

A brief introduction to Test Driven Development with Excel, for beginners.

Introduction

Test Driven Development (TDD) is one of the most extraordinary techniques to come out of eXtreme Programming (XP). It's easy enough to describe - write a failing test, make the test run, refactor the code so the design is cleaner - but it's not until you do it that it's power comes through.

Here's a simple TDD example - converting integers into their roman number equivalent - that you can do at your own PC using Excel. We put the test data into a spreadsheet (1=i, 2=ii, etc) and then we write our own function in Excel using Visual Basic for Applications. It will help if you've had a little programming experience in the past ... but not much.

All you need is Excel, an open mind and an hour or two to experiment.

BTW: I used to be a good programmer but I haven't coded since the 90's so please forgive me any deficiencies below. It's the TDD process that is important, not the coding detail.

First test and testing environment

In this section we (a) write our first test and (b) set up a testing environment

I start with a blank spreadsheet.

- I type column headings in row 1 ("Integer" in Column A and "Roman" in column B).
- I add my first row of test data in row 2, using the test case that 1=i.

Sheet1:

Integer	Roman
1	I

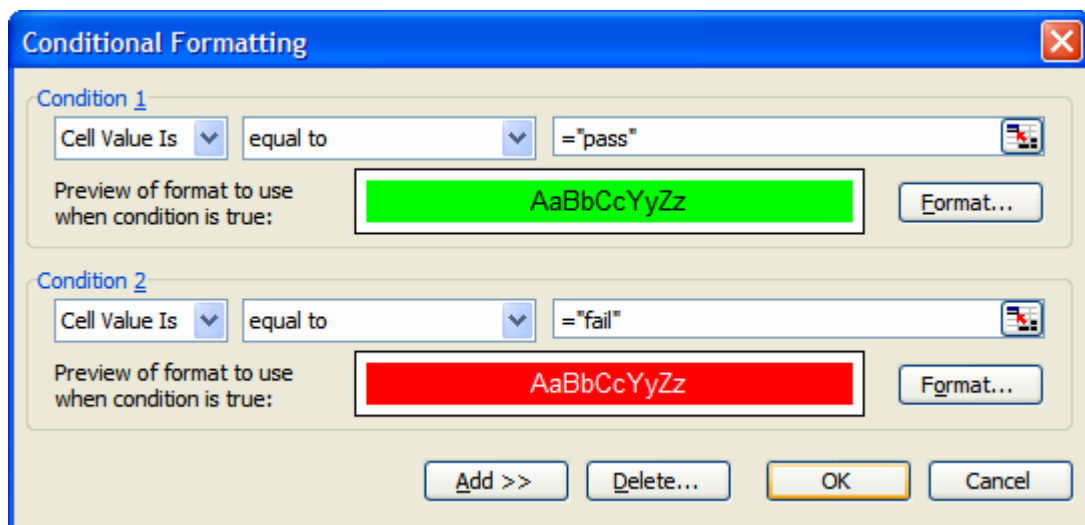
- Then I add another heading into cell C1 called "i2r" and I use in cell C2 I type the formula '=i2r(A2)'. This is the function that I'm going to build, test-case by test-case.

But, since the function i2r() doesn't exist yet, excel gives an error. This is the same as a compile error in, say, Java.

Integer	Roman	i2r
I	1	#NAME?

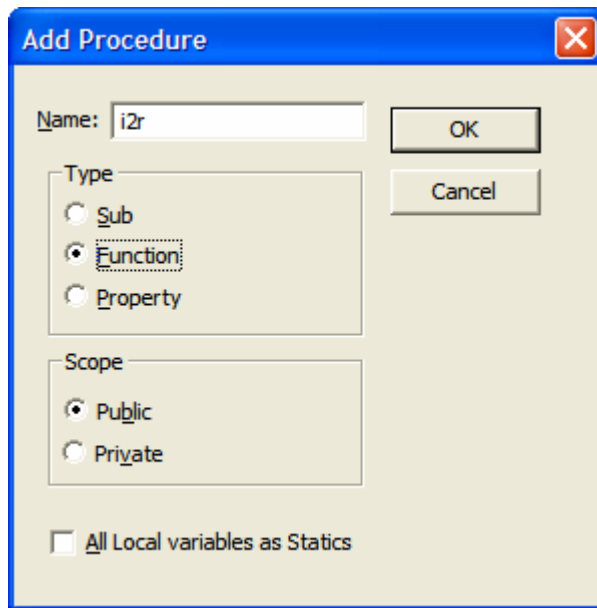
I want to use column D as my automated testing column. I want it to show **Green** if the test works and **Red** (with white ink) if it fails. That way the spreadsheet should shout out at me if tests are failing.

