The Craftsman: 41
Dosage Tracking XVIII
Yeah, Sorta.

Robert C. Martin
23 August 2005

...Continued from last month.

In April of 1944, the three leaders of the Axis, Hitler, Stalin, and Tojo (Mussolini's status had declined of late) traveled to the Canary Islands to witness the launch of "Operation Underbelly", the Mexican Invasion. They were expecting to make very short work of Mexico and then to plow through the breadbasket of the United States.

At exactly *8pm on the eve of the launch, the sky directly above the Canary Islands suddenly blazed forth with actinic fury. For a fraction of a second the landscape was bathed in noon-day light. Anyone who happened to be looking straight up at the time was dazzled as though they had looked at the sun. There was no sound, no thunder, indeed, the silence was eerie. A second or so after the flash the sky took on a strange green glow that faded into a blood-red aura spreading across the horizon for several minutes.*

In the hotel where the three Axis leaders were enjoying an evening meal, the electrical power failed, and telephone and radio communications were disrupted for a few minutes. When communications were restored the leaders learned that similar events had occurred directly above Berlin, Tokyo, and Moscow.

---

22 Feb 2002, 1000

"Damn, when did we break that unit test?" I was pretty embarrassed that I had allowed the tests fail. I made a mental note to run the tests more frequently.

Jasper's big toothy grin and his condescending demeanor continued to grate on me. "Why don't you check the error, Alphonse?"

The error said:

```
Expected:Suit Registration
Actual  :dtrack.messages.SuitRegistrationApprovalRequest
```

I clicked on the error and it took me to the failing test. (I've marked the failing line with an arrow.)

```
public class ManufacturingTest extends TestCase {
  public void testRegisterSuitSendsMessageToMfg() throws Exception {
    MockManufacturing mfg = new MockManufacturing();
    mfg.requestApprovalForRegistration(7734);
```

```
    SuitRegistrationApprovalRequest message =
              (SuitRegistrationApprovalRequest) mfg.getLastMessage();
 -> assertEquals("Suit Registration", message.id);
    assertEquals("Outside Maintenance", message.sender);
    assertEquals(7734, message.argument);
  }
}
```

"Oh, OK." I said. "When I changed the ID of the message, I forgot to change the test. That's easy to fix." So I changed the test as follows.

```
    assertEquals("dtrack.messages.SuitRegistrationApprovalRequest",
                 message.id);
```

I ran the unit tests and they all passed.

But Jasper didn't like it, and he said so: "I don't like it Alphonse."

"Why not, Jasper? It works."

"Sure it works, Alphonse, but if the message id changes again, the test will break again."

"What do you suggest we do about it, Jasper?"

"How about this?" And with that, Jasper took the keyboard and changed the test to this:

```
    assertEquals(SuitRegistrationApprovalRequest.ID, message.id);
```

This didn't compile because the `ID` field wasn't public. Jasper made than simple change than then ran all the tests. They all passed.

Then Jasper passed the keyboard back to me, locked my gaze with those sparking eyes of his, and gave me a quick double eyebrow raise. He clearly thought that his change was very clever. I was beginning to wonder just what it took to become a Journeyman around here. If Jasper was any indication, Mr. C's standards must be pretty low.

I looked back at the screen and tried to focus on my job. "OK, I see your point. It's better not to embed constants in the code if you can avoid it. Now, let's see why so many of the acceptance tests are failing."

"I'll bet it's for the same reason." Jasper quipped.

He was probably right. The acceptance tests probably also mentioned the message ID by name.

"Yeah." I said, and brought up the first failing acceptance test.

Message sent to manufacturing message id?	message argument?	message sender?
Suit Registration expected dtrack.messages.SuitRegistrationApprovalRequest actual	314159	Outside Maintenance

"You see!" Cried Jasper. "It's the same issue."

"Yes, I see Jasper. You are right. It's the same issue." I gave a deep sigh and shook my head.

Jasper noticed my mood and hung his head. "I'm sorry Alphonse, I'm doing it again aren't I."

"Yeah, sorta." I said, without looking at him.

"OK, look, I'll try to back off. I guess I'm just easily excited."

I gave a snort, looked over at him with a grin and said: "Yeah, sorta!".

He smiled back, still embarrassed, and then we both turned back to the code.

"OK, Jasper, I think we need to apply your solution again. Instead of just changing the string in the test table, we need to see if the ID in the message equals the contents of the static ID variable in the class."

"Yeah, I agree. Here's something Jean showed me a few months back." And he took the keyboard and began to modify the test tables.

```
!|Message sent to manufacturing|
|Suit registration approval request?|message argument?|message sender?    |
|true                               |314159           |Outside Maintenance|
```

"OK, I see." I said. "We're simply asking whether or not the message is the right kind. We aren't even mentioning the notion of an ID."

