https://www.halvorsen.blog

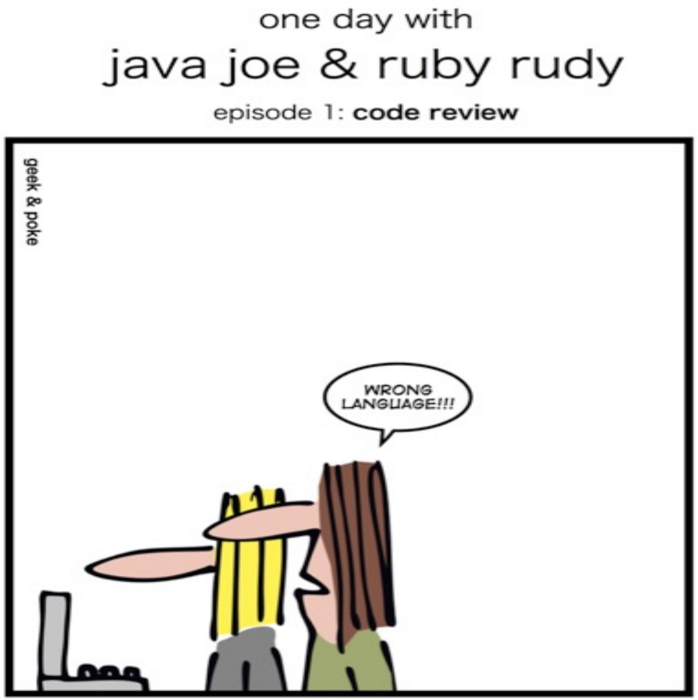
Week Assignment

Software Implementation

Hans-Petter Halvorsen

Week Assignment

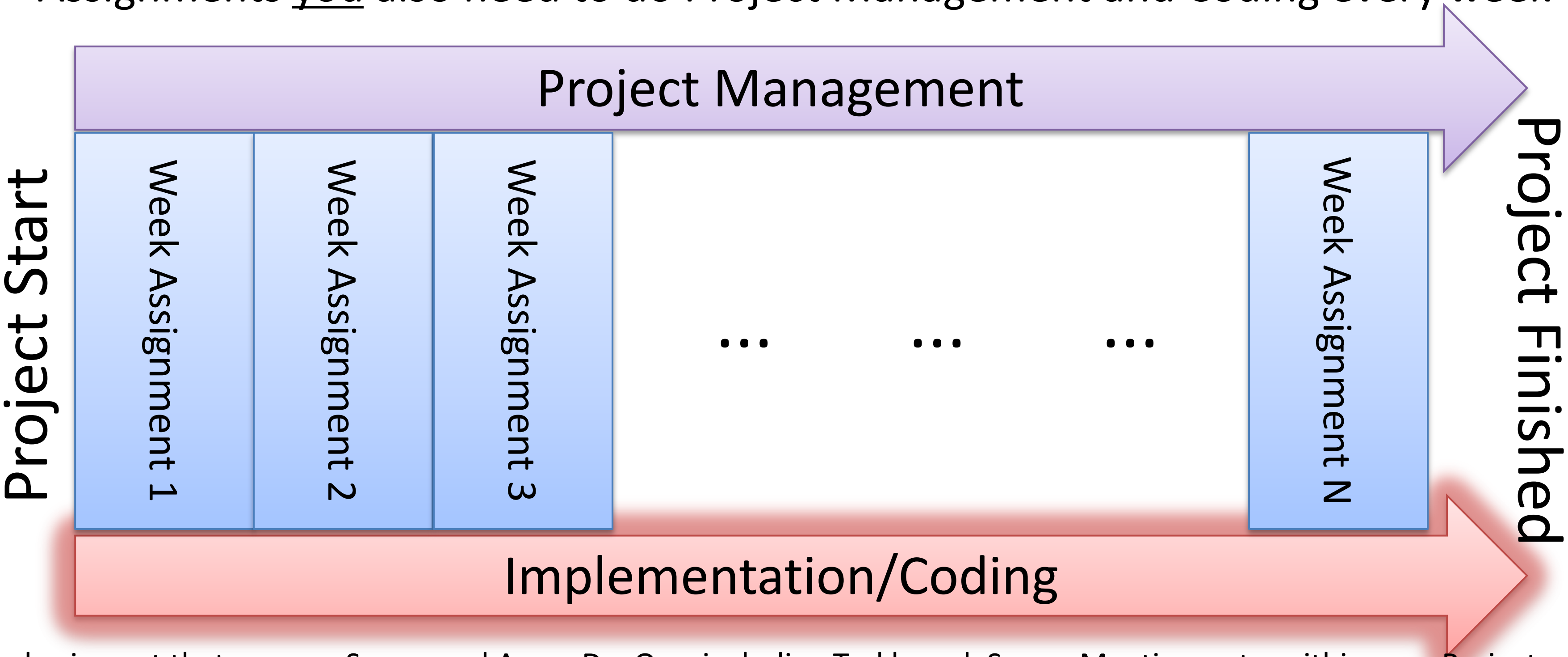
1. **Code Reviews**
   - **Code Review Checklist**
2. Refactoring
3. Pair Programming