- So I add another heading Pass/Fail and add the formula =IF(B2=C2,"pass","fail") into cell D2.
- But, since C2 is already showing an error, Excel also shows an error in D2.
- I quickly overtype cell C2 with "i" to check it works with a pass and it does. Then I put in the word "blah" to check that the fail works and it does to. I undo both of those and play around with Excel's conditional formatting feature and make "true" show in Green background and "false" show in Red Background.



I now have a testing environment, of sorts, and a failing test. I need to add a new function called i2r() before it will work.

- So I chose Tools/Macros/Visual Basic Editor which, unsurprisingly, opened up the visual basic editor. I could also have pressed ALT-F11. I then hunt around the menus and find INSERT/PROCEDURE which sounded handy - I presume that this is how I insert a new function (not just a procedure).
- Bad news: it's greyed out.
- But, just underneath it is INSERT/MODULE which I bravely click and it gives me a new module. I check under the INSERT menu and PROCEDURE is now enabled. So I click on it and add in a new function (not procedure) - naming it i2r().



Then I add in an integer parameter to the function and (lazily) call it i.

```
Public Function i2r(i As Integer)
End Function
```

I switch back to my spreadsheet. Nothing's changed - the #NAME? is still displayed.

- I guess that Excel didn't realise that I'd updated the function so I try manually recalculating the spreadsheet by pressing F9 and it shows me:

Integer	Roman	i2r	pass/fail
1	i	0	fail

My first test is now failing! This is progress. But, why is i2r returning 0? I look at the code and realise that this must be the VBA default.

- I switch to VBA, take a peek in the help files to learn how VBA returns parameters and then change the function so that it returns the string "i".

```
Public Function i2r(i As Integer) As String
i2r = "i"
End Function
```

- Then I pop back to excel. Do a manual "F9" recalc and hey-presto! my first test has passed.

Integer	Roman	i2r	Pass/fail
1	i	i	pass

1	i	i	Pass
---	---	---	------

Phewww! Since I'm journaling my steps this has taken a while, but in normal circumstances I reckon it would take 1-2 minutes, tops.

Passing my 2nd test

I figure now is as good a time as any to try my second test.

- I add in the test 2 = "ii" and copy the C and D formulas from the existing row.

Integer	Roman	i2r	Pass/fail
1	i	i	Pass
2	ii	i	Fail

So, now I have my 2nd test and it is failing. This is good. It's the rhythm of TDD - write a failing test, make it work using the simplest way possibly, then refactor (i.e. tidy up) the code.

I reckon I know how to fix up this code.

- After a quick peek in the help files on the "if" statement, I change the code

```
Public Function i2r(i As Integer) As String
    If i = 1 Then
        i2r = "i"
    Else
        i2r = "ii"
    End If
End Function
```

- Then I switch across to Excel and press F9 to recalculate.

Integer	Roman	i2r	Pass/fail
1	i	i	Pass
2	ii	ii	Pass

Two passing tests! I'm on a roll.

A few more tests, but no Refactoring.

- I take a look at the code and wonder if I can tidy it up at all (refactor) ... and I can't see anything.
- So I add in the next test: 3 = "iii", it fails, I change the code so it passes, but I still can't see anything to refactor.

- I repeat this process for tests 4, 5, 6, 7, 8, adding each test - one at a time - making it work, trying to refactor.
- I start to think that this could be a long and drawn out process. I can't see anything that I can easily refactor.

```
Public Function i2r(i As Integer) As String

If i = 1 Then
    i2r = "i"
ElseIf i = 2 Then
    i2r = "ii"
ElseIf i = 3 Then
    i2r = "iii"
ElseIf i = 4 Then
    i2r = "iv"
ElseIf i = 5 Then
    i2r = "v"
ElseIf i = 6 Then
    i2r = "vi"
ElseIf i = 7 Then
    i2r = "vii"
ElseIf i = 8 Then
    i2r = "viii"
End If

End Function
```

- But then I look a little harder.

Some Refactoring

I see a pattern with the 1, 2 and 3 and 6, 7, 8. I wonder if I can do something with the repeating i's. They seem to repeat.

But what to do?

When I consider the code, I see that there are two ways in which I think I can tidy up the code (refactor). First, I could turn 1, 2 and 3 into a loop. Second, I could turn 6, 7 and 8 into (5 + 1, 2 or 3). I *think* I'm clever enough to do both at the same time but then I recall the XP idea of doing things one at a time, since it's more likely to work/ less likely to break. I flip an imaginary coin and decide to turn 1, 2 and 3 into a loop.

