The Craftsman: 32
Dosage Tracking IX
Indubitably!

Robert C. Martin
16 November 2004

...Continued from last month.

In August of 1939, Hitler and Stalin signed a non-aggression pact. Though the pact between such ideological enemies was very unpopular with European and American Communists, the practical result was that by the end of 1940 the USSR had seized the eastern half of Poland and annexed the Baltic States; vastly expanding it's empire and creating a long shared border with Germany.

Over the years Stalin had deeply insinuated an espionage network within the highest levels of the United States government. One member of this network, Whittaker Chambers, was so outraged by the growing "friendliness" between Stalin and Hitler that he defected to the U.S. in 1938. Chambers provided the Roosevelt administration with details on dozens of other communist spies. One of these spies, Alger Hiss, was a top State Department official and a trusted friend of FDR. FDR refused to believe the accusations against Hiss and denounced Chambers instead – allowing Stalin's network to continue operating through 1940.

So Stalin knew all about Nimbus, including the Atomic Bomb research and Von Braun's rocketry successes. Stalin also knew he could not defend against the growing US threat alone. So in the early months of 1941 he negotiated a deep alliance with Hitler and the Axis powers, offering some of the Nimbus intelligence as part of the bargain.

The US had a monopoly on the great minds, but their secrets had been sold, and they had lost the element of surprise. Now a great Eurasian power was rapidly growing, and aligning against them.

---

21 Feb 2002, 1430

Jean didn't give us a minute to respond. "Avery and I have had a nice little chat, and I think we've got everything worked out. Don't you agree Avery? I know you're a fine young man, just a bit high spirited really. You wouldn't make such a fuss if you didn't *care*, would you? Of course you wouldn't. Anyway, Jerry and Alphonse, I'll leave Avery in your fine care and go on about the rest of my day. My goodness, but don't I have a lot of little details to take care of! It never ends, boys, it never ends." And she picked up her knitting basket and shambled off to chat with Carole.

Avery just stood there looking sheepish while Jerry and I put our hands in our pockets and looked at the floor. Finally Jerry said: "I'm glad you're out of there Avery. It's no fun in there – I know. Look, I have to go work with Jasper for awhile. Alphonse, can you and Avery finish up this test page?"

He was going to leave me alone with Avery after the woodshed! What would I say to him? "Sure, Jerry, we can work on it."

"Great, thanks. I'll be back in about an hour or so." And Jerry walked away.

"You were in there a long time." I said to Avery. "Was it bad? What did she say? Did she yell at you? I felt awful watching you in there."

"Yeah, well, Jerry's right about it not being fun. She didn't actually yell, but... Look, I probably shouldn't talk all that much about it. She made me realize a few things about myself that I don't like. I'm going to think about them for awhile. Perhaps I'll be able to tell you more later, but now I'd just like to get back to work; if you don't mind."

"Oh, yeah, sure. No problem. I'll show you what we've been doing."

"...and Alphonse, it means a lot that you were concerned about me. Thanks."

"Er, sure... uh, so here's the thing. Jerry and I have been working on the Register Normal Suit page. We just got the first failing table to pass. You and I should get the second failing table to pass."

I pointed to the page and Avery studied it.

Normal suit registration.

```
Import
dtrack.fixtures
```

We assume that today is 2/21/2002.

```
DTrack Context
Today's date
2/21/2002
```

We also assume that there are no suits in inventory.

```
Suit inventory parameters
Number of suits?
0
```

We register suit 314159.

```
Suit Registration Request
bar code
314159
```

DTrack sends the registration confirmation to Manufacturing.

```
Message sent to manufacturing
message id?                  message argument?   message sender?
Suit Registration expected   314159 expected     Outside Maintenance expected
null actual                  0 actual            null actual
```

Manufacturing accepts the confirmation.

```
Message received from manufacturing
message id                  message argument   message sender   message recipient
Suit Registration Accepted  314159             Manufacturing    Outside Maintenance
```

And now the suit is in inventory, and is scheduled for immediate inspection

```
Suits in inventory
bar code?          next inspection date?
314159 missing     2/21/2002
```

"OK, so we want to get that Message sent to manufacturing table working?"

"Right." I said. We want to make sure that when we register the suit in the Suit Registration Request table, a message gets sent to manufacturing with the registration information."

OK, well it looks real easy to get that to pass. All we have to do is modify the `getLastMessageTo-Manufacturing()` method as follows:

```
public class Utilities {
  ...
  public static Object getLastMessageToManufacturing() {
    SuitRegistrationMessage message = new SuitRegistrationMessage();
    message.sender = "Outside Maintenance";
    message.id = "Suit Registration";
    message.argument = 314159;
    return message;
  }
  ...
}
```

Sure enough the table turned green.