Test out these 3 things regularly during the next weeks as part of your Project

This Course/Project

In Class we have focus on the Week Assignments – But In addition to the Week Assignments <u>you</u> also need to do Project Management and Coding every week



Project Management

Project Start

Week Assignment 1

Week Assignment 2

Week Assignment 3

… … …

Week Assignment N

Project Finished

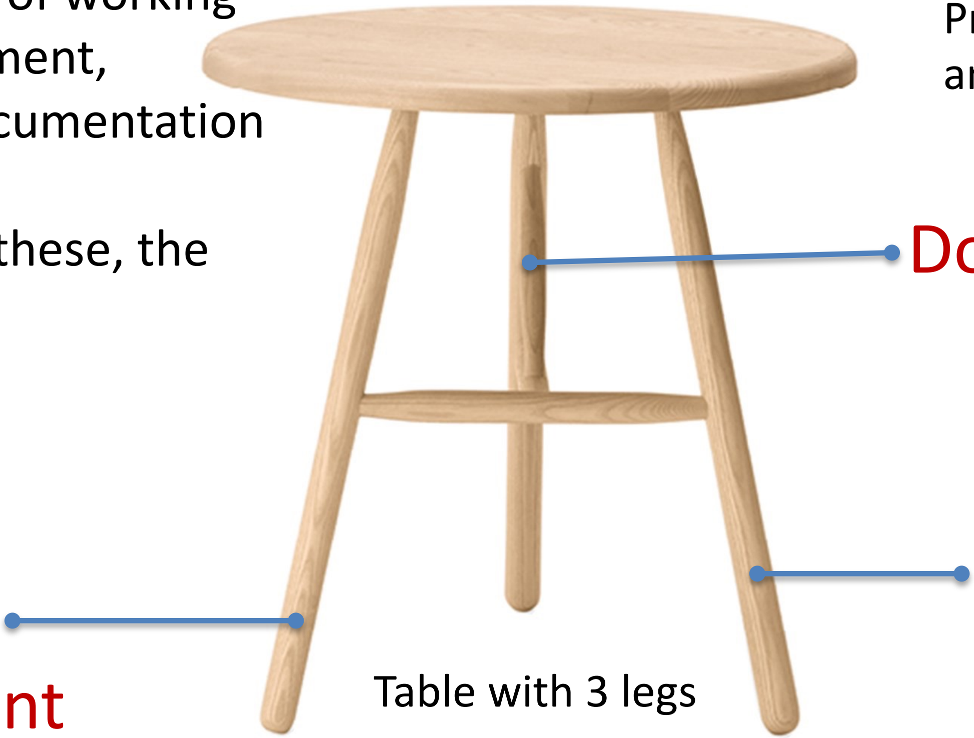
Implementation/Coding

It is also import that you use Scrum and Azure DevOps, including Taskboard, Scrum Meetings, etc. within your Project

Project Work

Project Work consists of working with Project Management, Development and Documentation in parallel.
If you remove one of these, the project will fail

So even when you are in "Implementation Mode", don't forget Project Management and Documentation



Documentation

Development

Project Management

Table with 3 legs

If you remove one of the legs, the table will fall apart

Main Focus this Week

- Continue **Implementing** your System
  - GUI
  - Classes and Methods
  - Database Logic (Stored Procedures, etc.)
  - etc.
  - Coding in general
- Use Scrum and **Azure DevOps** as your daily Tools
  - Sprint Backlog, **Taskboard**, Daily Scrum Meetings
- Apply **Agile Programming** Principles (see next slide)

