

<https://www.halvorsen.blog>



# Week Assignment

Software Testing – Test Execution

Hans-Petter Halvorsen

# Week Assignment

1. Start to Test your Software according to STP
2. Create Unit Tests for your Code
3. Scrum Activities and Meetings
  - Finishing Beta Iteration
    - Sprint Review Meeting)
    - Sprint Retrospective Meeting
  - Start Working on Next Iteration (RC)
    - Sprint Planning Meeting



In software engineering, a freeze is a point in time in the development process after which the rules for making changes to the source code or related resources become more strict, or the period during which those rules are applied. [https://en.wikipedia.org/wiki/Freeze\\_\(software\\_engineering\)](https://en.wikipedia.org/wiki/Freeze_(software_engineering))

Code Freeze: Tuesday 10:15-14:00 and Friday 10:15-14:00



~~Visual Studio~~

No Programming in Class these 2 weeks! – otherwise it is easy to loose focus on Testing

# Why Testing?

- Make sure the software fulfills the **Requirements** from the Customers (Software Requirements Specification, SRS)
- Make sure the Software don't contain critical **Bugs**
- Make sure the software can be **installed** at the customer. The customer don't have Visual Studio!
- Make sure the software are **user-friendly** and intuitive to use
- Make sure the software is **robust** and has acceptable **performance** (so it don't crash when more than 1 person are using it, or if the database contain lots of data, etc.)

# Test Planning and Execution

1

Create Software Test Plan (STP)

