Testing from the ground up

Software Hut 2008

Tell me about testing

- How do you test?
- What do you test?
- Why do you test?

Why do we test?

- Nobody writes perfect code.
- The client wants to be sure that the product works.
- We want to be sure that when we make changes, these don't break old code.
- Marks – one of the parts of the XP process mark is evidence of regular testing.

Question 1

- Is it testing if I write a class, link it into my system, and run it?

Testing demonstrates correctness

- By running your code you do not demonstrate correctness.
- You must document what the *expected* (as derived from a story card) output is, given your input.
- You must document if the test passed or failed.

Question 2

- Is debugging testing?

Tests are recorded

- You have some tests for your code, which you run and they are passed.
- If you then find a bug, your tests are obviously not good enough!
  - In fact you have demonstrated a new test case.
- So modify your tests to test for this new issue, as well as fixing the issue.

Question 3

- How many tests do I need for each function?

The number of tests depends on the function

- Example one:

```
Public boolean willMarry(int love) {
    if (love > 10) return true;
    else return false;
}
```

- Example two:

```
Public String hello(String name) {
    return "Hello "+name;
}
```

Question 4

- Can I test functions which don't return a value?

Functions that don't return a value

- Some functions, like 'setter' methods don't return a value.
- Other functions like those writing to database return a value, but this may not allow you to test if the value was correctly written.
- So to test them you must use another method to check that the state of the object/database has been changed correctly.

Question 5

- Is testing valid if it fails?

Tests that fail

- You should expect some or all (if you are using test first) of your tests to fail.
- This shows where there are errors in your code.
- This is a valid result because you have demonstrated a fault. This is much better than not finding the fault!

Question 6

- How can you test an interface automatically?

Testing interfaces

- To do this you need to use a specialised tool. For web based applications we have selected Selenium for you, there are some possibilities for java (ask Chris).
- To work out what to test, create an Extreme X-Machine. If you use the Eclipse tool it will generate a list of tests for you.

Methods of testing

- Manual testing
  - Requires a test plan
  - Can be time consuming
- Automated unit/interface testing
  - May take longer to set up a first
  - But is quick to run
  - It automatically records the results of tests

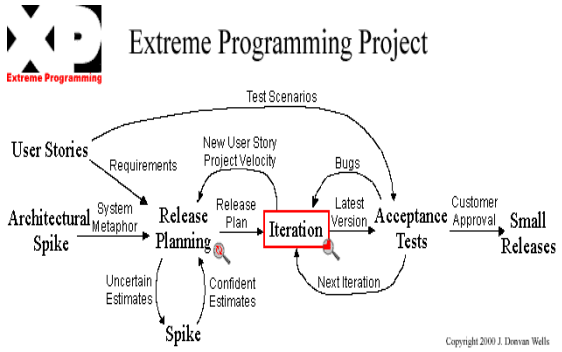
What is test first?

- *"There is a rhythm to developing software unit test first. You create one test to define some small aspect of the problem at hand. Then you create the simplest code that will make that test pass. Then you create a second test. Now you add to the code you just created to make this new test pass, but no more! Not until you have yet a third test. You continue until there is nothing left to test."*

(Source extremeprogramming.org)

Testing in XP

- In XP you should be running **all**(!) your tests very frequently.
  - In test first after you implement each method.
  - In test last after you implement a story.
- It is going to be time consuming to do this manually, so you need to write unit tests.



(Source extremeprogramming.org)

Test first

- Benefits:
  - You know when your code is correct.
  - No need to design for test.
  - Testing is creative.
- Drawbacks:
  - You may not be clear about what your code has to do.
  - Must use black box testing on the specification (story cards).

Design for test

- Some types of application are untestable by design!
- The solution is to design your code to be testable.
- If you actually write your tests first, you can be sure that you test everything.
- PHP is especially problematic.

Writing tests before code

- Identify the functionality required:
  - Examine the story cards.
  - Identify user interfaces, data to be stored, and jobs to do.
  - Each of these will need a separate class in the implementation.
- Write a function stub in your class (i.e. just empty methods that return default values).
- Write a test method to test this stubs.
- Run your tests, they should(!) all fail.
- Write the code.
- Run the tests – they should all pass.
- Rinse and repeat!

Use unit test frameworks

- JUnit – Java
- RUnit – Ruby
- PhpUnit / Simple Test – PHP
- nUnit – C#/ASP
- Or define your own.

JUnit Exercise

- **Story:** *The bank account stores any currency, the customer can deposit and withdraw funds, interest is credited or debited from the account every month.*

Mock objects

- Some things are hard to test: For example what if you depend on some external service?
- In this case make a mock object.
- Pass the mock object as a construction argument into the class to be tested. Instead of the real object!

Use toolkits with scaffolding

- By using a toolkit, some unit testing comes for free!
  - Symphony – for PHP
  - RubyOnRails – for Ruby
  - Grails – for Groovy
  - Castle – for .net
  - Trestle – for Java

How do I know I am doing well?

- https://devcreek.com/index.html/

Next weeks hand in

- In you team directory:
  - UNIX: /share/softhut/softhut08
  - Windows: \\juniper\softhut\softhut08.
- Create a directory called:
  - Release
- Add a directory under this called:
  - Week3
- Put the code you demo to the client in here.
- For each of the subsequent releases add a new directory and copy your code in.
- Create another directory called:
  - working.
- Use this to store a copy of all your current working code, and documents. Add directories called: (as required)
  - documents
  - apache
  - java
  - tomcat
- You should update this at least once a week.

Filling in minutes

- Some people are unsure what to use. At the movement I would expect you to be mainly using the “story card” option as the clients will be telling you about their stories.
- If you want to discuss a particular meeting or thing that happened, give me an email explaining it and I’ll explain how to enter it.

Additional comments

- You are going to be marked individually on the things you do.
- We will identify individual contribution by comments on code and document – which verify the list on the wiki:
- When you write a class, put a comment at the top of the code and on every function to identify you as the author. When you debug or modify a class add a comment at the top and on each modified function to identify the changes made, and you as the author.
- When you write a document, put your name at the top of it.
- And so on…

Nominate the Hut

- http://www.shef.ac.uk/cilass/awardscheme.html
- The university is handing out small awards to courses like the hut, if we get the award we will use it on enhance future Software Huts.
- Please nominate us!

Need help testing?

- Come to the sessions in the lab and ask the demonstrators.
- Or, email c.thomson@dcs.shef.ac.uk to arrange an appointment.