Agile Programming Principles

Continuous focus on improving the code is important in Agile methods (such as Scrum, XP, etc.). Some important Agile Programming principles are:

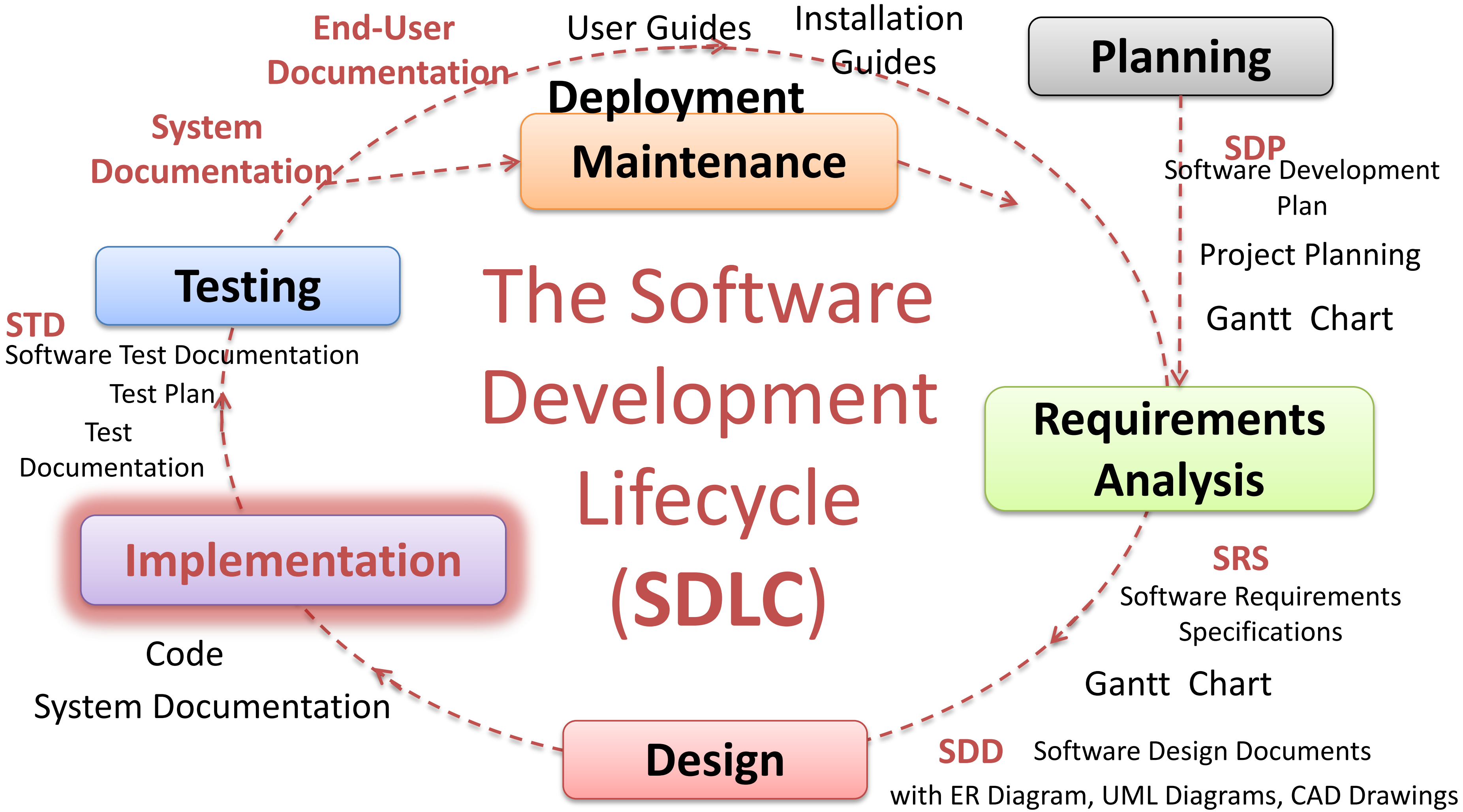
- Periodically internal **Code Reviews**
- **Refactoring**: Continuously Improving the Code
- **Pair Programming** – 2 persons sit together when they do coding. Goal: Better Code Quality. Very popular in XP.