2

Create Virtual Test Environment

3

Test the Software according to STP

4

Create Unit Tests in Visual Studio

Test Planning

Previous Week

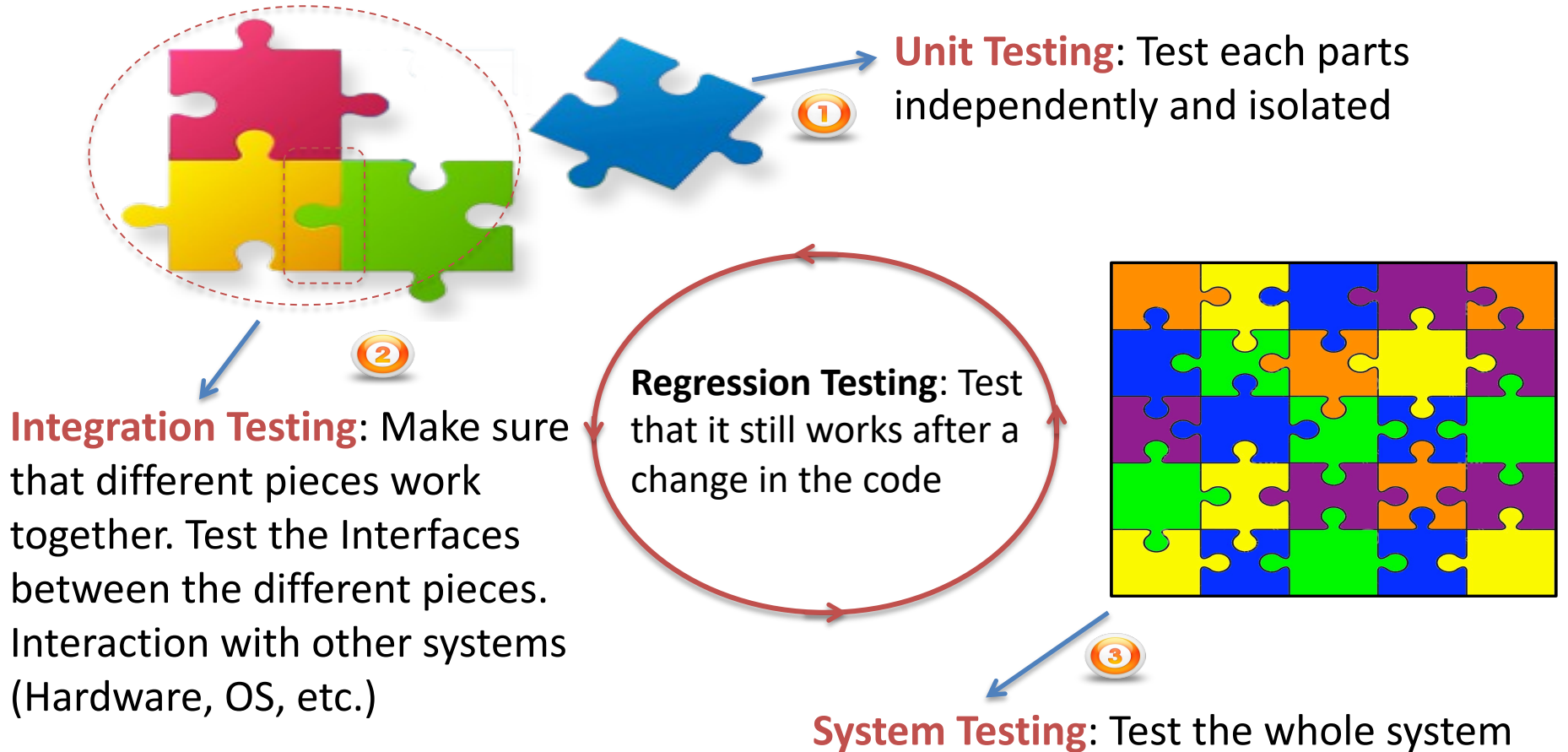
Test Execution

This Week



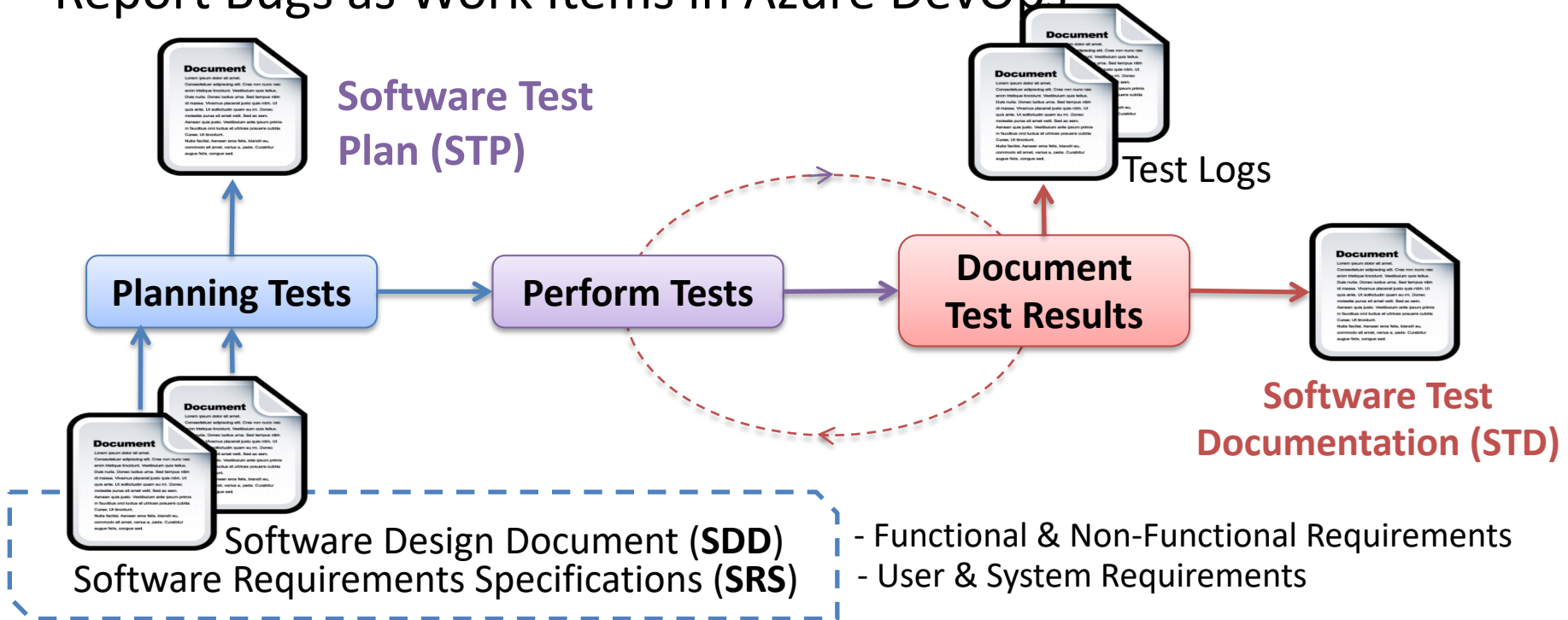
# Software Testing

# Levels of Testing



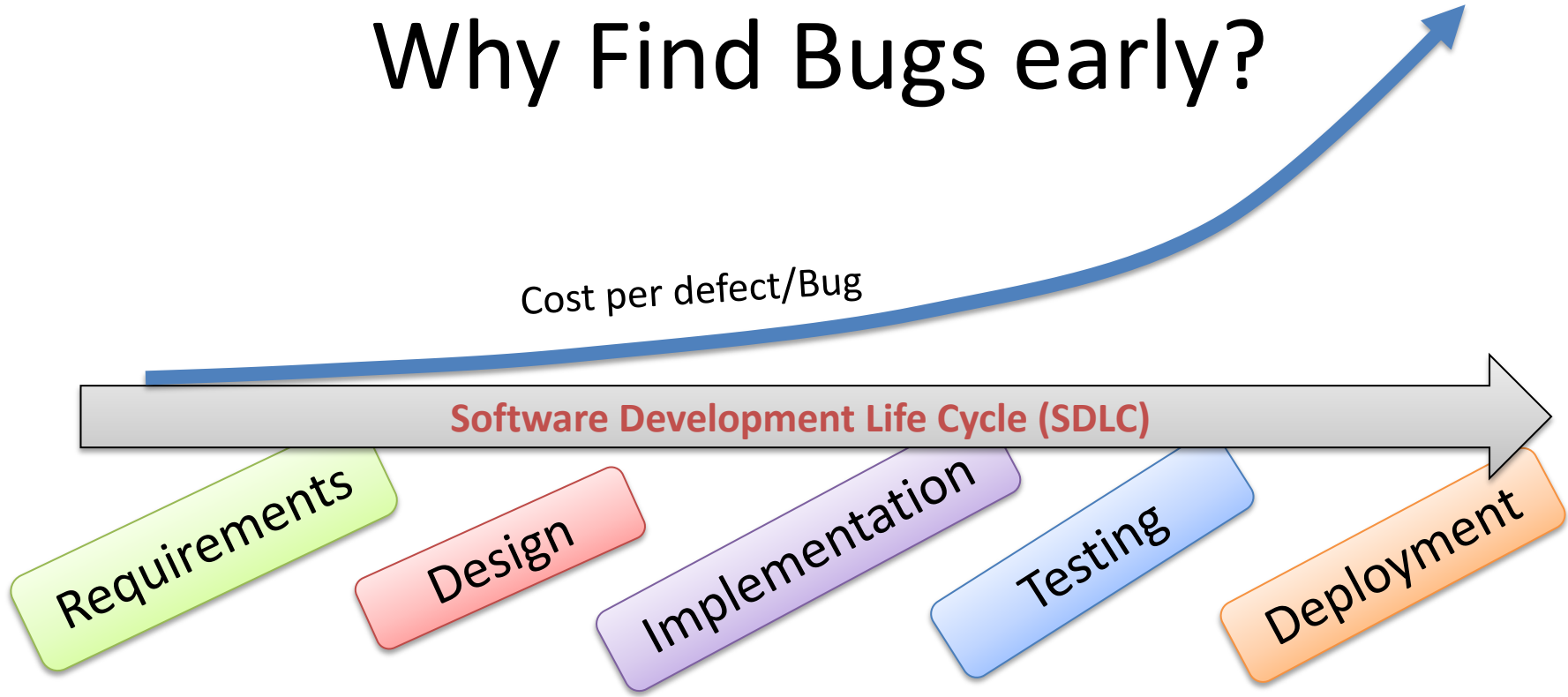
# Testing

- Test your software according to the **Requirements and Design Documents** and the **Software Test Plan**
- Report Bugs as Work Items in Azure DevOps

