Code Quality

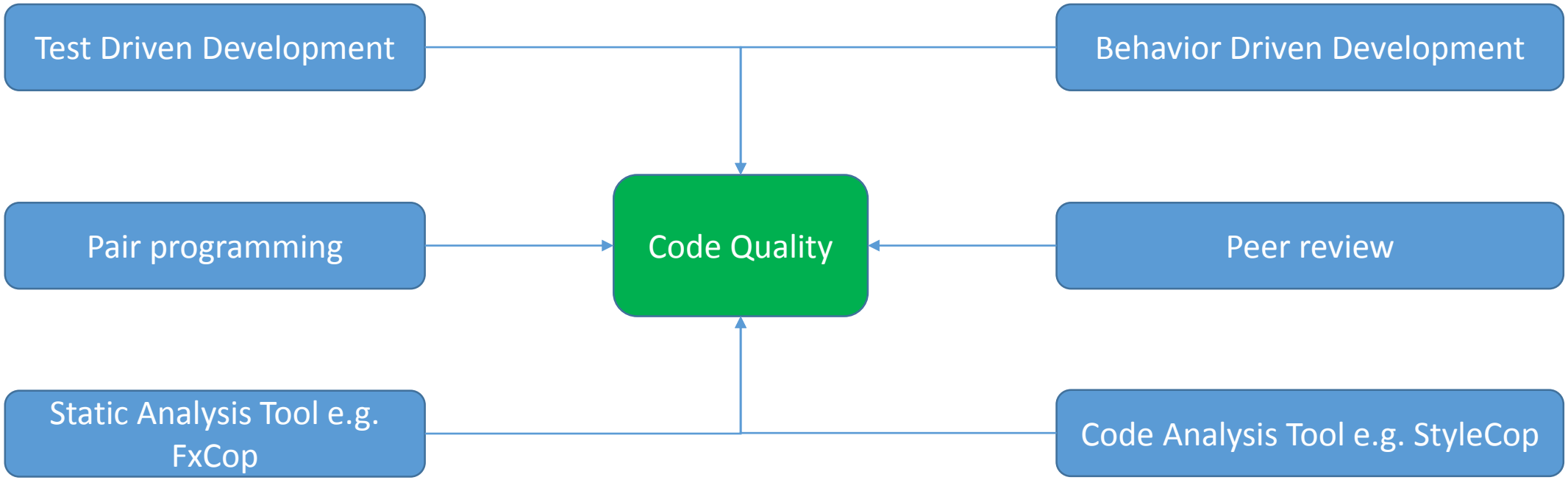
Part I

Rohit Sardana

Agenda

- Introduction
- Scope
- What to expect
- Setting up the development environment
- Setting up test application
- Code walk through
- Analyzing Result
- What’s next

Introduction



Scope

- Test Driver Development
- Tools
  - Unit Testing tools http://www.c-sharpcorner.com/UploadFile/Santhi.M/comparison-of-unit-testing-tools-in-net/
    - **Nunit** : Open Source, Selective running of test cases, Integration with 3rd party tools (any script i.e rake/fake etc., Jenkins, TeamCity), Reliable, Reports generation, lots of documentation
    - MSTest : Integrated with Visual Studio, no rich reporting, Automatically links bugs in TFS
    - Xunit : Similar to Nunit, Integrated tightly with MSBuild, lack of documentation
  - Mocking tool
    - Rhino, **Moq**, NSubstitute
  - Code Coverage tool
    - Ncover, PartCover, dotCover, **OpenCover**
  - Report Generator tool
    - ReportGenerator

What to expect

Summary

Generated on:	16-Jan-18 - 1:49:23 PM
Parser:	OpenCoverParser
Assemblies:	2
Classes:	2
Files:	2
Covered lines:	7
Uncovered lines:	9
Coverable lines:	16
Total lines:	49
Line coverage:	43.7%

Assemblies

Collapse all | Expand all

Grouping: By assembly

Filter:

Name	Covered	Uncovered	Coverable	Total	Line coverage	Branch coverage
App.BL	**0**	**6**	**6**	**21**	**0%**	
App.BL.CalculatorBl	0	6	6	21	0%	
App.Service	**7**	**3**	**10**	**28**	**70%**	
App.Service.CalService	7	3	10	28	70%	