"Right. There's no point in exposing a detail like the message ID, especially if it's likely to change."

Jasper saved the page and ran the test. Of course it complained.

"OK, now we have to add the `SuitRegistrationApprovalRequest` method to the fixture." Said Jasper.

"I'll do it." And I grabbed the keyboard and made the appropriate changes to the fixture.

```
public class MessageSentToManufacturing extends ColumnFixture {
  ...

  public boolean suitRegistrationApprovalRequest() {
    return message.id.equals(SuitRegistrationApprovalRequest.ID);
  }

  ...
}
```

I saved this and ran the test, and it passed.

For a second Jasper's eyes showed the same old excitement, but he reigned it back in and simply said: "OK, good. Now how about the rest of the acceptance tests?"

We found another that was failing for the same reason, and quickly fixed it. That left one more failing acceptance test: `RejectDuplicateRegistration`.

"Oh yeah!" I said happily. "That's what we started working on this morning. This one should be failing because we haven't finished getting it working yet."

fitlibrary.DoFixture			
start	dtrack.fixtures.DTrackFixture		
set suit	314159	as registered	
check	register suit	314159	false expected / true actual
check	was a message sent to manufacturing	false expected / true actual	
check	count of registered suits is	1	
check	error message	Suit 314159 already registered.	true expected / false actual

"OK." I said. "This is failing because the `Registrar` is not detecting that suit# 314159 has already been registered. We should be able to fix that pretty easily."

"Be my guest." Said Jasper, still somewhat subdued.

So I brought up the code for the `Registrar` and found the appropriate method.

```
public static boolean attemptToRegisterNewSuit(int barcode) {
  return Utilities.manufacturing.requestApprovalForRegistration(barcode);
}
```

"Yeah, see, it's just passing the request to manufacturing without checking."

So I modified the code as follows:

```
public static boolean attemptToRegisterNewSuit(int barcode) {
  if (SuitGateway.isSuitRegistered(barcode))
    return false;
  return Utilities.manufacturing.requestApprovalForRegistration(barcode);
}
```

I talked while I typed. "OK, now we need to write the `IsSuitRegistered` method."

Jasper stopped me. "No, Alphonse. Now we need to write the unit test for `IsSuitRegistered`."

"Right! I almost forgot." So I looked for the appropriate unit test to modify, but couldn't fine one.

"Wow, it looks like we wrote those gateways without any unit tests!" I brought up the `InMemorySuitGateway` that we'd been using.

```
public class InMemorySuitGateway implements ISuitGateway {
  private Map suits = new HashMap();
  public void add(Suit suit) {
    suits.put(new Integer(suit.barCode()), suit);
  }

  public int getNumberOfSuits() {
    return suits.size();
  }

  public Suit[] getArrayOfSuits() {
    return (Suit[]) suits.values().toArray(new Suit[0]);
  }
}
```

Jasper looked at the code and said: "OK, this is pretty simple code. I can see why the unit tests didn't get written, but it's time to write one now." And he grabbed the keyboard and started to write the test.

```
public class InMemorySuiteGatewayTest extends TestCase {
  public void testIsSuitRegistered() throws Exception {
    InMemorySuitGateway g = new InMemorySuitGateway();
    assertFalse(g.isSuitRegistered(314159));
    g.add(new Suit(314159, new Date()));
    assertTrue(g.isSuitRegistered(314159));
  }
}
```

Then, step by step, he wrote the implementation.

```
public boolean IsSuitRegistered(int barcode) {
  return suits.containsKey(barcode);
}
```

The unit tests all passed.

"OK, now we have to connect that to the `SuitGateway` class."

```
public interface ISuitGateway {
  ...
  boolean isSuitRegistered(int barcode);
}

public class SuitGateway {
  public static ISuitGateway instance = null;

  ...

  public static boolean isSuitRegistered(int barcode) {
    return instance.isSuitRegistered(barcode);
  }
}
```

We ran the unit tests, and they all passed. Then we ran the `RejectDuplicateRegistration` acceptance test.

fitlibrary.DoFixture			
start	dtrack.fixtures.DTrackFixture		
set suit	314159	as registered	
check	register suit	314159	false
check	was a message sent to manufacturing	false expected true actual	
check	count of registered suits is	1	
check	error message	Suit 314159 already registered.	true expected false actual

"Great!" I said. "That first failing cell of the test is now passing. Only two more to go."

Jasper looked up at me and said: "I need a break. Let's go watch the starbow for awhile."

"Sounds like a plan." Working with Jasper was tiring. Perhaps he found working with me to be tiring too.

The code for this article can be located at:

`http://www.objectmentor.com/resources/articles/CraftsmanCode/Craftsman_40_DosageTrackingSystem.zip`

To be continued...