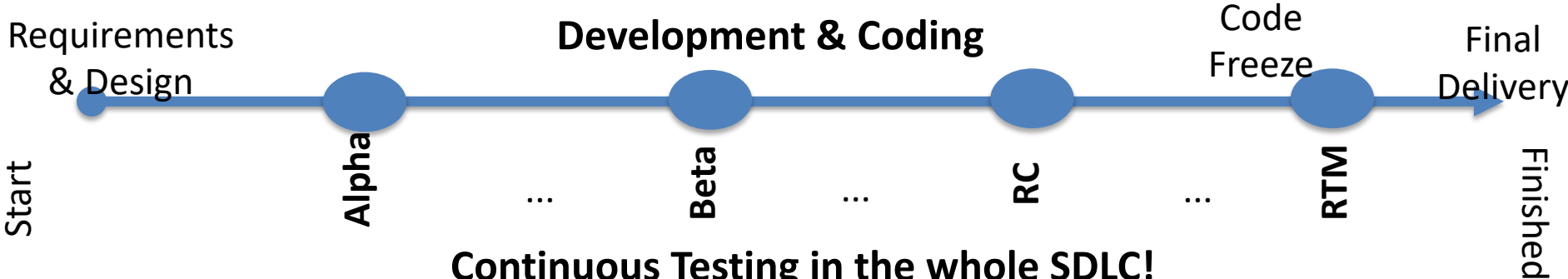




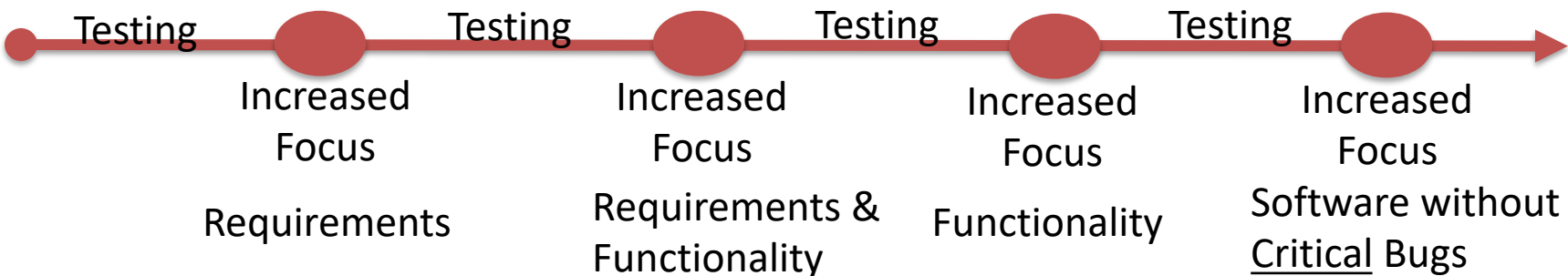
# Why Find Bugs early?



# Testing



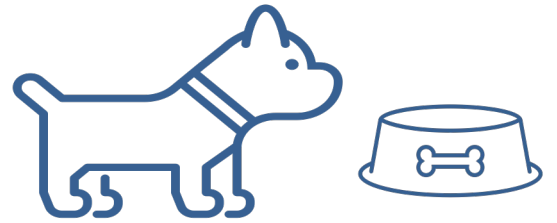
## Continuous Testing in the whole SDLC!



Agile/Scrum: Periodically Iterations/Sprint every 14-30 days

You can never find all Bugs!  
Released Software do have Bugs!

# Eat your own dog food



- You are not giving food to your dog without first taste it yourself?
- You are not giving things to others that you dont like yourself?
- This means that you should use the Software you create yourself on a daily basis.
- If you are happy with the Software and are able to use it, most likely the Customer will also like it and be able to use it



# Test your Software

Hans-Petter Halvorsen

[Table of Contents](#)

# Test your Software

- Test your system according to the **Software Test Plan (STP)**.
  - Make sure to test the part of the Software that you have not created
- Make sure to document the Test Results.
  - Bugs should be in addition be reported as “Bug” Work Items in **Azure DevOps**.
- Create **Queries** in Azure DevOps (a “Buglist”).
  - Prioritize and Fix the Bugs according to the List of Bugs
- Make **Software Test Documentation (STD)**

See Next Slides for more details...

# Test Execution and Reporting

Name: \_\_\_\_\_

One of the students submitting this form will receive a prize!

Start Testing the entire Software (and Documentation)

- My Test Environment is ready
- Read the **STP** and start testing according to STP
- Execute and Fill in **Test Cases**.
- Try to break the system. Do the unexpected!, go crazy!
- Each Person should Report at least **10 Bugs**

# Bugs: \_\_\_\_\_

# New Features: \_\_\_\_\_

or **Features** in **Azure DevOps** (Work Items)

- Each Person should create at least **2 personal (My Queries) Queries and 2**

**Shared Queries in Azure DevOps**. Examples:

- Bugs Assigned to Me
- Bugs Reported by Me
- New Bugs Last 24 Hours, etc.

My Query 1: \_\_\_\_\_

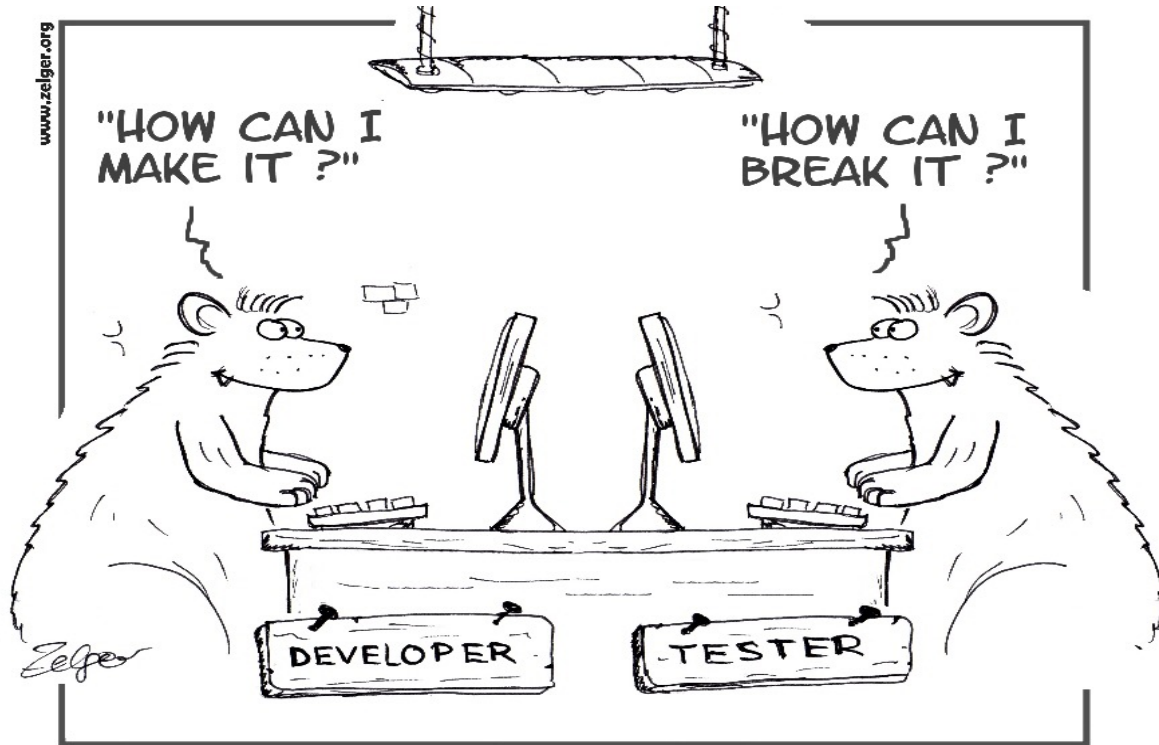
My Query 2: \_\_\_\_\_

Team Query 2: \_\_\_\_\_

Team Query 2: \_\_\_\_\_

- Upload **Test Cases** you have filled out to **Teams** or **Azure DevOps**
- Prioritize and (later) Fix the Bugs according to the List of Bugs

