CS31310: Test Driven Development and Pair Programming Session Reflection

Edgar Ivanov
edi@aber.ac.uk
Department of Computer Science, Aberystwyth University

December 16, 2013

We have recently had a workshop session; we were asked to write some code in pairs using TDD technique. In the beginning I worked with my course mate with very limited knowledge of Java; this fact influenced our decision when picking up the roles. We decided that I would take the navigator role and my friend would be in a driver role. We then discussed how our application might look like and what we could write the tests for. The simplest thing we could think of was to test the creation of the new fruit objects (banana, apple, orange) and the default price value assigned to them. While my friend was writing the tests I was following on the screen everything that was typed in to make sure that there were no mistakes in the code and that he followed our plan. In the beginning it was a bit difficult to concentrate on writing tests since it was tempting to write the code that was required for the implementation first and then write the actual tests. When we finished coding tests we ran them to make sure that they would actually fail. We then coded the simplest implementations for our fruit classes that would allow them to pass the tests. While coding the tests I thought that there was probably no point in testing such a simple behaviour and we should move straight to adding objects to the list and then testing if the total value of all fruits in cart matches the expected one; however, it was all about getting an overall idea of how things should be done and getting the experience. While writing the tests we were continuously sharing thoughts of what else could be tested, which was pretty helpful, because both of us came up with a few good ideas and throughout the discussion we rounded them up.

With regards to the "red-green-refactor" approach we have only used the first two. There was no need to refactor anything in the fruit classes; they were very simple - just returning value of the variable. Later on, however, we had to move some functionality to the superclass to make the code management easier.

When we changed pairs I explained to my new partner what our code was doing and what tests we have implemented so far. As I found out later my new partner was much more experienced in Java than me. He was a bit quicker in making design decisions and I was asking questions with regards to them.

In overall it was an interesting session, but I have some contradictory feelings about TDD. It seems to be a waist of time testing trivial things like a creation of the new object. I could not understand the reason for writing the tests for each and every method in the class, like we did in the workshop session. In my opinion TDD technique is not the most suitable for the small projects with a few classes where testing can be done by simply running an application.

Pair programming seems to be very efficient when both participants are equally knowledgeable and quickly catch up with what is going on. But if one of them has more knowledge that the other then he needs to explain what is being done to his colleague, otherwise there is no point in carrying the work together since one partner will not be able to understand the code and provide help when an issue arises.