Faculty of Mathematics and Physics
Charles University in Prague
4th May 2015

C# Made Easy!

Programming II

Workshop 11 – Functional Testing

Workshop 11
Outline

1. Test
2. Functional Testing – What, Why, How
3. Homework

Test 11

Test

Find the test here (no-ads):

http://goo.gl/bgQp7p

Permanent link:

https://docs.google.com/forms/d/1Cqmf-U46lo8KlYpAmF3pBLF4b8xuIeesjLbiBn98IwI/viewform

Time for the test:

8 min

Testing
What?

WHAT?

Testing
What?

- Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

To put it simply…

- Running a code that executes another code and compares results with pre-computed/pre-specified ones.

Testing
What?

- Simple example

```
class Calc {
     int Add(int a1, int a2)
}

class CalculatorTest {

     public boolean TestAdd() {
          Calc c = new Calculator();
          if (c.Add(1, 1) == 2) return true;
          return false;
     }

}
```

Testing
Why?

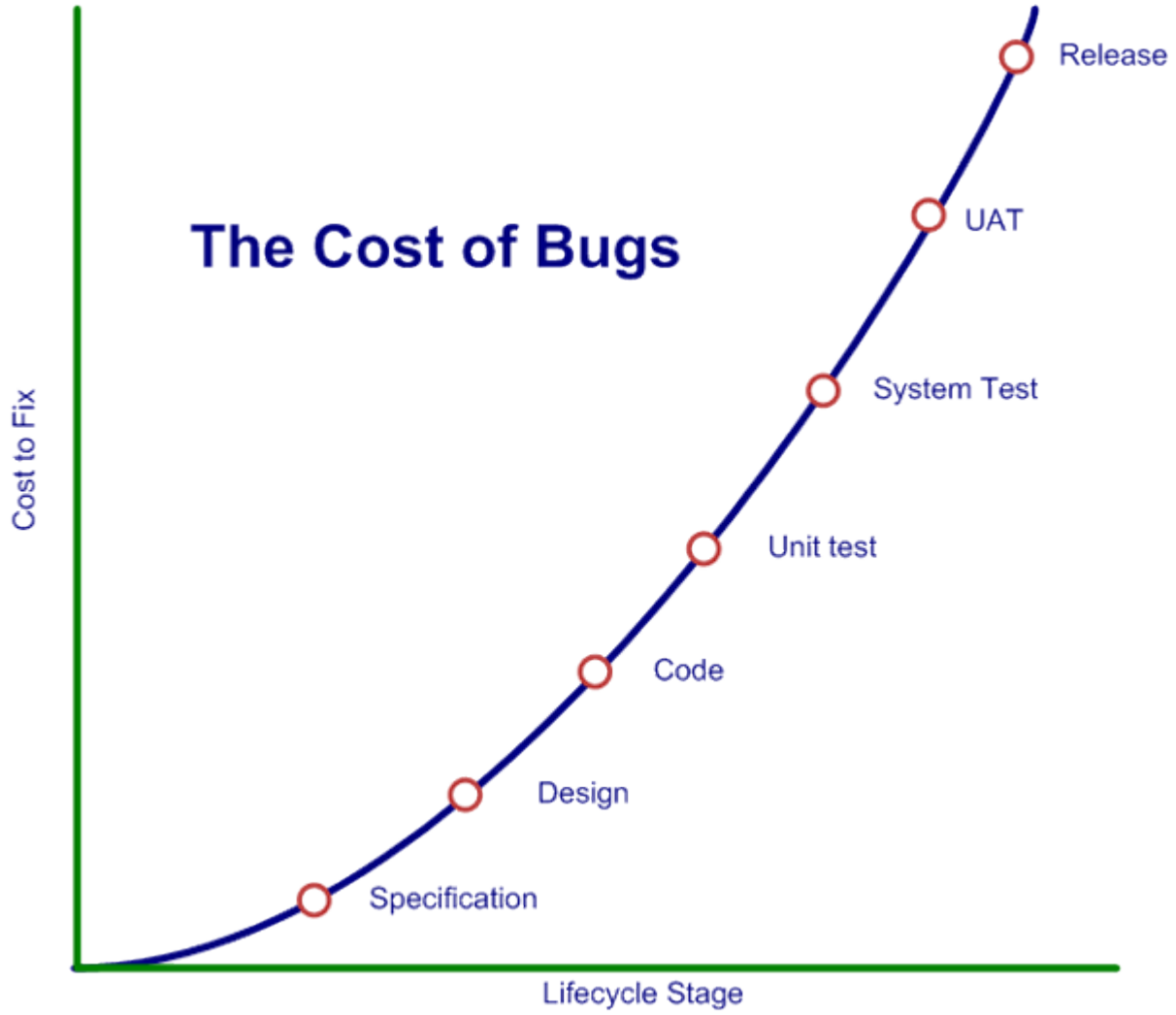
WHY?

Testing
Why?

Because **TIME** (translates as **MONEY**) matters!