# Development vs. Testing



They weren't so much different, but they had different goals

You need to think different when Testing compared to when Developing the System

Goal: Find as much bugs, improvements, etc. as possible!!!

# How shall tests be documented?

- It is not enough simply to run tests
- The results of the tests must be systematically recorded.
  - Typically, the Customers wants to see documentation that the system has been properly tested
  - Or the Quality Manager will not approve that you release a Software without seeing the documentation that the system has been properly tested
- It must be possible to audit the testing process to check that it has been carried out correctly
- How you do this must be described in the Software Test Plan (STP)



# Software Test Documentation (STD)

Suggestions of contents:

- Executed Test Cases that the Testers have filled out (Passed/Not Passed, Comments, etc.)
- Export of reported Bugs from Azure DevOps
- Some Plots/Charts that gives an overview of number of Bugs Reported and Fixed, etc.
- In addition, some text explaining these things

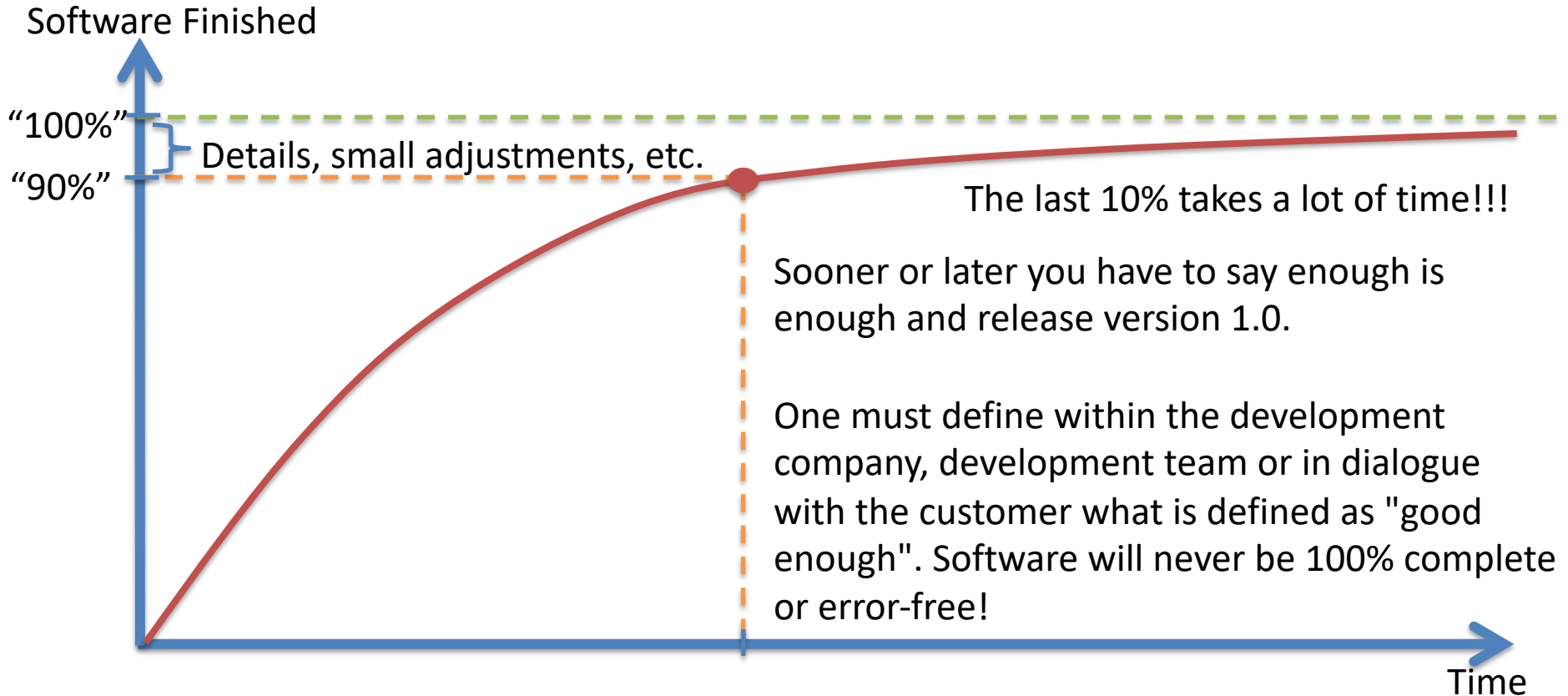
# STD Contents Examples

- Code Reviews
- Test Environment
- Executed Test Cases
- Test Results
- Bug Reports, Bug Queries
- Analysis of Testing and Test Results (Tables, Charts, etc.)
  - Number of Bugs found, Number of fixed bugs, remaining bugs, etc.
- Discussions and Conclusions
  - Is the system ready for release?