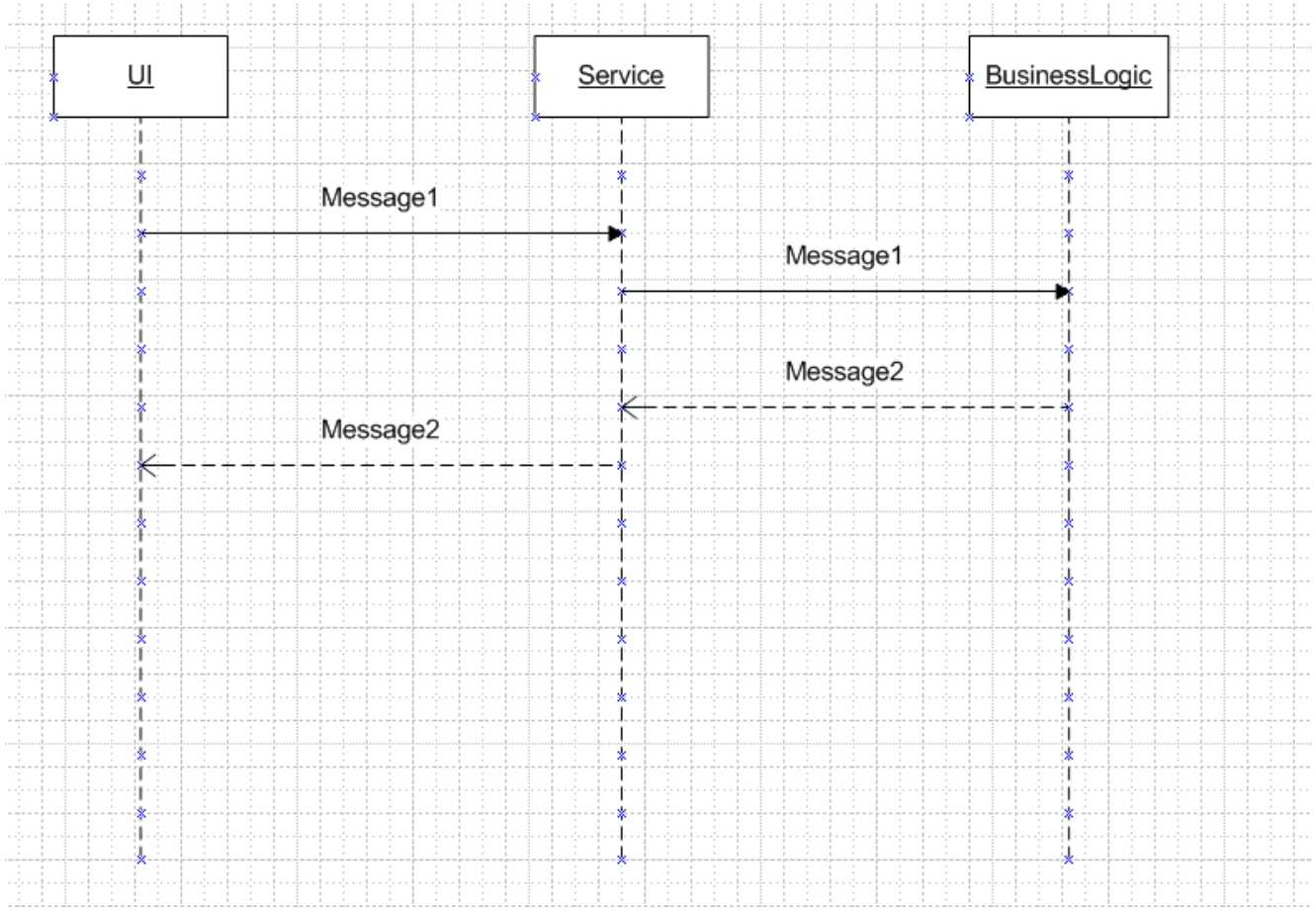
Generated by: ReportGenerator 2.5.9.0
16-Jan-18 - 1:49:23 PM
GitHub | www.palmmedia.de

Setting up Environment

- Install .NET Core SDK 1.0.3
  - Browse the directory
    - Check C:\Prorgram Files\dotnet\shared\Microsoft.NETCore.App for installed Runtimes
    - Check C:\Prorgram Files\dotnet\sdk for installed SDKs
  - Command Line tool
    - dotnet - -version
    - dotnet - -info

  dotnet runtime 1.1.1. is compatible with sdk 1.0.3
- Download (we need them while creating coverage.bat file)
  - OpenCover 4.6.519.zip
    - **OpenCover** is a code coverage tool
  - ReportGenerator 2.5.9.0
    - **ReportGenerator** converts xml reports generated by Coverage tool into human readable reports in various formats.