- First, I'm not sure quite why, but I move the code for 1, 2 and 3 down to the bottom of the code.

```
Public Function i2r(i As Integer) As String

If i = 4 Then
    i2r = "iv"
ElseIf i = 5 Then
    i2r = "v"
ElseIf i = 6 Then
    i2r = "vi"
ElseIf i = 7 Then
    i2r = "vii"
ElseIf i = 8 Then
    i2r = "viii"
End If

If i = 1 Then
    i2r = "i"
ElseIf i = 2 Then
    i2r = "ii"
ElseIf i = 3 Then
    i2r = "iii"
End If

End Function
```

```

If i = 1 Then
    i2r = "i"
ElseIf i = 2 Then
    i2r = "ii"
ElseIf i = 3 Then
    i2r = "iii"
End If

End Function

```

- Just to make sure this hasn't broken anything I switch back to the spreadsheet and recalc. I probably didn't need to do this but it has been a long time since I've coded and I'm a bit nervous doing this. All of the tests still work.

Integer	Roman	i2r	pass/fail
1	i	i	pass
2	ii	ii	pass
3	iii	iii	pass
4	iv	iv	pass
5	v	v	pass
6	vi	vi	pass
7	vii	vii	pass
8	viii	viii	pass

- Then I turn the 1,2 and 3 bits into a loop.

This is my first attempt:

```

Public Function i2r(i As Integer) As String

If i = 4 Then
    i2r = "iv"
ElseIf i = 5 Then
    i2r = "v"
ElseIf i = 6 Then
    i2r = "vi"
ElseIf i = 7 Then
    i2r = "vii"
ElseIf i = 8 Then
    i2r = "viii"
End If

i2r = ""
While i < 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
End

End Function

```

I recalc the spreadsheet and it works!

But it doesn't. I realise this when I look at the code and realise I put a END, rather than a WEND, to close off the While loop. The code wasn't actually executed because Excel should have given me an error, but it didn't. The problem is that Excel doesn't recalc unless one of its inputs has changed. It doesn't realise that the VBA function has been modified. One way around this is to edit each cell (F2 and Enter) and force it to manually recalc every

cell, but I've since found that Adding the following statement to the top of the function gets around it.

```
Public Function i2r(i As Integer) As String
Application.Volatile
```

...

So, I pop back to the spreadsheet and press F9 to rerun the tests again.

Ooops. I've broken lots of tests.

Integer	Roman	i2r	pass/fail
1	i	i	pass
2	ii	ii	pass
3	iii		fail
4	iv		fail
5	v		fail
6	vi		fail
7	vii		fail
8	viii		fail

When I look back at the code I realise that I've done two things wrong.

- First, the loop should be ≤ 3 , not < 3 . I change that, recalc the tests and 3 is fixed.

But what about 4 - 8? They're still broken. And, no wonder, the `i2r = ""` before the loop destroy's them.

- I fix the code for the current tests by moving `i2r = ""` to the top of the code.

When I recalc the spreadsheet ... all of the tests work! The tests are very comforting ... a safety net.

```
Public Function i2r(i As Integer) As String
Application.Volatile
```

```
i2r = ""
```

```
If i = 4 Then
    i2r = "iv"
ElseIf i = 5 Then
    i2r = "v"
ElseIf i = 6 Then
    i2r = "vi"
ElseIf i = 7 Then
    i2r = "vii"
ElseIf i = 8 Then
    i2r = "viii"
End If
```

```
While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend
```

```
End Function
```

So. The 2nd Refactoring. Hmm. I want to change the code so that 6, 7, and 8 are 5 + 1, 2, or 3.

I have a wee discussion with myself before making any changes and decide to comment out the 6, 7 and 8 elseif's, change the if i = 5 to if i >= 5, and subtract 5 from the i2r, and see how it works. I make these changes and break a lot of the tests. Hmm. What to do?

- I turn on the debugger. And the moment I step through the very simple code I see my blindingly obvious mistake. Next time I'll look a little harder (but don't forget I'm learning VBA as I do this).
- I fix my mistake. I turn off the debugger.
- I switch to Excel and recalc. All of the tests pass.
- I remove the comments I'd added.