Details can be in Appendices or available as separate documents

→ STD document and details should be available from HTML Web Site

# When to Stop Development?



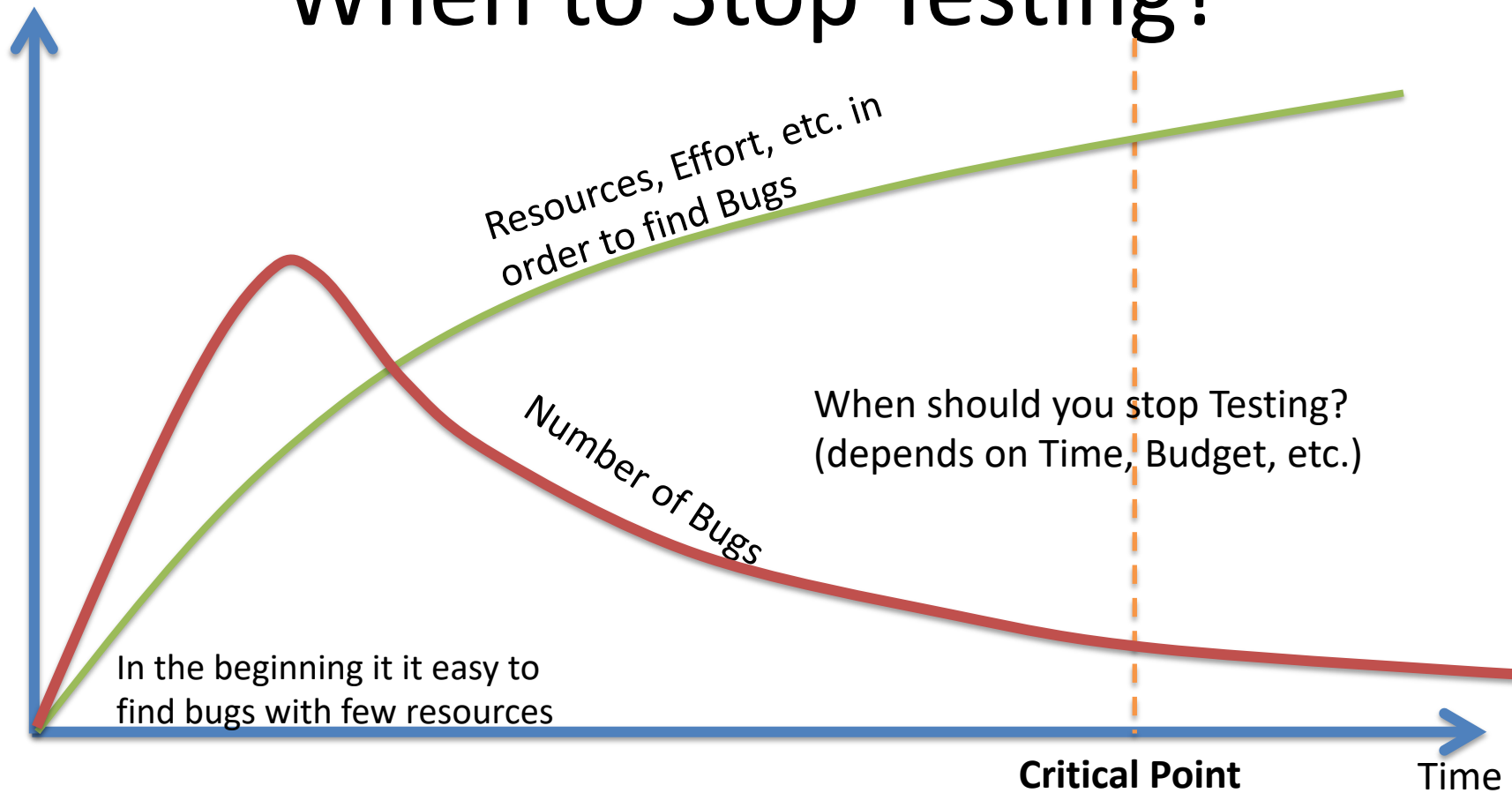
# When to Stop Testing?

<http://dilbert.com>



A simple answer is to stop testing when all the planned test cases are executed and all the problems found are fixed. In reality, it may not be that simple. We are often pressured by schedule to release software product.

# When to Stop Testing?



# When to Stop Testing?

- When the tester has not been able to find another defect in 5 (10? 30? 100?) minutes of testing
  - All code reviews and walkthroughs have certified the code as ok
  - When a given checklist of test types has been completed
  - The code has passed all unit tests
  - When testing runs out of its scheduled time
  - ...
- These things needs to be documented in the Software Test Plan (STP)

# 80 – 20 Rule

- It takes 20% of the time to finish 80% of your application -> “Prototype” (80% finished)
- 80% of the users only use 20% of the features
- 80% of performance improvements are found by optimizing 20% of the code
- **80% of the bugs are found in 20% of the code**

# Work Items – New Bug

We use Azure DevOps as our Bug Reporting Tool

New Bug 1\*: WS is not working

Copy template URL

Tags Add...

WS is not working

## STATUS

Assigned To *<No one>*  
State *Active*  
Reason *New*

## CLASSIFICATION

Area *Development Project 1\Desktop*  
Iteration *Development Project 1\Beta*

## PLANNING

Stack Rank *<None>*  
Priority *2*  
Severity *3 - Medium*

## REPRO STEPS

### SYSTEM INFO

### TEST CASES

B / U [Rich text editor icons]

## HISTORY

### ALL LINKS

### ATTACHMENTS

B / U [Rich text editor icons]

## DISCUSSION ONLY

### ALL CHANGES

[No entries with comments]



# Queries

- Used to find existing Work Items
- You may create different Queries to make it easy to find the Work Items you need
- Queries may be personal or visible for everybody in the project (Team Queries)

The screenshot shows a query editor interface for a project. At the top, it says "New Query 1" and "5 work items (1 selected)". Below this, there are tabs for "results" and "editor". The interface includes a toolbar with icons for save, refresh, and other actions, along with a "Column Options" button. The "Type of Query" section has three options: "Flat List of Work Items" (selected), "Work Items and Direct Links", and "Tree of Work Items". The "Filters for top level work items" section contains three filter clauses:

	And/Or	Field	Operator	Value
+ X	<input type="checkbox"/>	Team Project	=	@Project
+ X	<input type="checkbox"/> And	Work Item Type	=	[Any]
+ X	<input type="checkbox"/> And	State	=	[Any]








Below the filters, there is a "Save query" button and another "Column Options" button. The main area displays a table of work items:

ID	Work Item...	Title	Assigned To	State	Tags
1	Bug	Database Error	Hans-Pett...	Active	
2	Task	Add Web functionality		New	
4	Test Case	Test Empty Fields	Hans-Pett...	Design	
3	Test Case	Test Web Service	Hans-Pett...	Design	
5	Bug	WS is not working		Active	

# Creating a Query - Example

results [editor](#)    | Column OptionsType of Query  Flat List of Work Items  Work Items and Direct Links  Tree of Work Items

Filters for top level work items

	(☰) And/Or	Field	Operator	Value
 	<input type="checkbox"/>	Team Project	=	@Project
 	<input type="checkbox"/> And	Work Item Type	=	[Any]
 	<input type="checkbox"/> And	State	=	[Any]
	<a href="#">Add new clause</a>			