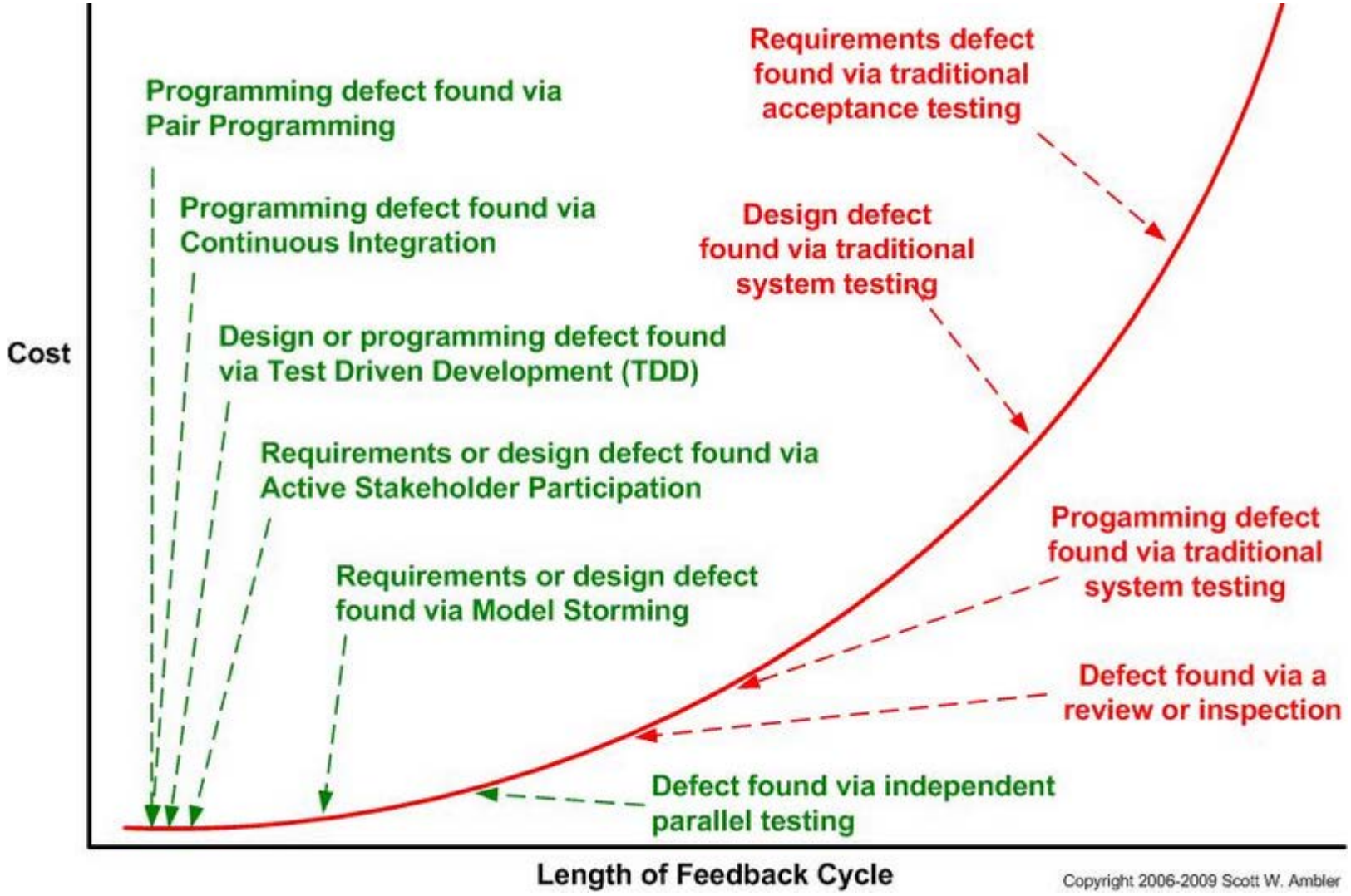
Testing

Why?


The Cost of Bugs
Cost to Fix
Lifecycle Stage
Specification
Design
Code
Unit test
System Test
UAT
Release

Testing

Why?


Requirements defect found via traditional acceptance testing
Programming defect found via Pair Programming
Programming defect found via Continuous Integration
Design defect found via traditional system testing
Design or programming defect found via Test Driven Development (TDD)
Requirements or design defect found via Active Stakeholder Participation
Progamming defect found via traditional system testing
Requirements or design defect found via Model Storming
Defect found via a review or inspection
Defect found via independent parallel testing
Cost
Length of Feedback Cycle
Copyright 2006-2009 Scott W. Ambler

Testing
Why?

- Tests reduce bugs in existing features
- Tests reduce bugs in new features
- Tests (of complex code base) are good documentation
- Tests improve design
- Tests reduce cost of change
- Tests constrain features
- Tests reduce fear of making changes
  - For your colleagues as well as you!

Testing
Why not to?

WHY NOT TO?

Testing
Why not to write tests?

- It takes too much time to write tests
- It takes too much time to execute tests
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

Testing
Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

Testing
Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually … and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
- I don't really know how the code should behave so I can't test it!

Testing
Why not to write tests?

- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually … and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
  - Oh, and you expect to have customers?
- I don't really know how the code should behave so I can't test it!

Testing
Why not to write tests?

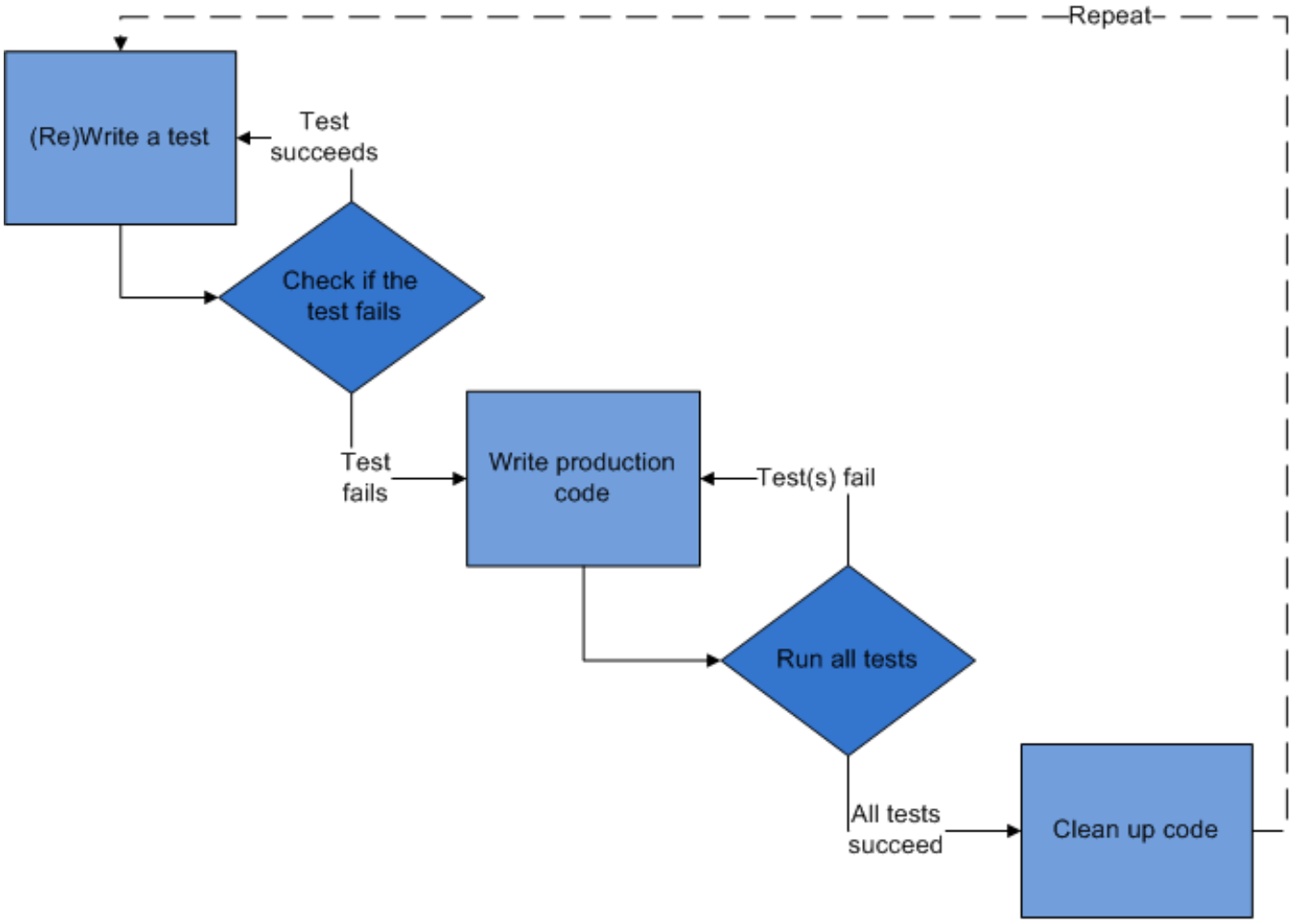
- It takes too much time to write tests
  - But you have time to hunt bugs down?
- It takes too much time to execute tests
  - You are running your tests manually … and the same time you're considering yourself to be THE programmer?
- It's not your job to test the code
  - Oh, and you expect to have customers?
- I don't really know how the code should behave so I can't test it!
  - You should have not started writing such code from the very beginning!

Testing
How?

HOW?

Testing

How?

