
<div class="example">
  <h3>Examples</h3>
  <ul>
    <li concordion:execute="
      #offer = checkFreeDelivery(#type, #books)">
      A <b concordion:set="#type">VIP</b> customer with
        <b concordion:set="#books">9</b> books:
          <b concordion:assertEquals="#offer">not offered</b>.
    </li>
    <li concordion:execute="
      #offer = checkFreeDelivery(#type, #books)">
      A <b concordion:set="#type">Regular</b> customer with
        <b concordion:set="#books">9</b> books:
          <b concordion:assertEquals="#offer">not offered</b>.
    </li>
    . . . .
```

# End result

## Free delivery

Free delivery is offered to VIP customers with **10** books or more in their cart. Regular customers, and VIP customers with 9 books or fewer in their cart, do not receive free delivery.

### Examples

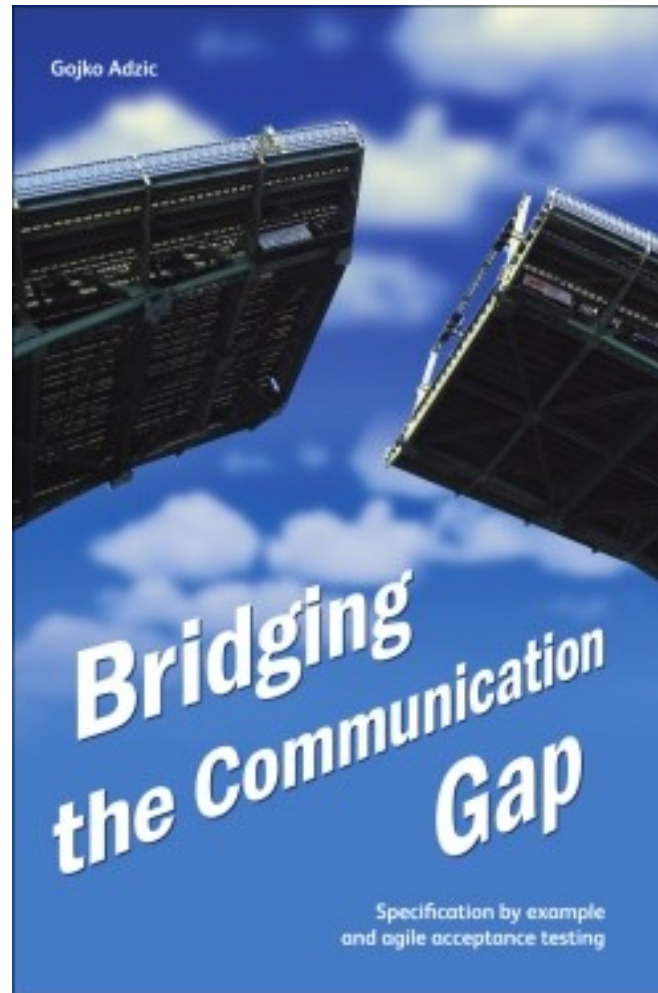
- A **VIP** customer with **9** books: ~~not offered~~ **offered**.
- A **Regular** customer with **9** books: **not offered**.
- A **VIP** customer with **10** books: **offered**.
- A **Regular** customer with **10** books: **not offered**.
- A **VIP** customer with **20** books: **offered**.
- A **Regular** customer with **20** books: **not offered**.

# Compared to FIT

- Free-form text, not tables
- Easy IDE/CI integration
- No test management tool
- No domain helpers

# Links

- <http://gojko.net>
- <http://code.google.com/p/concordion-net/>
- <http://concordion.org/>



gojko@gojko.com  
<http://gojko.net>  
@gojkoadzic