 Save query      || Column Options

ID	Work It...	Title	Assigned To	State	Tags
1	Bug	Database Error	Hans-Pett...	Active	
2	Task	Add Web functionality		New	
4	Test Case	Test Empty Fields	Hans-Pett...	Design	
3	Test Case	Test Web Service	Hans-Pett...	Design	
5	Bug	WS is not working		Active	

# Work Items Example

New | [Icons]

Assigned to me

Unsaved work items

**My favorites**

- All Bugs
- All My Work Items

**Team favorites**

- All Bugs

**My Queries**

- All My Work Items

**Shared Queries**

- Current Sprint**
- All Bugs
- All Work Items
- Feedback
- My Bugs
- New Features

**All Work Items** 18 work items (1 selected)

Results Editor Charts

Work item pane Bottom [Icon]

Save query [Icons] Column options Copy query URL Filter [Icon]

ID	Work Item Type	Title	Assigned To	State	Created By
100	Product Backlog Item	Introduction	Olav Dæhli	New	Hans-Petter Halvors...
101	Product Backlog Item	Requirement Analysis	Hans-Petter Halvors...	New	Hans-Petter Halvors...
102	Product Backlog Item	Software Design	Olav Dæhli	New	Hans-Petter Halvors...
103	Product Backlog Item	Development Processes	Hans-Petter Halvors...	New	Hans-Petter Halvors...
104	Task	What is System Engineering	Hans-Petter Halvors...	Done	Hans-Petter Halvors...
105	Task	SRS	Hans-Petter Halvors...	In Progress	Hans-Petter Halvors...
106	Task	SDD	Hans-Petter Halvors...	In Progress	Hans-Petter Halvors...
107	Task	ERwin	Olav Dæhli	To Do	Hans-Petter Halvors...
108	Bug	Database Communication fails	Hans-Petter Halvors...	New	Hans-Petter Halvors...
109	Bug	Database Script not Working	Olav Dæhli	New	Hans-Petter Halvors...

List of Work Items

**Product Backlog Item 100: Introduction** 1 of 18

[Icons]

Work Item Details

Tags Add...

Introduction

Iteration Systemutvikling 2015\Release 1\Sprint 1

**STATUS**

Assigned To Olav Dæhli

State New

Reason New backlog item

**DETAILS**

Business Value

Area Systemutvikling 2015

**DESCRIPTION** STORYBOARDS TEST CASES TASKS (1) ACCEPTANCE CRITERIA HISTORY LINKS (1) ATTACHMENTS

Make sure to fill out "Assigned To", Correct "Iteration" and "Area"

You can create Queries (both Personal and Team Queries)



# Unit Testing

# Unit Testing

- Create Unit Tests for your code
  - Not for all your code, just a few examples in order to learn how it is done in practice.
- Each member in the Team should write at least 2 Unit Tests for their code
- Planning to Create New Code? Next Time you should try to Create a Unit Test before you start Coding
- Unit Testing should also be part of the System Documentation (coming up soon)

See Next Slides for more details...

# Create Unit Tests in Visual Studio

Individual Activity

Name: \_\_\_\_\_

One of the students submitting this form will receive a prize!

- Watch/Do the Examples in the Video
- Create at least Unit Tests for 2 different Methods in your code:
  - Unit Test 1: \_\_\_\_\_
  - Unit Test 2: \_\_\_\_\_
- Run the Units Tests:
  - Failed/Passed? \_\_\_\_\_
  - Code Coverage \_\_\_\_\_

# What are Unit Tests

- Unit Testing (or *component testing*) refers to tests that verify the functionality of a specific section of code, usually at the function level.
- **In an object-oriented environment, this is usually at the class and methods level.**
- **Unit Tests are typically written by the Developers as part of the programming**
- **Automatically executed** (e.g., Visual Studio and Azure DevOps have built-in functionality for Unit Testing)



# Unit Tests Frameworks

Unit Tests Framework are usually integrated with the IDE

- **Visual Studio Unit Test Framework.** Unit Tests are built into Visual Studio (no additional installation needed)

Others:

- **JUnit** (Java)
  - JUnit is a unit testing framework for the Java programming language.
- **NUnit** (.NET)
  - NUnit is an open-source unit testing framework for Microsoft .NET. It serves the same purpose as JUnit does in the Java world
- **PHPUnit** (PHP)
- LabVIEW Unit Test Framework Toolkit
- etc.

All of them work in the same manner – but we will use the Visual Studio Unit Test Framework



# Unit Testing and Documentation

How/where shall Unit Testing be included in the documentation?

- Software Test Plan (STP)
  - Test Planning, Unit Testing is an important part of that!
- Test Documentation
  - Documentation, Test execution and Test Results, Unit Testing is an important part of that!
- System Documentation
  - Used to maintain and further development of the system, Unit Testing is an important part of that!
  - More about System Documentation Next Week!

# Unit Tests – Best Practice

- A Unit Test must only do one thing
- Unit Test must run independently
- Unit Tests must not be depend on the environment
- Test Functionality rather than implementation
- Test public behavior; private behavior relates to implementation details
- Avoid testing UI components
- Testing Methods writing/retrieving Data to/from a Database can also be challenging
- Unit Tests must be easy to read and understand
- Create rules that make sure you need to run Unit Tests (and they need to pass) before you are allowed to Check-in your Code in the Source Code Control System