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 4 Then
    i2r = "iv"
ElseIf i >= 5 Then
    i2r = "v"
    i = i - 5
End If

While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend

End Function
```

I look at the code and I'm quite happy with it. I'm not thrilled with it and I wonder if I could refactor it a bit more, but I think I'd be better adding a few test cases to understand the problem better. In other words, I want to do a bit more analysis and I am going to do it with the code.

- I add the test 9 = ix. If fails. I make it work by adding another ifelse, just like the 4. Can't see any way to refactor it.
- I add the test 10 = x. If fails. Again I add it as another ifelse, just like the 4 and 9.

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 4 Then
    i2r = "iv"
ElseIf i = 9 Then
    i2r = "ix"
ElseIf i = 10 Then
    i2r = "x"
ElseIf i >= 5 Then
    i2r = "v"
    i = i - 5
End If
```

```
While i <= 3 And i > 0
  i2r = i2r + "i"
  i = i - 1
Wend
End Function
```

- As I add the test 11 = xi, I notice the i following the x and I think about the 12 and 13. This looks like it's the good old repeating "i" situation.

I'm tempted to add the 12 and 13 test cases in too, but I decided not to. Why? After all, it seems easy enough to do, but then I'd move into trying to make 3 tests work at once and although they'd probably work I suspect that it's not the intention of TDD. And besides, it is easy enough to add the 12 and 13 in afterwards.

- I make the 11 test work very easily

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 4 Then
  i2r = "iv"
ElseIf i = 9 Then
  i2r = "ix"
ElseIf i >= 10 Then
  i2r = "x"
  i = i - 10
ElseIf i >= 5 Then
  i2r = "v"
  i = i - 5
End If

While i <= 3 And i > 0
  i2r = i2r + "i"
  i = i - 1
Wend

End Function
```

- I look to see if I can refactor anything. I can't see anything.
- I add in the 12 = xii test case. It works straight away. As does the 13 = xiii.
- I add in the 14 = xiv test case. Now the easiest way to make this test work is to put in an if x = 14 and I do this. But when I go to refactor I realise that it's really just 10 + 4, so I move the if x=4 out of the string of if's and move it after the 5 and between the 3, 2 and 1 code. It should just trickle through.

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 9 Then
  i2r = "ix"
ElseIf i >= 10 Then
  i2r = "x"
  i = i - 10
ElseIf i >= 5 Then
```

```

    i2r = "v"
    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
End If

While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend

End Function

```

- I look for ways to refactoring but I (with my limited coding ability) can't see any that I can do with confidence.
- I add the 15 = xv test case. I'm starting to see a pattern here and I change the code easily.

```

Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 9 Then
    i2r = "ix"
ElseIf i >= 10 Then
    i2r = "x"
    i = i - 10
End If

If i >= 5 Then
    i2r = "v"
    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
End If

While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend

End Function

```

But 15 doesn't work and the test for 9 breaks.

Integer	Roman	i2r	pass/fail
1	i	I	pass
2	ii	li	pass
3	iii	lii	pass
4	iv	lv	pass
5	v	V	pass
6	vi	Vi	pass
7	vii	Vii	pass
8	viii	Viii	pass
9	ix	Viv	fail
10	x	X	pass
11	xi	Xi	pass

12	xii	Xii	pass
13	xiii	Xiii	pass
14	xiv	Xiv	pass
15	xv	V	fail

I keep wanting to remind you, dear reader, that I haven't coded for years and VBA is new to me, just so you don't think I'm incompetent. But, I suspect if you've been following along in Excel then you've discovered you too make lots of little mistakes as you go. At least I hope so.

- When I see that the 9 test case is also broken it is immediately obvious what I've done wrong.
- I fix it quickly. The tests work.

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i = 9 Then
    i2r = "ix"
    i = i - 9
End If

If i >= 10 Then
    i2r = i2r + "x"
    i = i - 10
End If

If i >= 5 Then
    i2r = i2r + "v"
    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
End If

While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend

End Function
```

- Then I do a bit of tidying up - moving things around a little.
- For each small change I make, I recalc the tests.

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i >= 10 Then
    i2r = i2r + "x"
    i = i - 10
End If

If i = 9 Then
    i2r = i2r + "ix"
    i = i - 9
End If

If i >= 5 Then
    i2r = i2r + "v"
End If
```

```

    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
End If

While i <= 3 And i > 0
    i2r = i2r + "i"
    i = i - 1
Wend

End Function

```

- Then I do a bit more tidying up.

```

Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i >= 10 Then
    i2r = i2r + "x"
    i = i - 10
End If

If i = 9 Then
    i2r = i2r + "ix"
    i = i - 9
End If

If i >= 5 Then
    i2r = i2r + "v"
    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
    i = i - 4
End If

While i >= 1
    i2r = i2r + "i"
    i = i - 1
Wend

End Function

```

- Then I look at the code and I see the obvious pattern. It jumps out at me now. I can create new function for each of the numbers.
- I try to add the new function - but out of laziness and lack of imagination I give up and decide to add more test cases first.
- I add test case for 16 =xvi and then I get enthusiastic and add the tests into the spreadsheets for 17 through to 19. I think they should work straight away and they do.
- I add 20 = xx and it doesn't work immediately.
- I look at the code and quickly change the IF x >= 10 to a while loop.