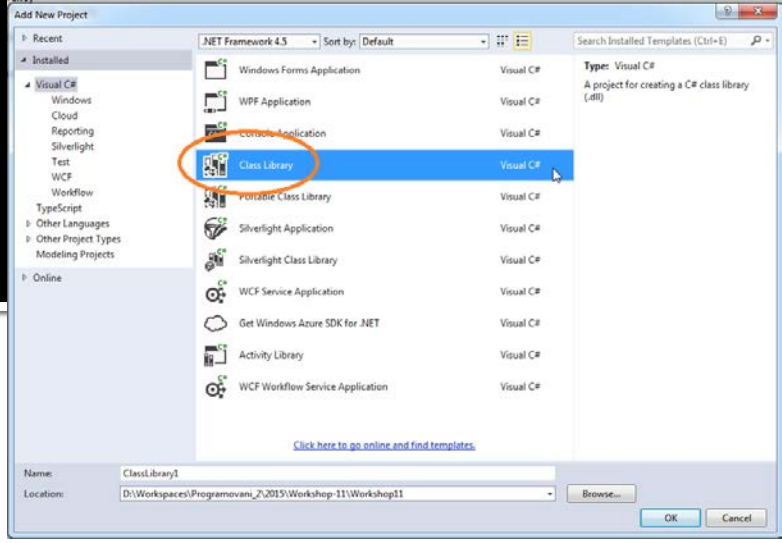

Repeat
(Re)Write a test
Test succeeds
Check if the test fails
Test fails
Write production code
Test(s) fail
Run all tests
All tests succeed
Clean up code

Testing

How?

Show time...

