TDD: Test Driven Development with Visual Studio 2010

Stefano Paluello
stefano.paluello@pastesoft.com
http://stefanopaluello.wordpress.com
Twitter: @palutz

Test Driven Development

- **Test-driven development** (**TDD**) is a software development process that relies on the repetition of a very short development cycle:
  - the developer writes a failing automated test case that defines a desired improvement or new function (RED)
  - produces code to pass that test (GREEN)
  - finally refactors the new code to acceptable standards (REFACTOR)

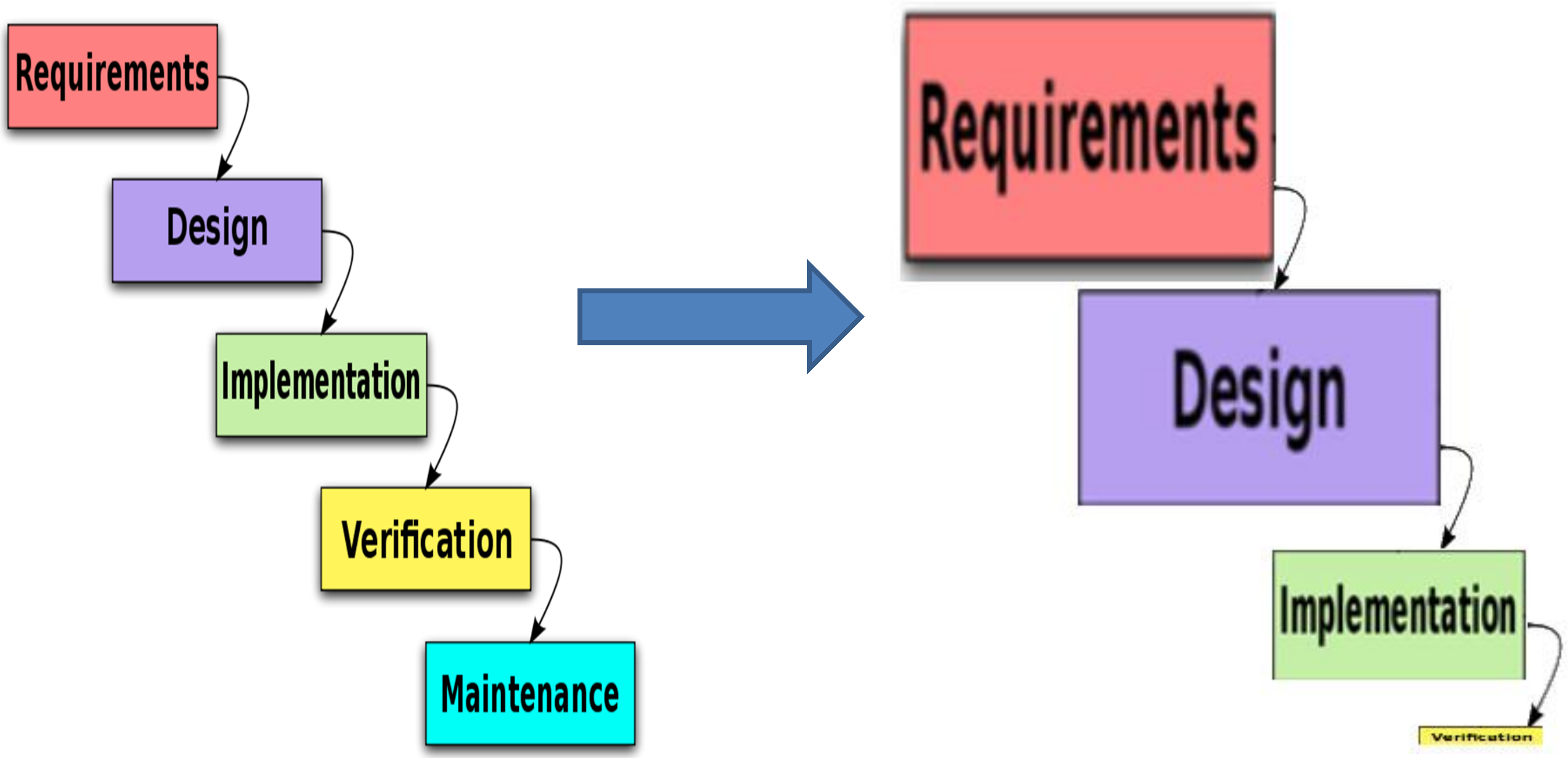
[Wikipedia]

RED GREEN REFACTOR
TEST-DRIVEN DEVELOPMENT

Test Driven Development

- Test Driven Development is one of the Extreme Programming principles (actually is the first one), but can be used also by itself
- TDD is a CHANGE OF APPROACH for developers, is a new MINDSET, from writing code and then testing it (if there is enough time) to writing the test first and then writing the code to satisfy the test

Why do I need TDD?