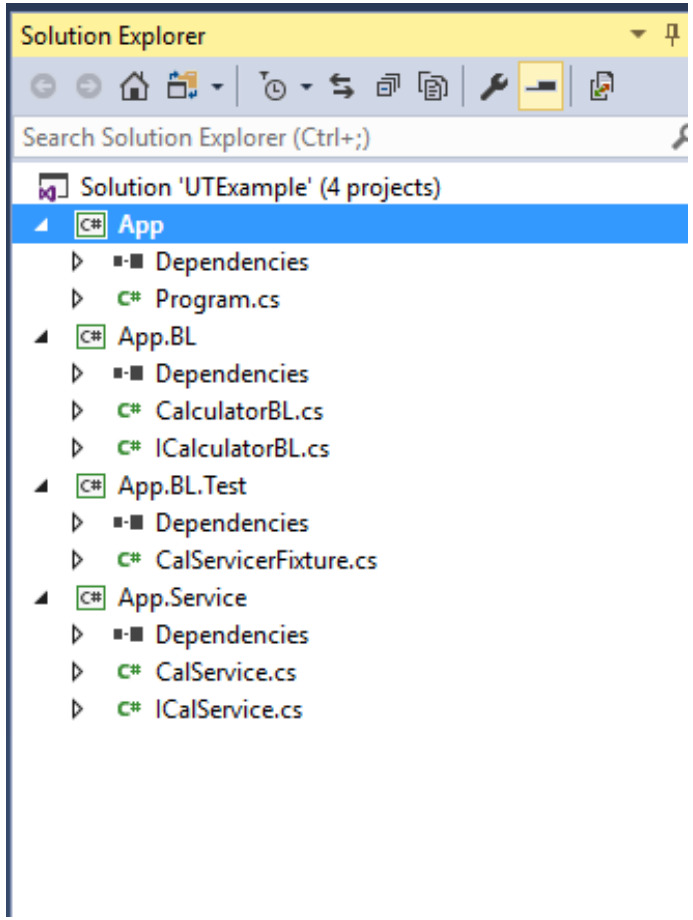
Setting up test project



Setting the test project

- Visual Studio -> .NET Core -> Console App
  - App.BL.Test project should install
    - Nunit 3.7.1
    - NUnitConsoleRunner 3.6.1
    - NUnit3TestAdapter 3.8.0
    - Moq 4.7.63

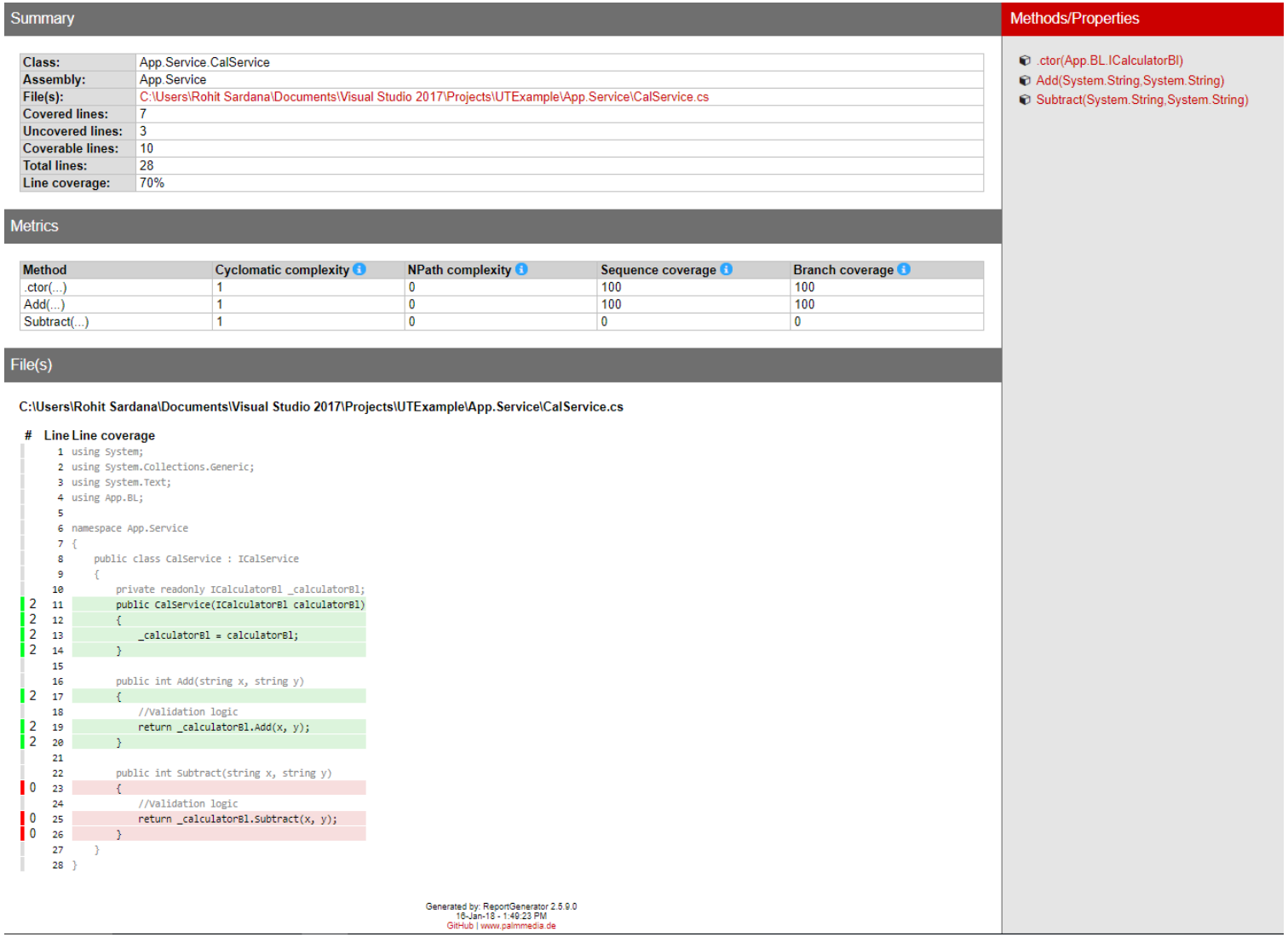
- **NUnit3TestAdapter** allows the built-in Visual studio unit test runner to pick up and run NUnit3 tests.
- **NUnitConsoleRunner** is a text base runner to run tests. It doesn't show graphical indicator (green/red) It automatically saves its result in XML format, allowing you to produce reports or otherwise process results.
- **Moq** Mocking library