Testing
Creating projects



Create two (Class Library) projects

1. Fist will contain your classes (unit) for testing

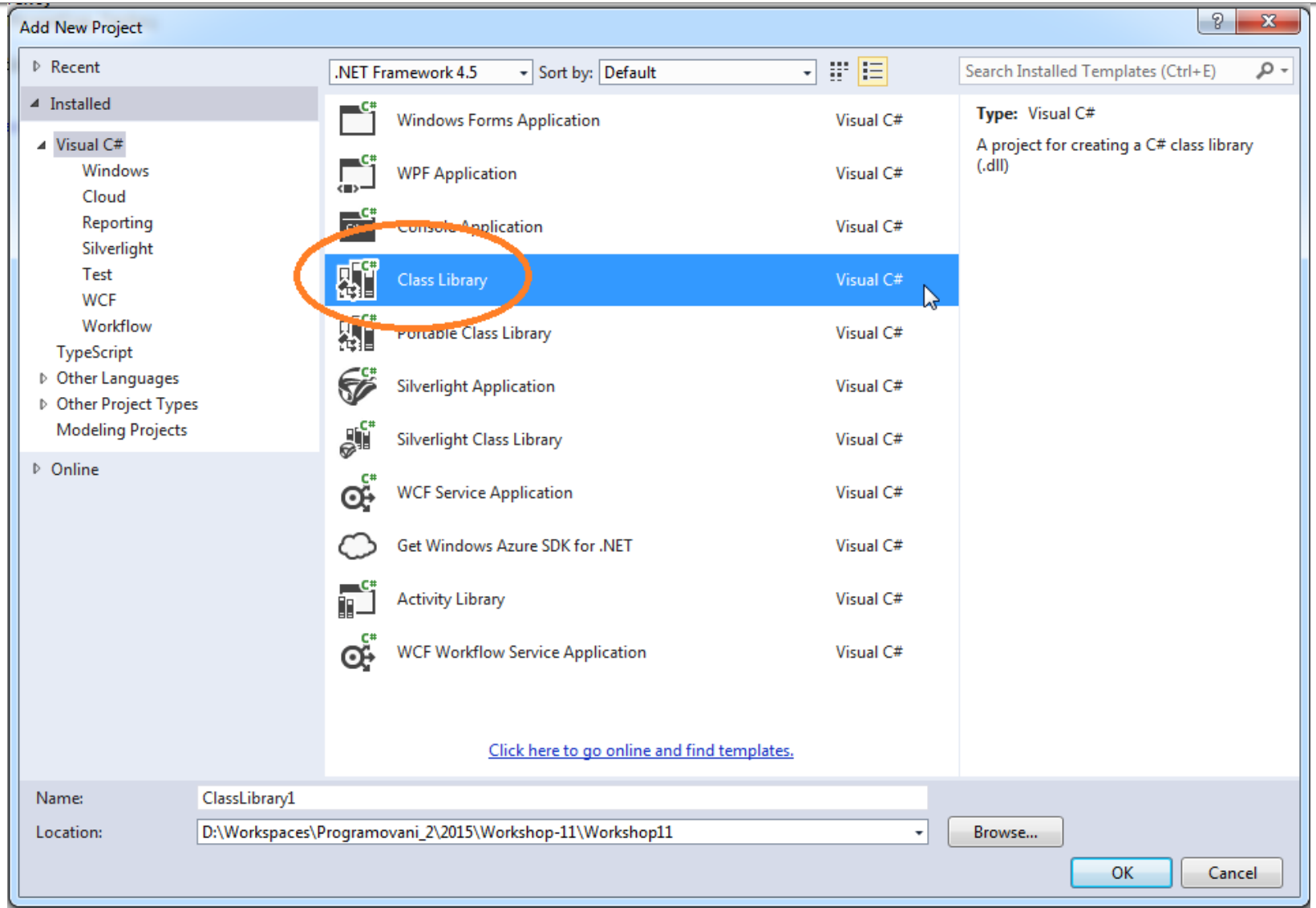
   Name it e.g.: MyLibrary

2. Second will contain TESTs that will be executed to test your first project

   Name it e.g.: MyLibrary.Test

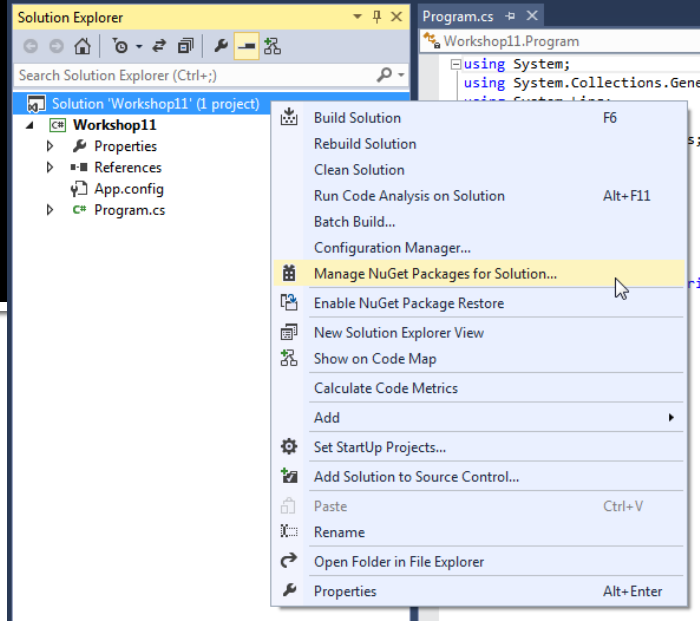
Testing

Creating projects



Testing

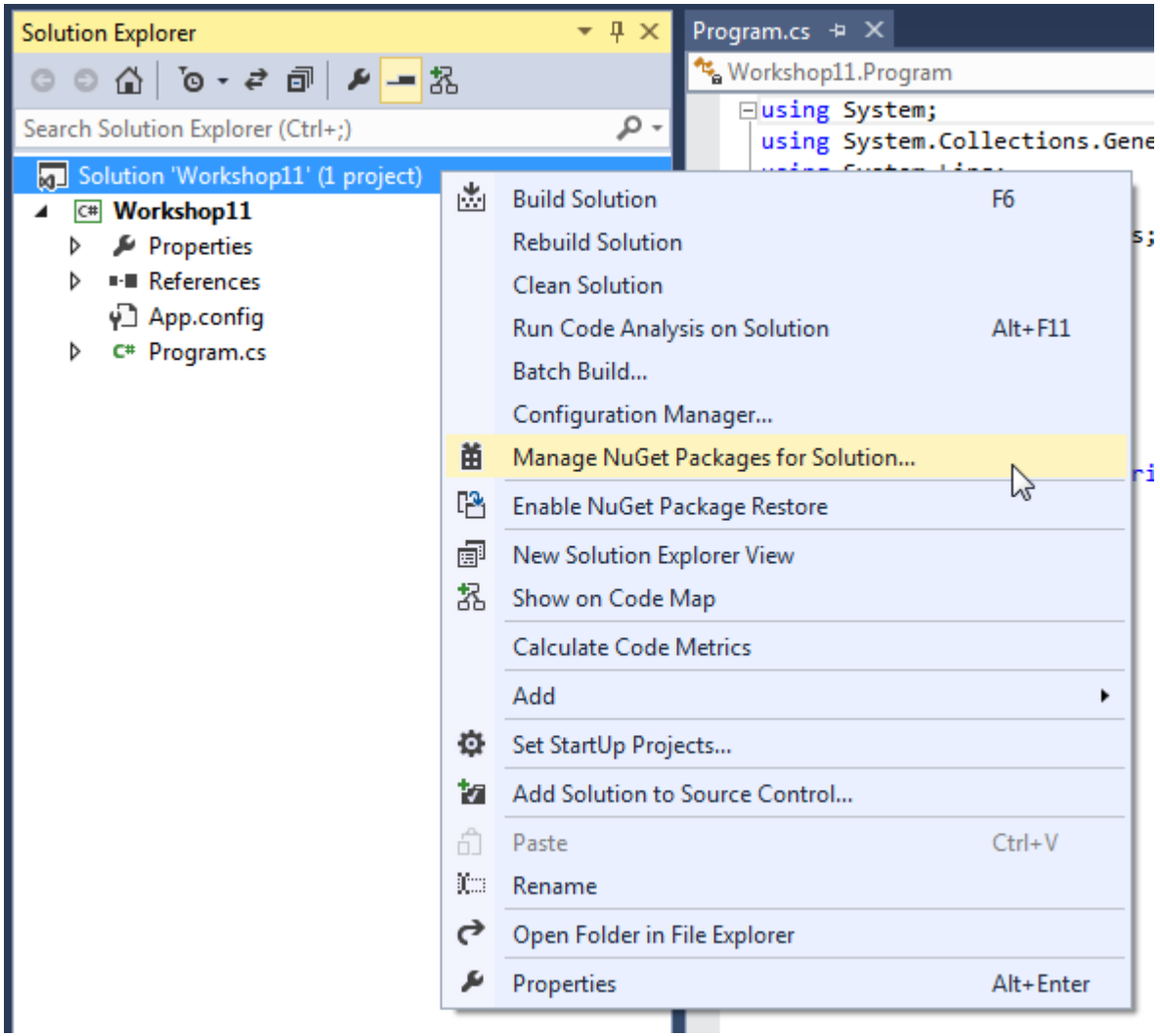
Creating projects



Import required libraries / packages that you will need to run tests, perform code coverage and generate reports.