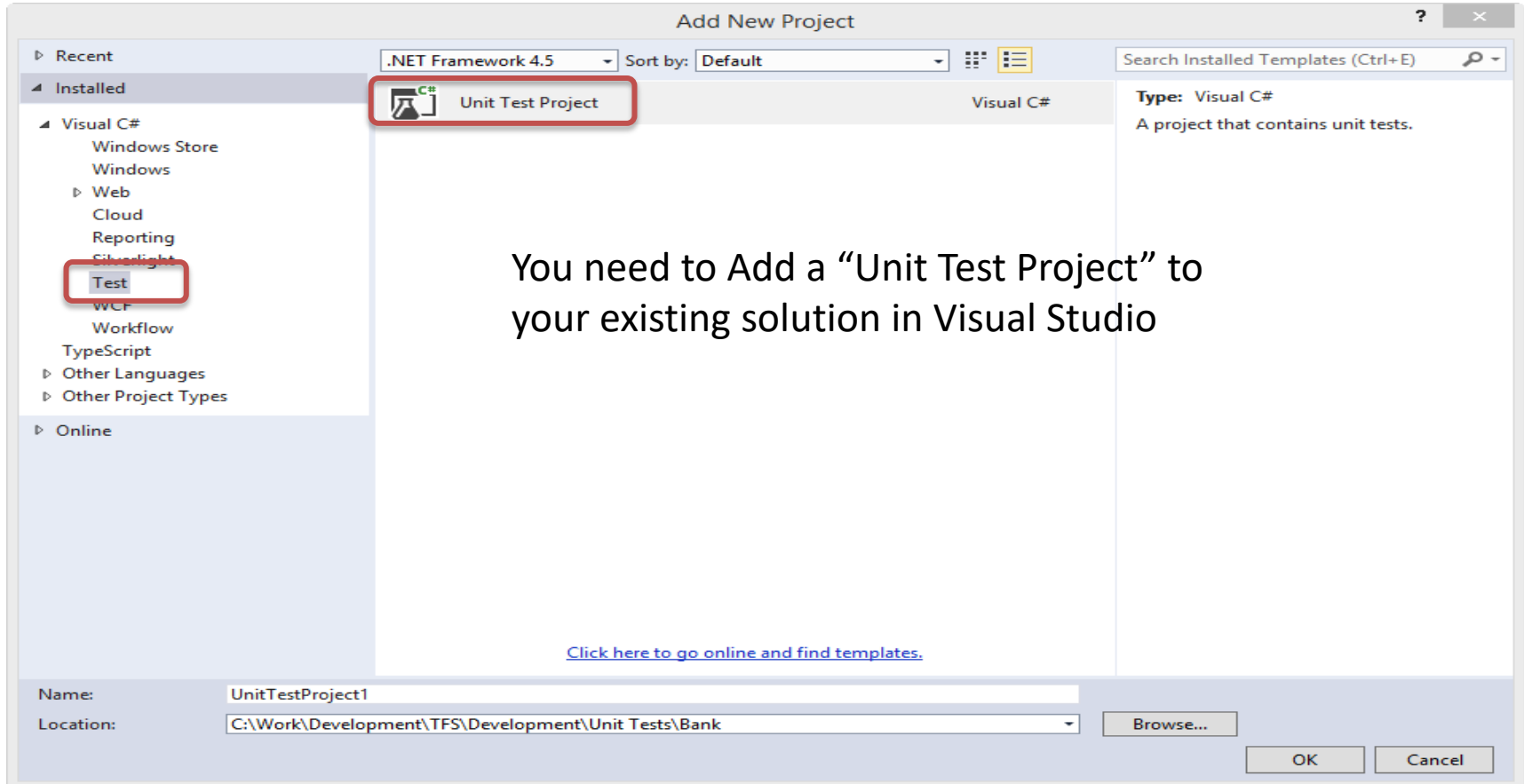
<http://www.uio.no/studier/emner/matnat/ifi/INF5530>

# Unit Testing in Visual Studio

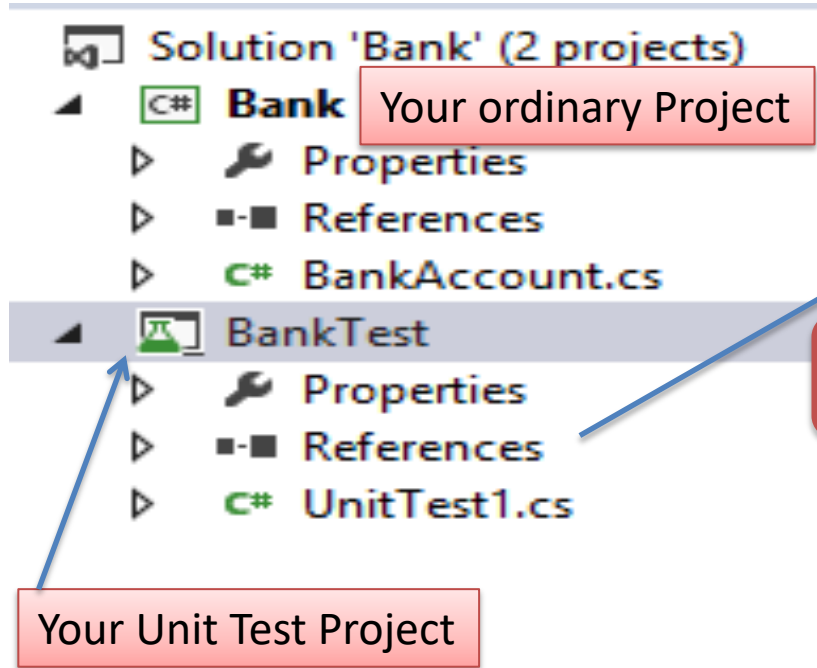
To create a unit test project:

1. On the File menu, choose **New** and then choose **Project** (Ctrl + Shift + N).
2. In the New Project dialog box, expand the Installed node, choose the language that you want to use for your test project, and then choose **Test**.
3. To use one of the Microsoft Unit Test frameworks, choose **Unit Test Project** from the list of project templates. Otherwise, choose the project template of the Unit Test Framework that you want to use.
4. In your Unit Test Project, add a reference to the code under test. Here's how to create the reference to a code project in the same solution:
  - a. Select the project in Solution Explorer.
  - b. On the Project menu, choose **Add Reference....**
  - c. In the Reference Manager dialog box, open the **Solution** node and choose **Projects**. Check the code project name and close the dialog box.