Message sent to manufacturing		
message id?	message argument?	message sender?
Suit Registration	314159	Outside Maintenance

"Yeah, that's what I thought too. But Jerry told me not to do that. He said that it was OK to do the simplest thing to get a *unit* test to pass; but not an acceptance test. He said that if you are tempted to do something too simple to get an acceptance test to pass, you should write some unit tests instead."

Avery looked confused. "What kind of unit test should we write?"

"Well, we want to make sure that we send a message to manufacturing when a suit is registered."

"Yeah, but that's what the acceptance test checks."

He had a point. "You have a point."

"Yeah, do we want to write a unit test that checks the exact same thing as the acceptance test?"

He had another point. "You have another point."

"So what do we do?" Avery was clearly trying to fight his incredulity. If he burst out again so soon after the woodshed, Jean might just leave him in there permanently.

"Look, writing a unit test is no big deal. If it happens to overlap with the acceptance test, then so be it. Let's just write the unit test and then show Jerry when he gets back."

"OK, whatever you say Alphonse, but I'll write it if you don't mind."

"Sure, go ahead."

So Avery grabbed the keyboard and wrote:

```
public class UtilitiesTest extends TestCase {
  ...
  public void testRegisterSuitSendsMessageToMfg() throws Exception {
    Utilities.registerSuit(7734);
    SuitRegistrationMessage message =
      (SuitRegistrationMessage) Utilities.getLastMessageToManufacturing();
    assertEquals("Suit Registration", message.id);
    assertEquals("Outside Maintenance", message.sender);
    assertEquals(7734, message.argument);
  }
}
```

This failed for the following reason:

```
expected:<7734> but was:<314159>
```

"OK, now if we follow Jerry's rule and make this fail the simplest way possible we have to change `getLastMessageFromManufacturing` again."

```
  public static Object getLastMessageToManufacturing() {
    SuitRegistrationMessage message = new SuitRegistrationMessage();
    message.sender = "Outside Maintenance";
    message.id = "Suit Registration";
    message.argument = 7734;
    return message;
  }
```

I clicked the unit test and the acceptance test. "OK, now the unit test passes, but the acceptance test fails."

"Hmmm." said Avery.

"Yes!" I responded.

"My point exactly!" said Avery.

"Indeed!" I responded.

We looked at each other and laughed for a second.

"OK, I think I'm seeing Jerry's logic." Avery said. "To get both tests to pass, we have to do somethign intelligent. We can't just keep doing the simplest thing."

"Do you have something in mind?"

"Yeah, watch this." And Avery started typing again.

```
public class Utilities {
  ...
  private static Manufacturing manufacturing = new MockManufacturing();
  ...
  public static void registerSuit(int barCode) {
    manufacturing.registerSuit(barCode);
  }

  public static Object getLastMessageToManufacturing() {
    SuitRegistrationMessage message =
      (SuitRegistrationMessage) manufacturing.getLastMessage();
    return message;
  }
  ...
}
```

```
package dtrack.external;

public interface Manufacturing {
  public void registerSuit(int barCode);
  public Object getLastMessage();
}
```

```
package dtrack.mocks;

import dtrack.external.Manufacturing;
import dtrack.messages.SuitRegistrationMessage;

public class MockManufacturing implements Manufacturing {
  private Object lastMessage;

  public void registerSuit(int barCode) {
    SuitRegistrationMessage msg = new SuitRegistrationMessage();
    msg.id = "Suit Registration";
```

```
    msg.sender = "Outside Maintenance";
    msg.argument = barCode;
    lastMessage = msg;
  }

  public Object getLastMessage() {
    return lastMessage;
  }
}
```

"Wow." I said in admiration as I pushed the test buttons. The unit tests pass now, and so does the Message to manufacturing table."

"Yeah." Said Avery. But his brow was furrowed. "I don't like this. The Mock object shouldn't be building the message. That should be done by the real Manufacturing object. I also don't like that argument variable in the message, or the fact that this Utilities class is turning in to one big hodge podge. This code is a mess. It really needs to be cleaned up."

"Yeah, I agree. I said the same thing to Jerry just before you came back."

"Should we clean it up before he gets back here?"

I was a little worried about that. I didn't want to get Jerry mad by changing things that he wasn't ready to change. On the other hand, the code *was* a mess, and if we didn't clean it he'd be just as likely to be mad about that.

"OK." I said. "Let's see how much damage we can do before Jerry gets back here!"

"Indeed!"

"Indubitably!"

To be continued...

The source code for this article can be found at:

`www.objectmentor.com/resources/articles/CraftsmanCode/Craftsman_32_DosageTrackingSystem.zip`