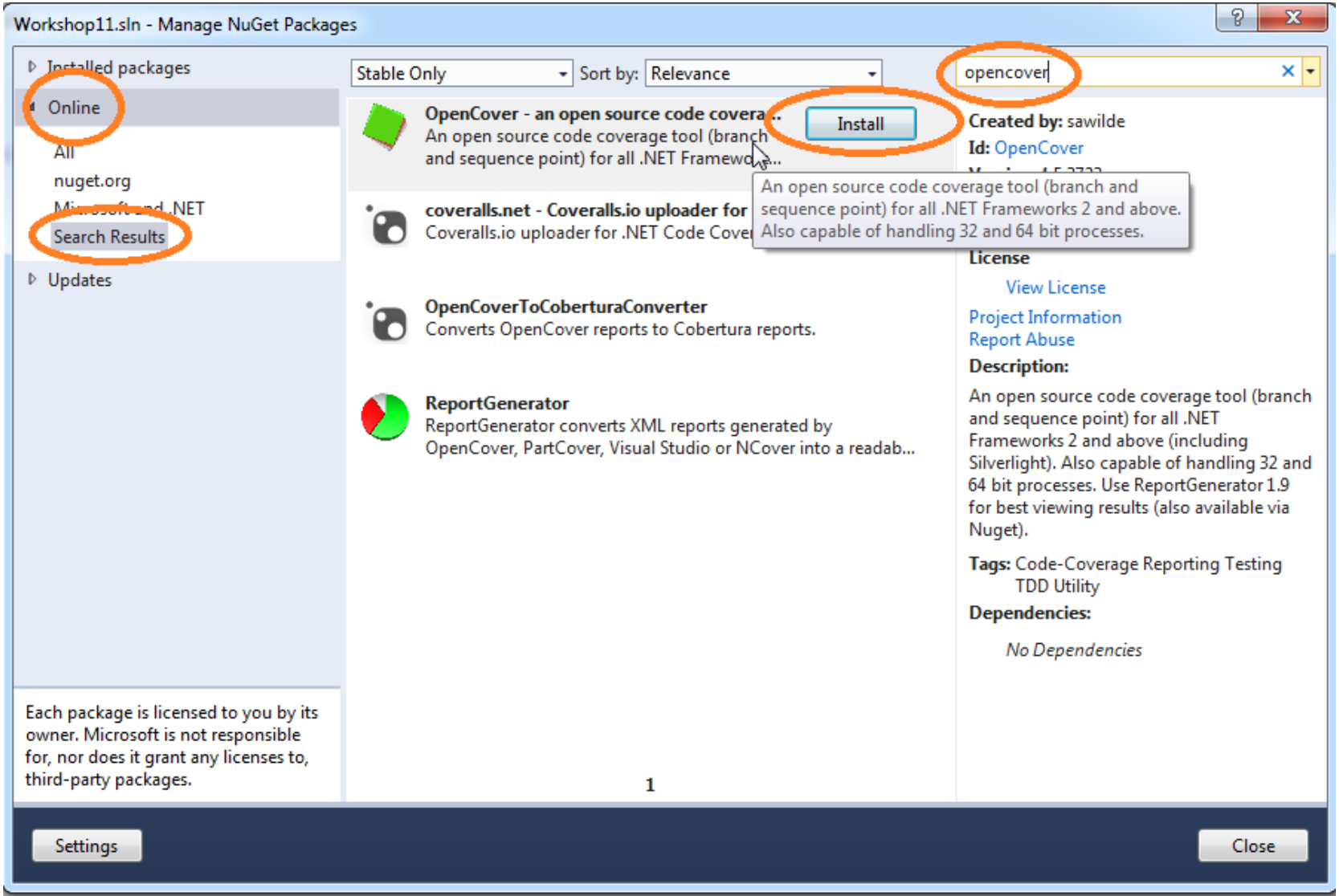
Testing

Importing required libraries via NuGet



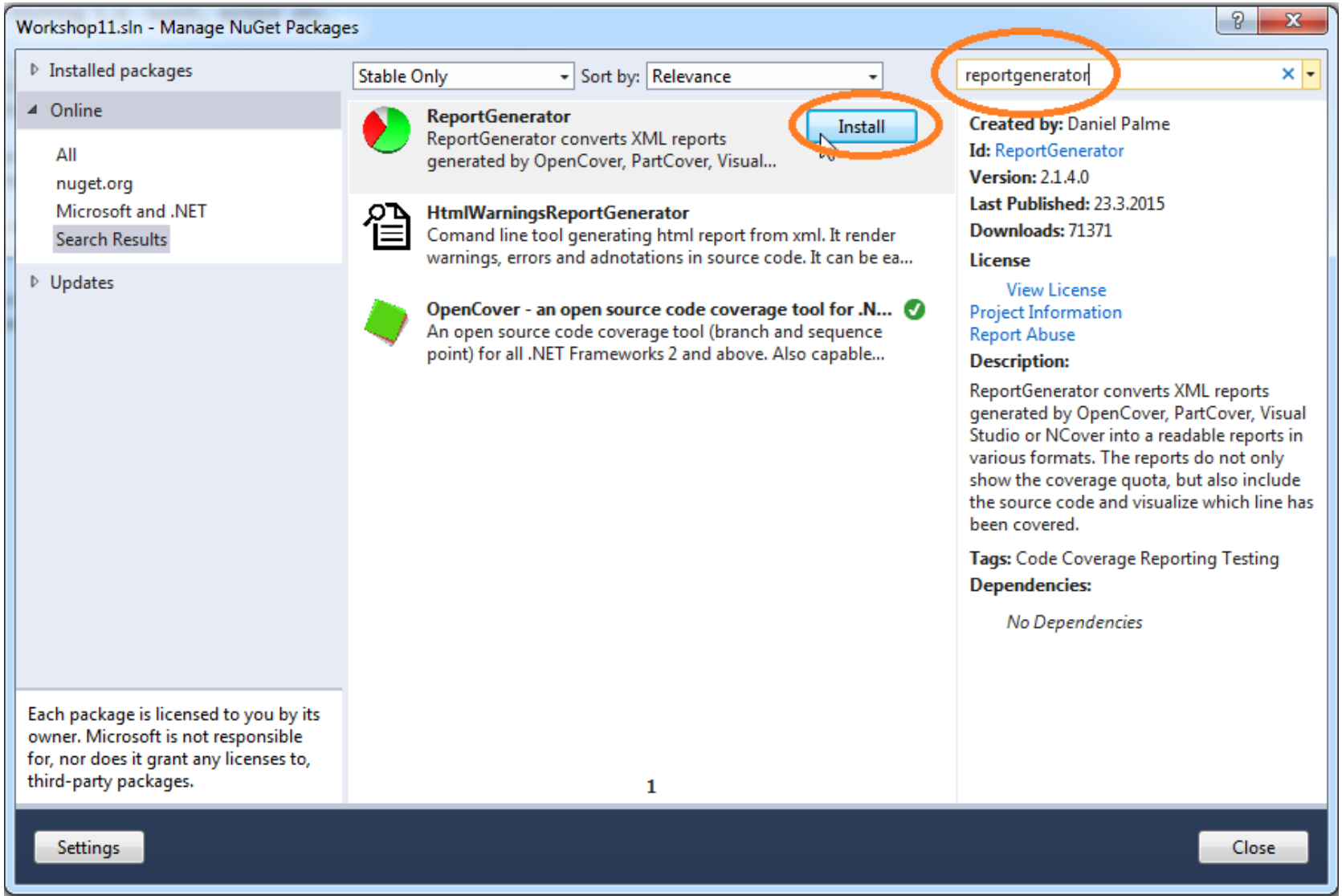
Testing

Importing required libraries via NuGet



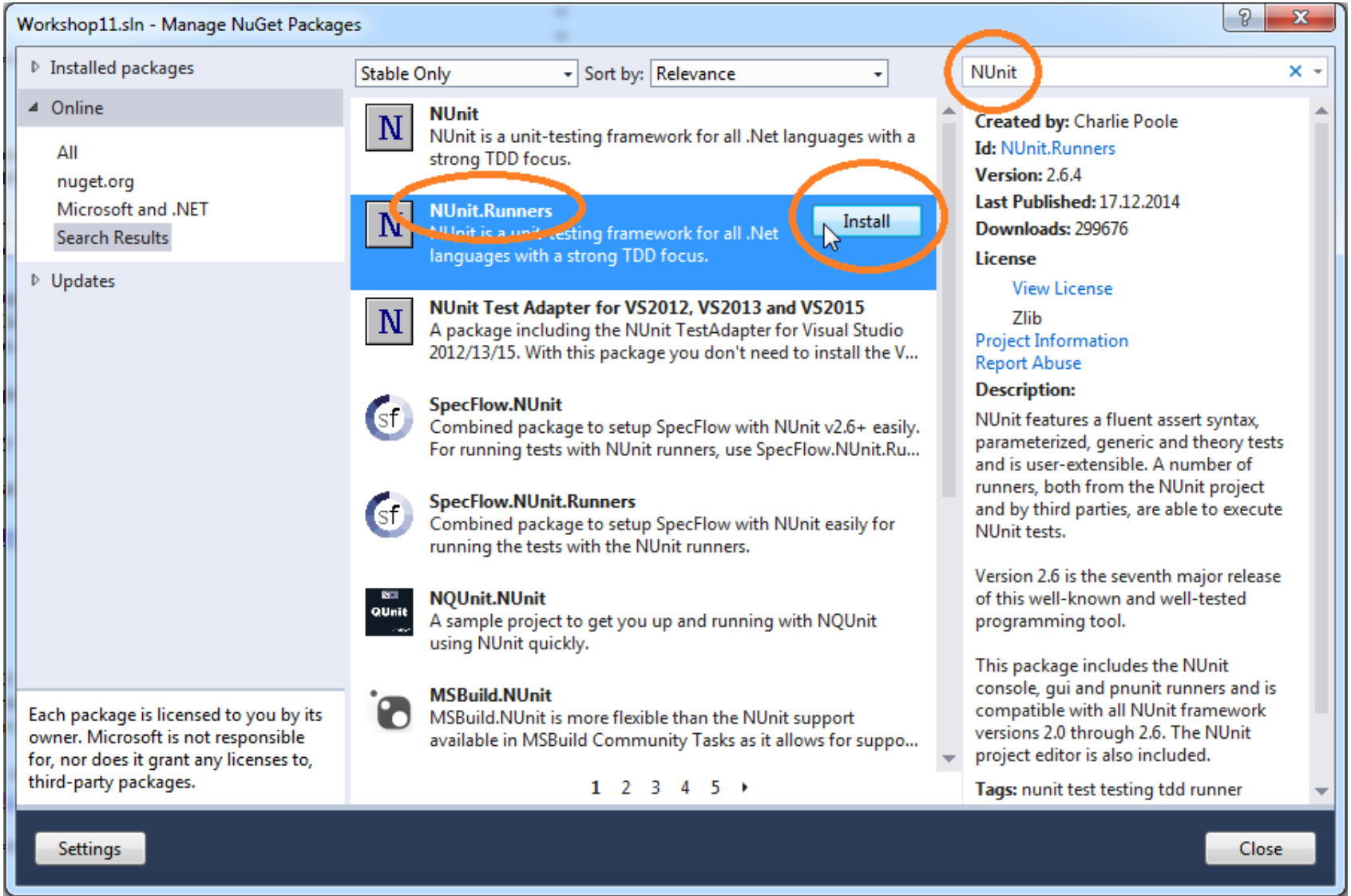
Testing

Importing required libraries via NuGet



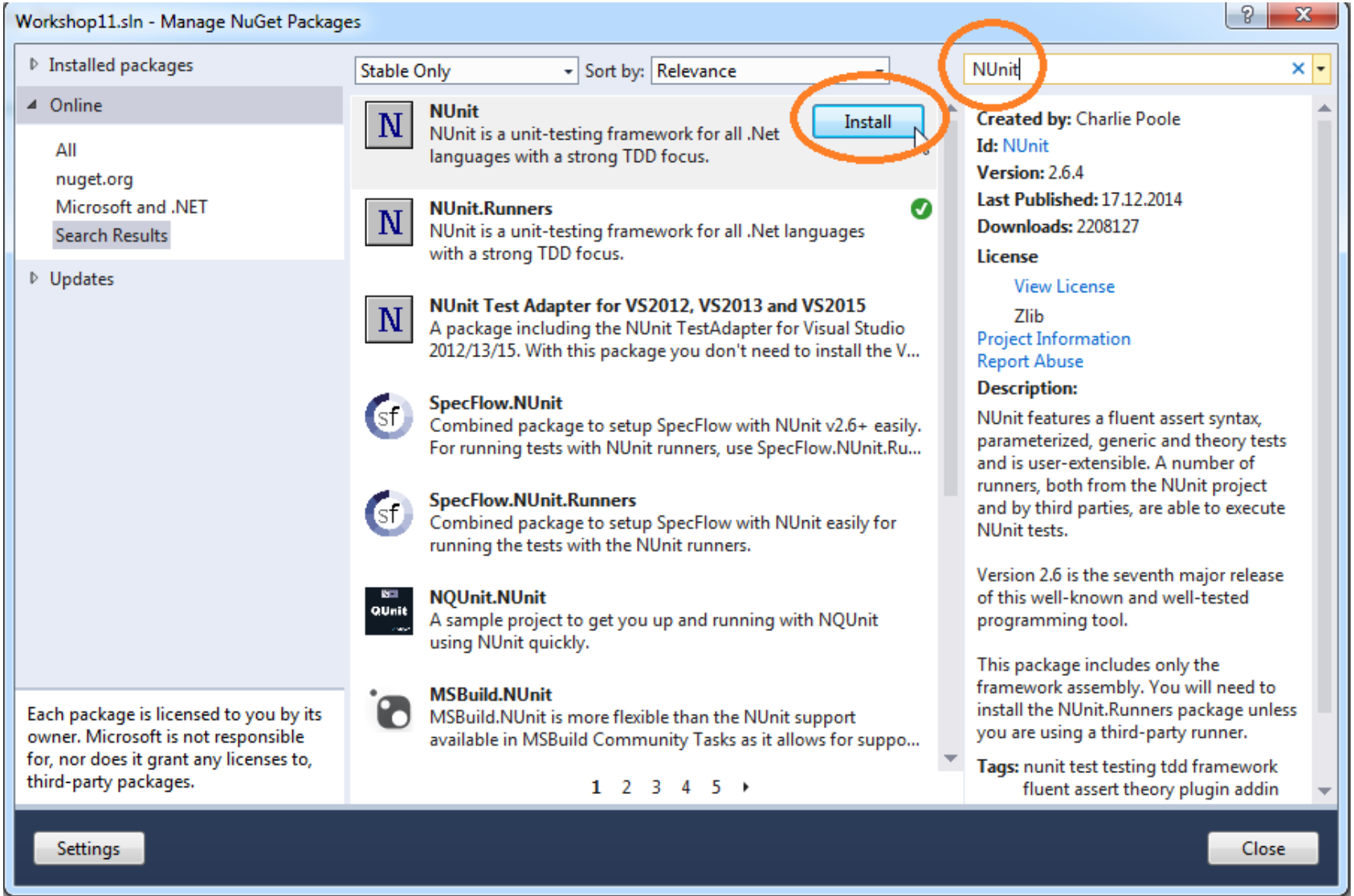