Before TDD…

- The industry “standard” behavior was to write the code, and then write the tests. This leads to lots of applications without tests at all
- Also the “good” applications with some tests written, weren’t (aren’t!!!) able to know whether the new code has broken the existing features

Typical result?

The new approach…

- Why do we have to stick on that plan?
- ***Kent Beck*** introduced a new method, inside his book about *Extreme Programming (XP)*, in which he proposed to reverse this order by starting with unit tests and then writing the implementation.

Why TDD is good?

- Thinking about the tests pushes the developer to understand better the gathered requirements
- With the tests defined, the developer can focus on writing ONLY the code to satisfy them (reduce the over-engineered code, or *dead-code*)
- The unit tests help to check that any new modification won’t break the existing features

TDD is good (2)

- TDD speed-up your code! Ok, writing a whole set of code to test your program could appear as too much work, but with test you can trust your code more and you can have a quick feedback about your design and how your objects behave. When all tests have succeeded, you don’t need to spend so much time debugging <u>the tested code</u>.

TDD is not

- Test Driven Development is good, not God! ☺
- Test Driven Development isn’t a magic stick that will solve ALL your problems
- Test Driven Development is not working if there is not a REALLY mind shift

TDD != Unit Test

- Ok, this could be a bit confusing but…
- Test Driven Development is a process, is a mindset, a different approach, focusing on isolated test case <u>to drive design</u>
- Unit Test is a procedure, a part of the test process

How to justify TDD to PM

- Microsoft just published a research report where was pointed out that using TDD, the bugs and defects are reduced from 40% to 90% (nice code was made before! ☺)

http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf

How to justify TDD to Dev Team Lead

- TDD increase the software's flexibility
- To be testable, a codebase using TDD is more decoupled and for this reason also adding more features is easier