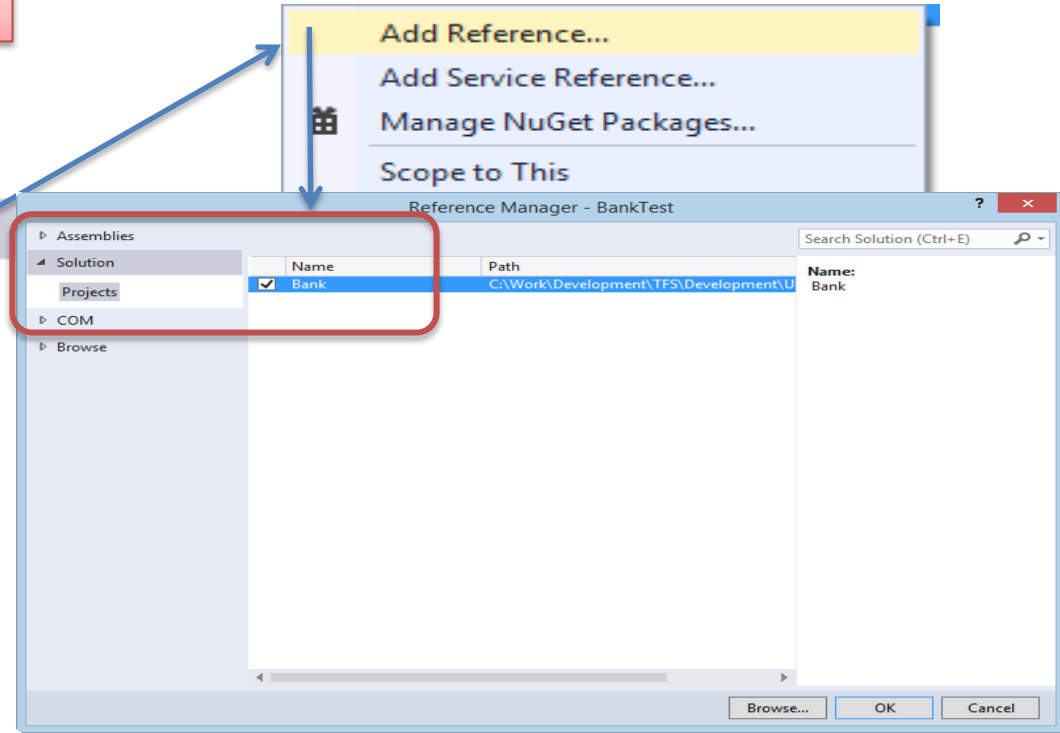
# Unit Testing in Visual Studio

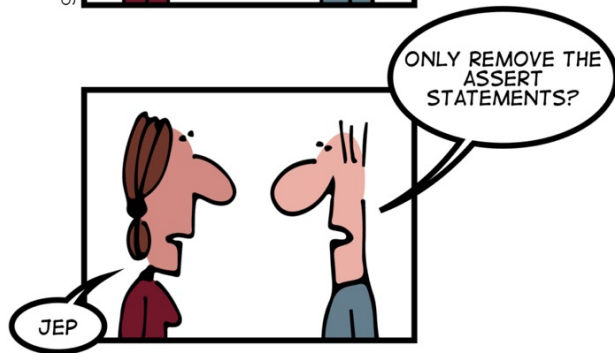
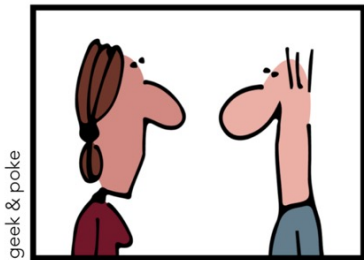


# Unit Testing in Visual Studio Example



Add Reference to the Code under Test





HOW TO SUSTAIN A DECENT  
COVERAGE



# Scrum Activities and Meetings

Hans-Petter Halvorsen

[Table of Contents](#)

# Scrum Activities and Meetings

Status: We are finished with "Beta" Iteration and should start on the next Iteration ("RC").

## Self-activities within the Teams:

1. Sprint Review Meeting ("Beta" Iteration)
2. Sprint Retrospective ("Beta" Iteration)
3. Sprint Planning ("RC" Iteration)

See Next Slides for more details...



# Scrum Activities and Meetings

The Team has done the following activities:

Team: \_\_\_\_\_

Sprint Review Meeting ("Beta" Iteration)

Short Status report: \_\_\_\_\_

---

---

Sprint Retrospective ("Beta" Iteration)

Action List:

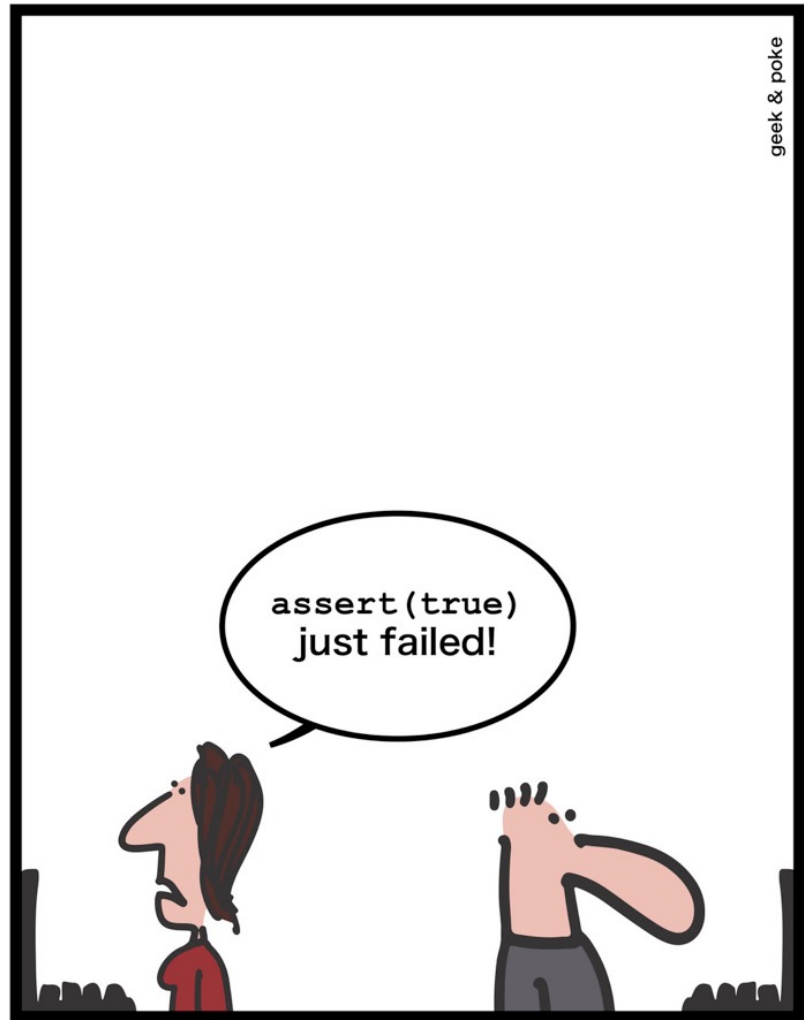
1. Keep doing: \_\_\_\_\_

2. Start doing: \_\_\_\_\_

3. Stop doing: \_\_\_\_\_

Sprint Planning ("RC" Iteration)

New Sprint Backlog has been made. # Tasks: \_\_\_\_ Est. Hours: \_\_\_\_



NEVER TAKE ANYTHING FOR GRANTED ANYMORE

# Hans-Petter Halvorsen

University of South-Eastern Norway

[www.usn.no](http://www.usn.no)

E-mail: [hans.p.halvorsen@usn.no](mailto:hans.p.halvorsen@usn.no)

Web: <https://www.halvorsen.blog>