Testing

Importing required libraries via NuGet



Testing

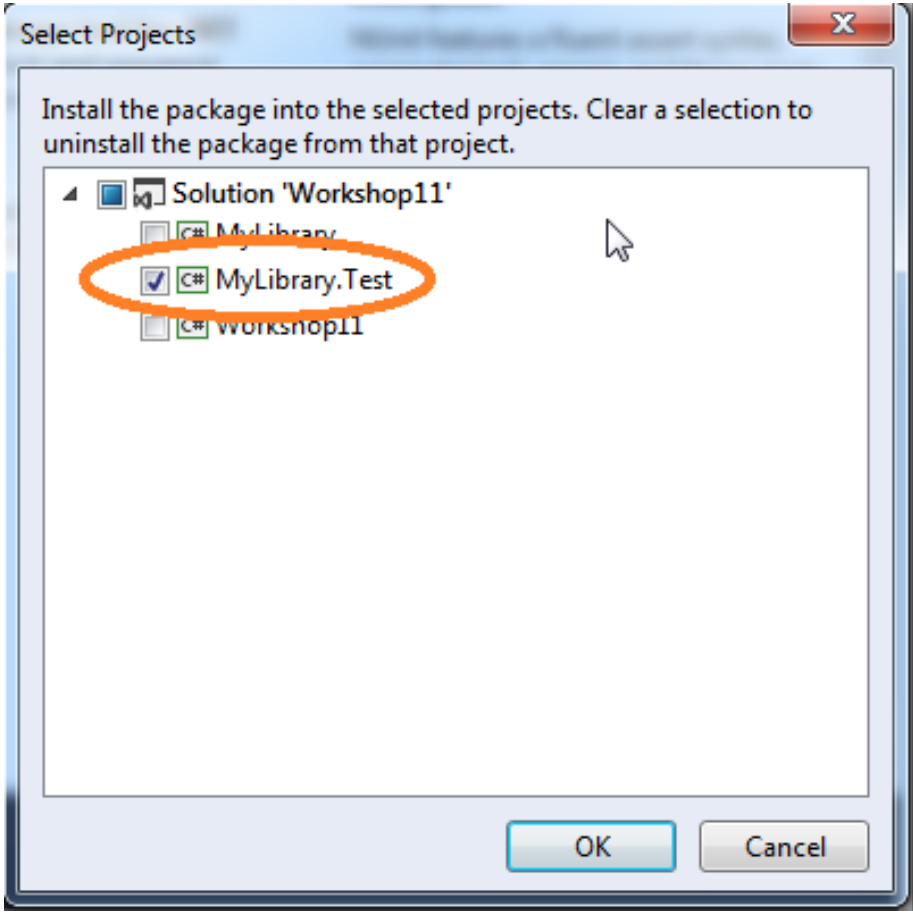
Importing required libraries via NuGet



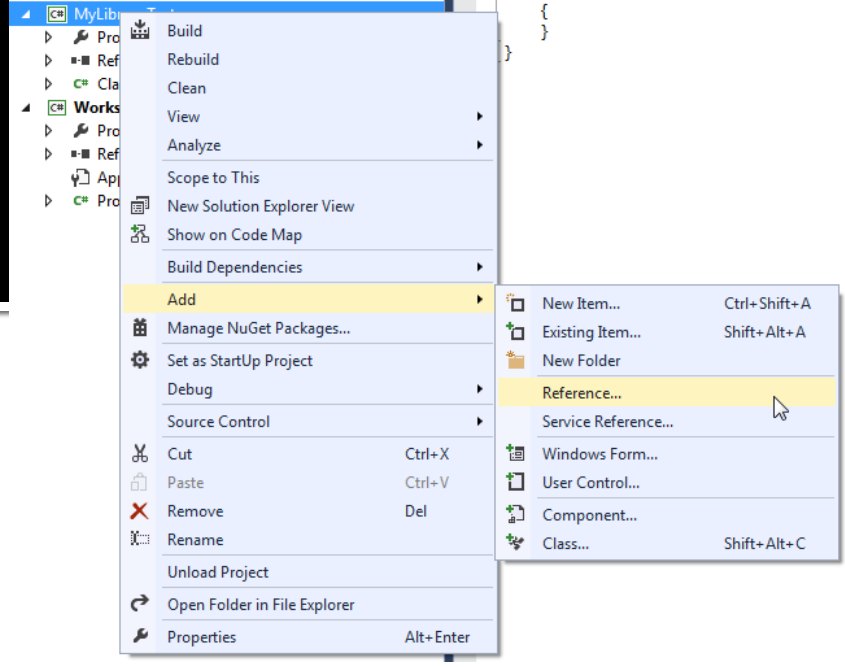
Testing
Reference NUnit

After you install NUnit, you will be asked, which projects will need to use it.