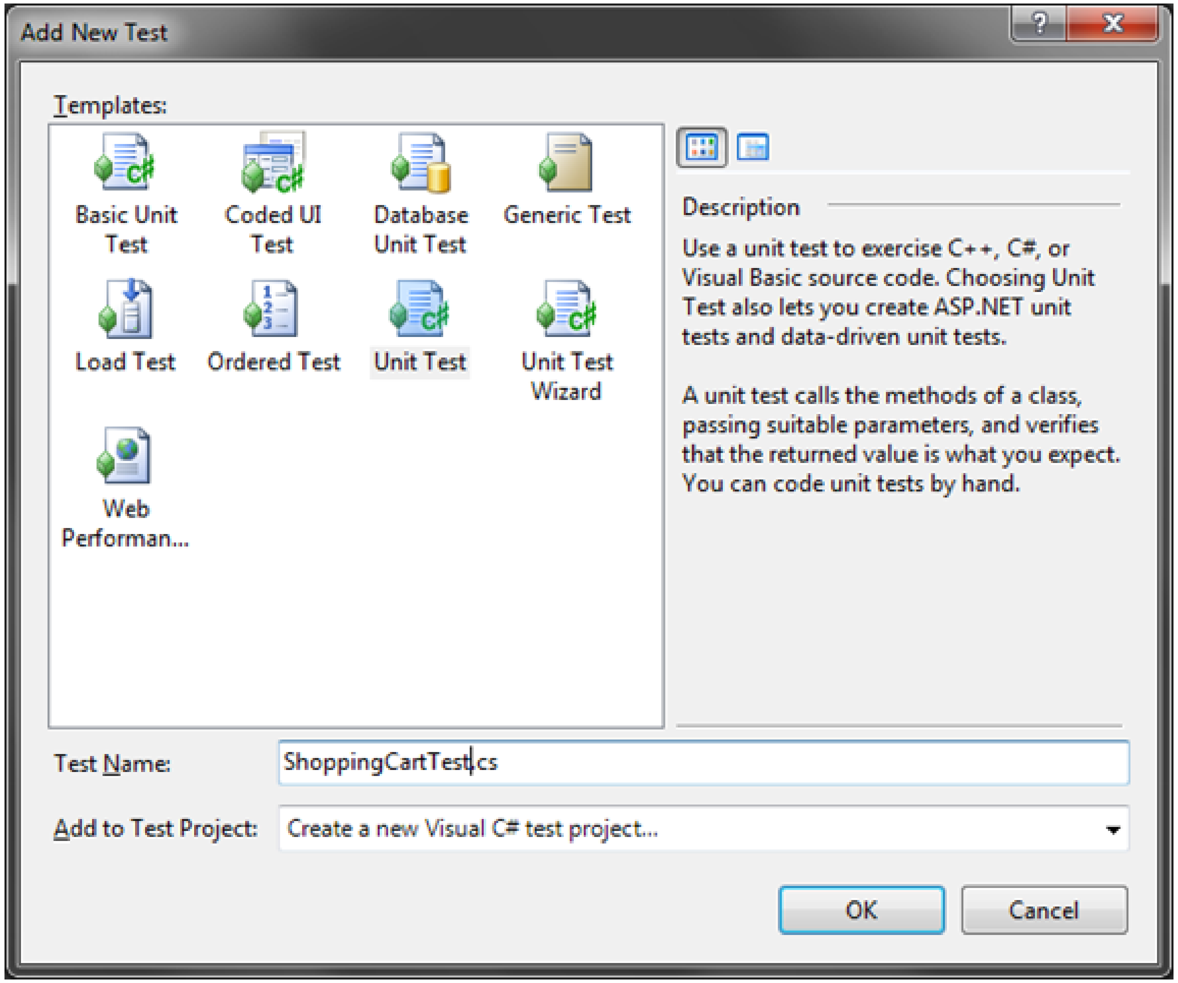
How to justify TDD to…

- Stop thinking and justify!
- Just CODE!

Ops, some advices before code…

- Write simple test that verify only ONE behavior
- Avoid dependencies between tests
- One test class for every class within the production code
- Use test initialization code before and test cleanup code after running your test (sealed context)
- Verify that all the tests are passed before move to another feature test
- Maximize the number of automated tests

TDD with Visual Studio



Description

Use a unit test to exercise C++, C#, or Visual Basic source code. Choosing Unit Test also lets you create ASP.NET unit tests and data-driven unit tests.

A unit test calls the methods of a class, passing suitable parameters, and verifies that the returned value is what you expect. You can code unit tests by hand.

Test Name: ShoppingCartTest.cs

Add to Test Project: Create a new Visual C# test project...

BEPS • BEST EMBEDDED PRODUCTS & SOLUTIONS

TTG Torino Technologies Group

Unit test with Visual Studio 2010

DEMO

BEST EMBEDDED
PRODUCTS & SOLUTIONS

Demo Recap

- Despite the Intellisense, Visual Studio 2010 has some tools that help you to focus on TDD
- Using Code Coverage, Visual Studio will show you the effectiveness of your tests

Web test with Visual Studio 2010

DEMO

BEST EMBEDDED
PRODUCTS & SOLUTIONS

Torino Technologies Group

Some test "tools" for VS2010

- ***Code Contracts***
  - allow you to explicitly declare the preconditions, postconditions and invariant part of your code
  - this improves also the testing via runtime checking (injecting the contracts)
  - it comes from Devlabs and it was an external assemblies for .Net 3.5, now included in .Net 4.0