Code walkthrough

Code walkthrough i.e.
https://github.com/RohitSardana/CodeQuality.git

Analyzing Result



Summary

Class:	App.Service.CalService
Assembly:	App.Service
File(s):	C:\Users\Rohit Sardana\Documents\Visual Studio 2017\Projects\UTExample\App.Service\CalService.cs
Covered lines:	7
Uncovered lines:	3
Coverable lines:	10
Total lines:	28
Line coverage:	70%

Metrics

Method	Cyclomatic complexity	NPath complexity	Sequence coverage	Branch coverage
.ctor(...)	1	0	100	100
Add(...)	1	0	100	100
Subtract(...)	1	0	0	0

File(s)

C:\Users\Rohit Sardana\Documents\Visual Studio 2017\Projects\UTExample\App.Service\CalService.cs

#	Line	Line coverage
	1	using System;
	2	using System.Collections.Generic;
	3	using System.Text;
	4	using App.BL;
	5	
	6	namespace App.Service
	7	{
	8	public class CalService : ICalService
	9	{
	10	private readonly ICalculatorBl _calculatorBl;
2	11	public CalService(ICalculatorBl calculatorBl)
2	12	{
2	13	_calculatorBl = calculatorBl;
2	14	}
	15	
	16	public int Add(string x, string y)
2	17	{
	18	//Validation logic
2	19	return _calculatorBl.Add(x, y);
2	20	}
	21	
	22	public int Subtract(string x, string y)
0	23	{
	24	//Validation logic
0	25	return _calculatorBl.Subtract(x, y);
0	26	}
	27	}
	28	}

Generated by: ReportGenerator 2.5.9.0
16-Jan-18 - 1:49:23 PM
GitHub | www.palmmedia.de

Methods/Properties

- .ctor(App.BL.ICalculatorBl)
- Add(System.String,System.String)
- Subtract(System.String,System.String)

↓ Cyclomatic complexity ↑ Code

↓ NPath complexity ↑ Code

NPath complexity of 16 means that if you need 100% code coverage you need to test for 16 possible outcomes.

Statement Coverage: what lines have been covered.
Method Coverage: what methods have been covered.
Branch Coverage: which branches were taken. This is related to cyclometric complexity
Sequence Coverage: lines covered represented in terms of sequence. One sequence point can span several lines of code.

Visit OpenCover or ReportGenerator documentation for more details.

What’s Next

- Dependency Injection tools such as StructureMap, Unity
- Behavior Driven Development
- Integrating it with Continuous Integration Server/Build server
- ….

Thank You