Select your "second" project, that is "one containing TESTs". We have suffixed its name with ".Test".



Testing
Add Reference

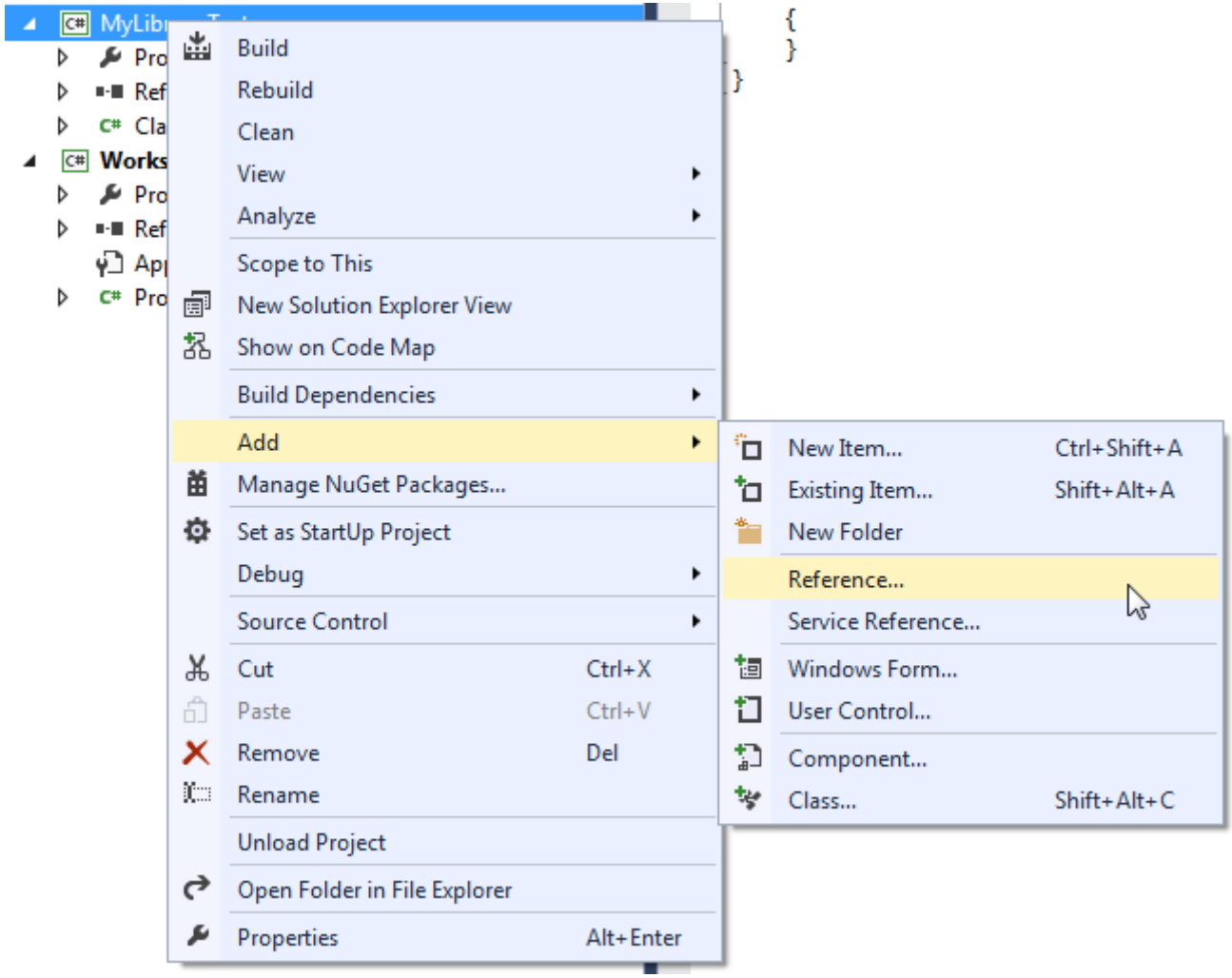


In order to be able to test code that exists in MyLibrary, we have to tell Visual Studio that project MyLibrary.Test references MyLibrary in order to be able to use namespaces from MyLibrary within MyLibrary.Test.