Hmmmm, I think. I reckon that these tests should now work all the way to 50.

- So I add in the integer values from 21 - 30 and instead of filling in the roman numerals, I cheat a little and copy down the i2r() function and

eyeball them to check that they're right. They all are so I copy the values from my function into the test data.

- Then I chose a few other numbers up to 50, which I figure will fail with the current code. But, when I type in 46 I realise that the 40's are a special case like 4 and 9, and later on 90, 400 and 900. So I check in google¹ what the symbol for 40 is ... it's XL. I add in the test case for 40 and make it work.
- I add in a test case for 49 and it works. I'm confident that the rest of the 40's will work too.
- I add in a test case for 50 = l and as expected it doesn't work. By now, the code is trivial so I make it work.
- Then I think about some Refactoring.

The code looks like this:

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

If i >= 50 Then
    i2r = i2r + "l"
    i = i - 50
End If

If i >= 40 Then
    i2r = i2r + "xl"
    i = i - 40
End If

While i >= 10
    i2r = i2r + "x"
    i = i - 10
Wend

If i = 9 Then
    i2r = i2r + "ix"
    i = i - 9
End If

If i >= 5 Then
    i2r = i2r + "v"
    i = i - 5
End If

If i = 4 Then
    i2r = i2r + "iv"
    i = i - 4
End If

While i >= 1
    i2r = i2r + "i"
    i = i - 1
Wend

End Function
```

I figure that I want to create a function where I feed in each *roman numeral* and if the current value of *i* > *the decimal equivalent* of the *roman numeral* then concatenate the *roman numeral* on the end of *i2r* and subtract the *decimal equivalent* from *i*.

¹ <http://www.google.co.uk/search?q=40+i+roman+numerals>

The only thing that is stopping me is the mixture of whiles and if's in the pattern above. The i's and x's are whiles, but the rest are if's. Hmmm, I think. After a bit of contemplation I'm pretty sure that each of the if's can simply be replaced with a while.

But, how can I be sure? The easiest way to be sure is to try it and see. I have the tests to save me if I make a mistake and I have undo to roll back if I need to. But, to be on the safe side I decide to try this from the bottom up. That is, I start with the i's.

This is what I come up with, for the i's (I couldn't think of a good function name, so m it is):

```
...
i2r = i2r + m(i, "i", 1)
'While i >= 1
' i2r = i2r + "i"
' i = i - 1
'Wend

End Function

Public Function m(ByRef i As Integer, ByVal RomanCharacter As String, ByVal
IntegerValue As Integer) As String
While i >= IntegerValue
    m = m + RomanCharacter
    i = i - IntegerValue
Wend
End Function
```

And it works.

- So I try it on the 4 IF, replacing the if i = 4 with i2r = i2r + m(i, "iv", 4)
- And it works.
- So I try it on the 5. It works to.
- So, I do all of the individual roman numerals. Recalculating the tests after each small change.

I end up with this:

```
Public Function i2r(i As Integer) As String
Application.Volatile

i2r = ""

i2r = i2r + m(i, "l", 50)
i2r = i2r + m(i, "xl", 40)
i2r = i2r + m(i, "x", 10)
i2r = i2r + m(i, "ix", 9)
i2r = i2r + m(i, "v", 5)
i2r = i2r + m(i, "iv", 4)
i2r = i2r + m(i, "i", 1)

End Function

Public Function m(ByRef i As Integer, ByVal RomanCharacter As String, ByVal
IntegerValue As Integer) As String
While i >= IntegerValue
    m = m + RomanCharacter
    i = i - IntegerValue
```

```
Wend  
End Function
```

And the tests all work afterwards.

- Then I refactor the i2r() function.

```
Public Function i2r(i As Integer) As String  
Application.Volatile  
  
i2r = m(i, "l", 50) + m(i, "x1", 40) + m(i, "x", 10) + m(i, "ix", 9) + m(i, "v", 5) +  
m(i, "iv", 4) + m(i, "i", 1)  
  
End Function
```

I'm confident that I understand how this function is supposed to work so I quickly add in the 90, 100, 400, 500, 900 and 1000 code, their test cases, and a smattering of tests in between.

But, I'm not going to show you that 😊

Clarke Ching
Clarke.Ching@gmail.com
www.clarkeching.com