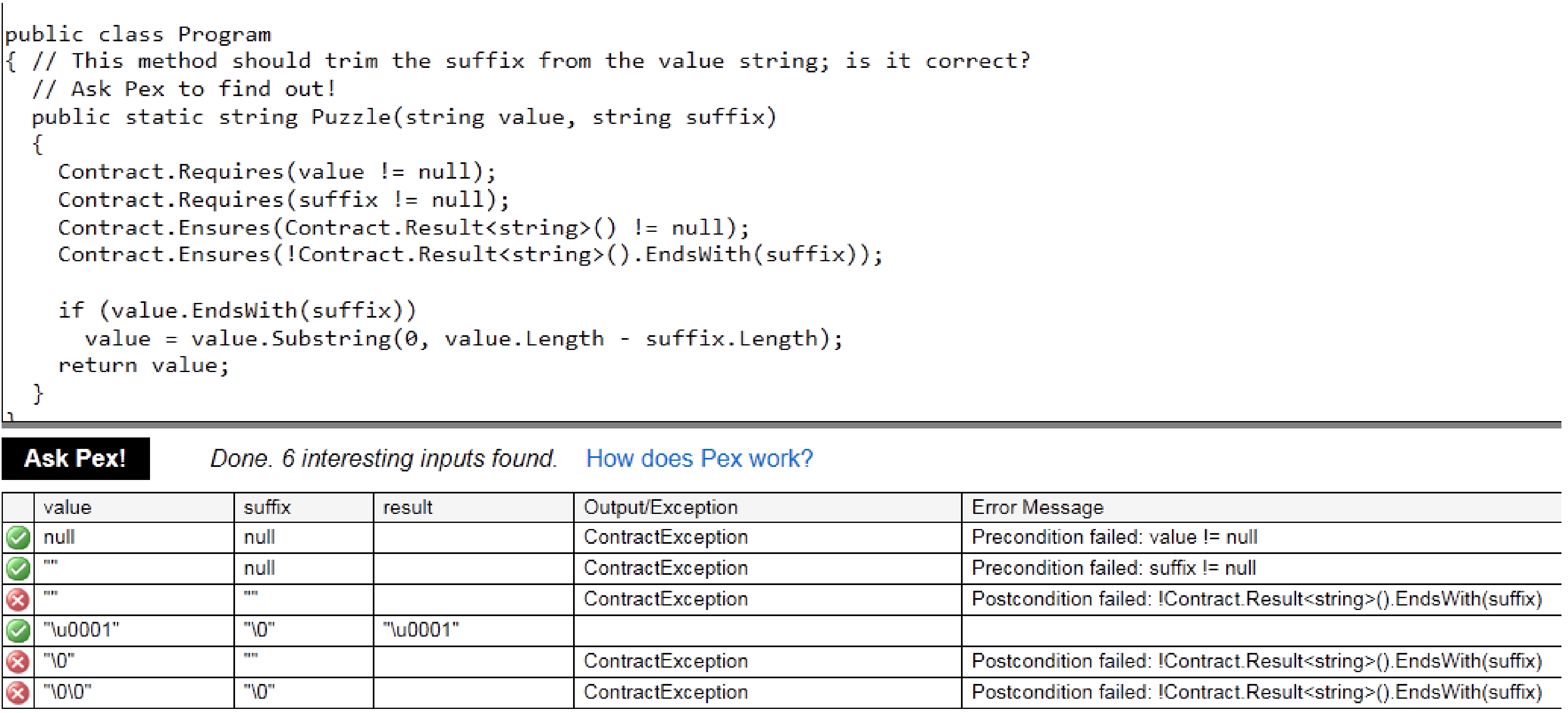
BEPS
BEST EMBEDDED PRODUCTS & SOLUTIONS

Torino Technologies Group

VS2010 tools (2)

- ***Pex and Moles***
  - Inside the *Visual Studio 2010 Power tools* extension
  - **Pex** automatically generates test suites with high code coverage**.** Right from the Visual Studio code editor, Pex finds interesting input-output values of your methods (white box testing), which you can save as a small test suite with high code coverage. Microsoft Pex is a Visual Studio add-in for testing .NET Framework applications.
  - **Moles** allows to replace any .NET method with a delegate**.** Moles supports unit testing by providing isolation by way of detours and stubs. The Moles framework is provided with Pex, or can be installed by itself as a Microsoft Visual Studio add-in (example test Asp.Net and Sharepoint applications!)

Contracts and Pex together



```
public class Program
{ // This method should trim the suffix from the value string; is it correct?
  // Ask Pex to find out!
  public static string Puzzle(string value, string suffix)
  {
    Contract.Requires(value != null);
    Contract.Requires(suffix != null);
    Contract.Ensures(Contract.Result<string>() != null);
    Contract.Ensures(!Contract.Result<string>().EndsWith(suffix));

    if (value.EndsWith(suffix))
      value = value.Substring(0, value.Length - suffix.Length);
    return value;
  }
}
```

Done. 6 interesting inputs found. How does Pex work?

	value	suffix	result	Output/Exception	Error Message
✓	null	null		ContractException	Precondition failed: value != null
✓	""	null		ContractException	Precondition failed: suffix != null
✗	""	""		ContractException	Postcondition failed: !Contract.Result<string>().EndsWith(suffix)
✓	"\u0001"	"\0"	"\u0001"		
✗	"\0"	""		ContractException	Postcondition failed: !Contract.Result<string>().EndsWith(suffix)
✗	"\0\0"	"\0"		ContractException	Postcondition failed: !Contract.Result<string>().EndsWith(suffix)

BEPS • BEST EMBEDDED PRODUCTS & SOLUTIONS

Torino Technologies Group

BEPS
BEST EMBEDDED PRODUCTS & SOLUTIONS

Excellence Awards 2008

Windows® Embedded Silver Partner

BEPS
BEST EMBEDDED PRODUCTS & SOLUTIONS

&

TTG Torino Technologies Group

Programmazione per Device: da Embedded a Desktop

Domande & Risposte

BEPS
BEST EMBEDDED PRODUCTS & SOLUTIONS

TTG Torino Technologies Group