Testing

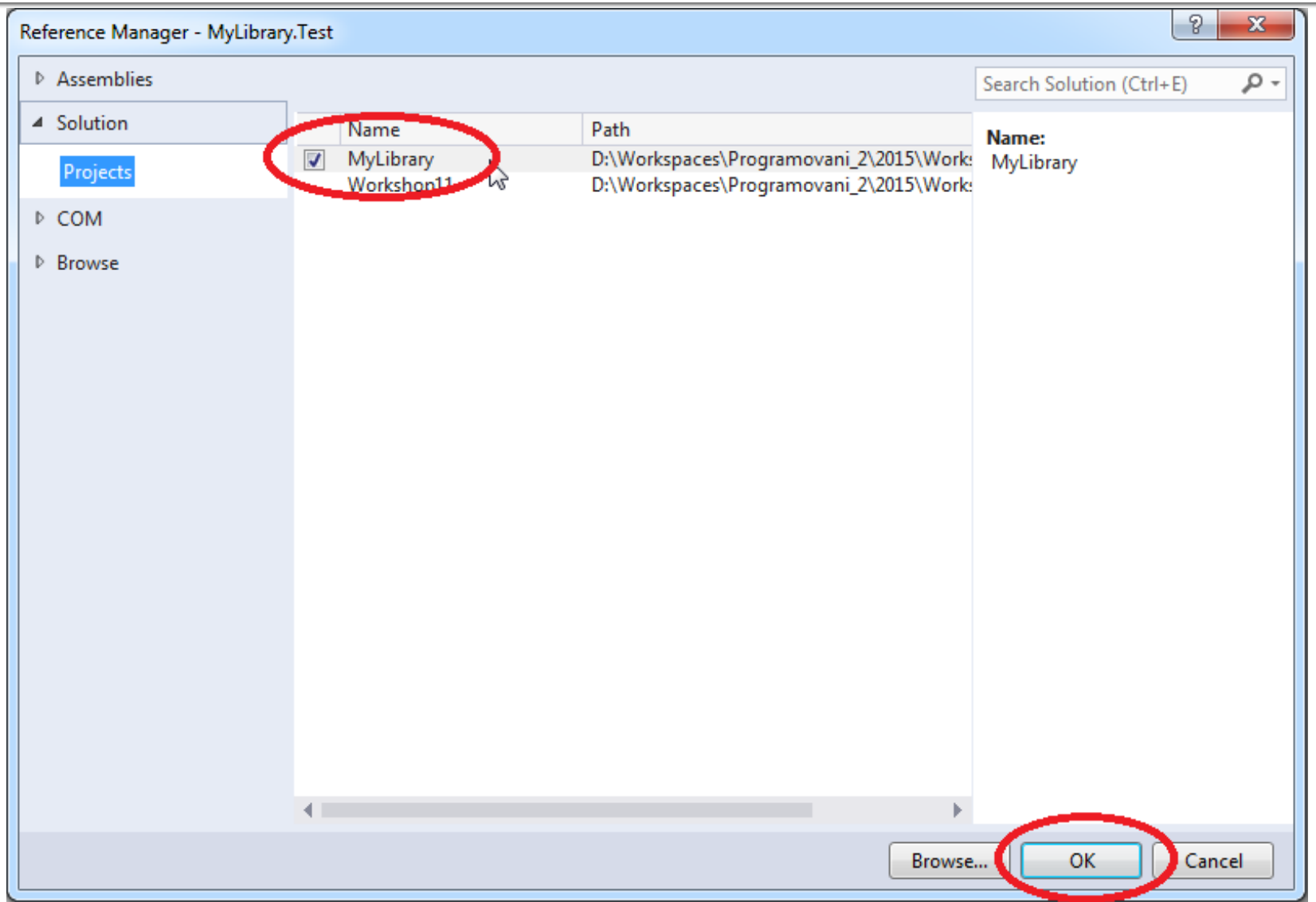
Add Reference to MyLibrary

Right-click MyLibrary.Test and navigate to Add-> Reference.



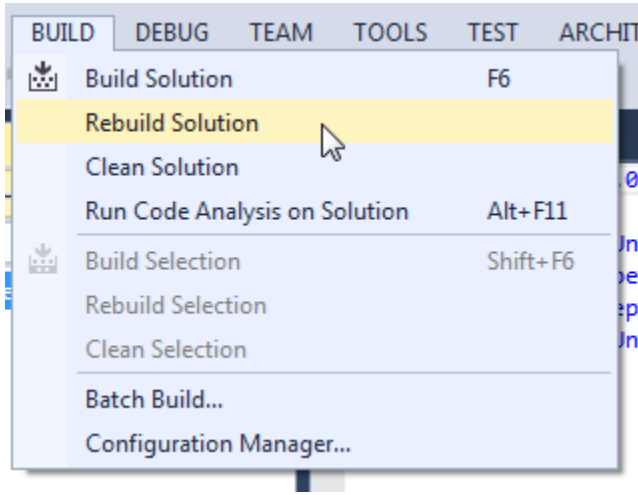
Testing

Add Reference to MyLibrary



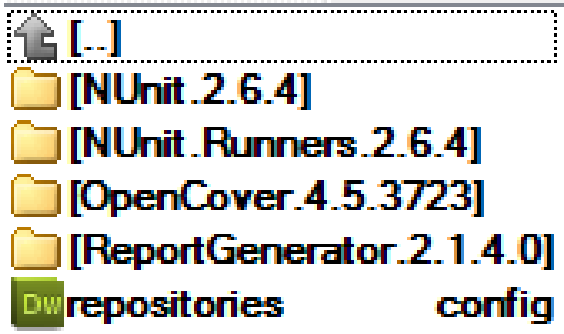
Testing
Build the solution

When "building" the solution Visual Studio will create .dll files of your libraries. We will need both to perform tests and generate reports about code coverage.



Testing
Running the tests

Navigate to the folder of your solution.
There you will find subdirectory
‘packages’.
The content should looke like the picture
to the right.



Testing
Running the tests

Now you will need to create 3 batch files that will

1. Run tests
2. Perform code coverage
3. Generate HTML report

Put those batch files into the folder of your solution.

Testing
Running the tests

Batch File 1: test.bat
2 lines

```
del TestResult.xml
.\packages\NUnit.Runners.2.6.4\tools\nunit-console.exe .\MyLibrary.Test\bin\Debug\MyLibrary.Test.dll /noshadow
```

del TestResult.xml
.\packages\NUnit.Runners.2.6.4\tools\nunit-console.exe
.\MyLibrary.Test\bin\Debug\MyLibrary.Test.dll /noshadow

You might need to adjust texts in red to match your configuration

Explanation: here we're running NUnit that executes code within your ".Test" project producing "TestResult.xml" file with the report.

Testing
Running the tests

Batch File 2: test-cover.bat

2 lines

```
del results.xml
.\packages\OpenCover.4.5.3723\OpenCover.Console.exe -target:test.bat -register:user -filter:+[MyLibrary]*
```

del results.xml
.\packages\OpenCover.4.5.3723\OpenCover.Console.exe -target:test.bat -register:user -filter:+[MyLibrary]*

You might need to adjust texts in red to match your configuration

Explanation: here we're running tests again but now under observation of OpenCover that will generate Code Coverage report for namespace "MyLibrary" (that's why you might need to change that…).

Testing
Running the tests

Batch File 3: test-cover-report.bat
3 lines

```
call test-cover.bat
.\packages\ReportGenerator.2.1.4.0\reportgenerator.exe -reports:results.xml -targetdir:coverage
start firefox file://%CD%/coverage/index.htm
```

call test-cover.bat
.\packages\ReportGenerator.2.1.4.0\reportgenerator.exe -
reports:results.xml -targetdir:coverage
start firefox file://%CD%/coverage/index.htm

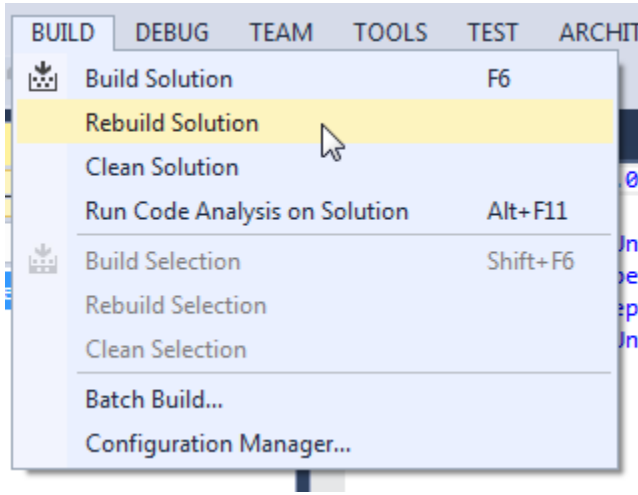
You might need to adjust text in red to match your configuration.

Explanation: Here we run ReportGenerator on the report generated by the OpenCover producing HTML pages visualizing the report.

Testing

Be sure to rebuild the solution

Don’t forget to rebuild your solution every time you do any changes to any of your projects!



Extreme Programming
Task – Visualization of Binary Search Tree

- Download the template: **http://alturl.com/vebg2**
  - http://artemis.ms.mff.cuni.cz/gemrot/lectures/prg2/2015/Workshop11-Homework.zip

- Code Heap tests to provide complete code coverage!

Assignment 11
Send me an email

- Email: jakub.gemrot@gmail.com
- Subject: **Programming II – 2015 – Assignment 11**
- Zip up the whole solution and send it
- You WILL NOT find the assignment in CoDex!
- Deadline:
  - **10.5.2015 23:59**
- Points: 10 + 3 (meeting the deadline)

Questions?
I sense a soul in search of answers…

- Sadly, I do not own the patent for perfection (and will never do)
- In case of doubts about the assignment or some other problems don't hesitate to contact me!
  - Jakub Gemrot
    - gemrot@gamedev.cuni.cz