Software Implementation

Hans-Petter Halvorsen

Table of Contents

The Software Development Lifecycle (SDLC)



Planning

SDP
Software Development Plan

Project Planning

Gantt Chart

Requirements Analysis

SRS
Software Requirements Specifications

Gantt Chart

Design

SDD Software Design Documents
with ER Diagram, UML Diagrams, CAD Drawings

Implementation

Code

System Documentation

Testing

STD
Software Test Documentation

Test Plan

Test Documentation

Deployment

Maintenance

System Documentation

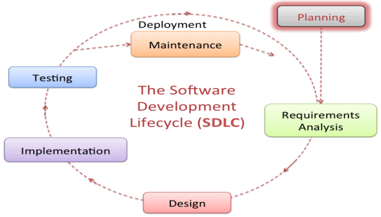
End-User Documentation

User Guides

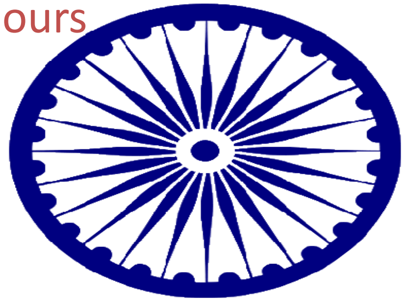
Installation Guides

Software Development

Daily Scrum Meetings



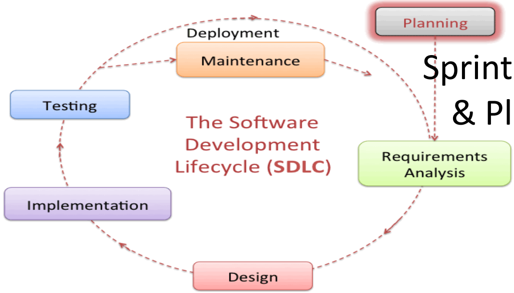
24 hours



Days

Working Software at all times.
Testing every day

Sprint Reviews & Planning

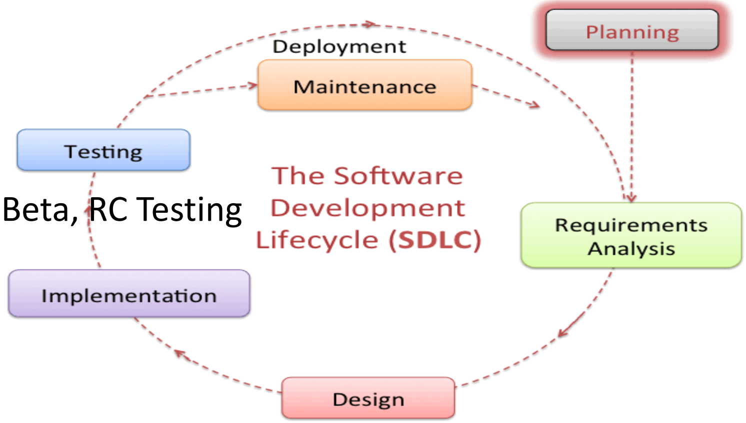


2-4 weeks

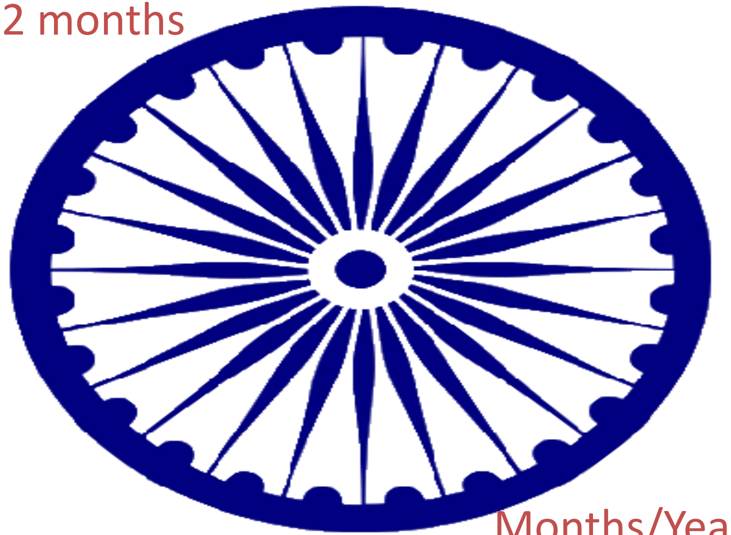
Weeks

Internal Iterations/Sprints

Beta, RC Testing



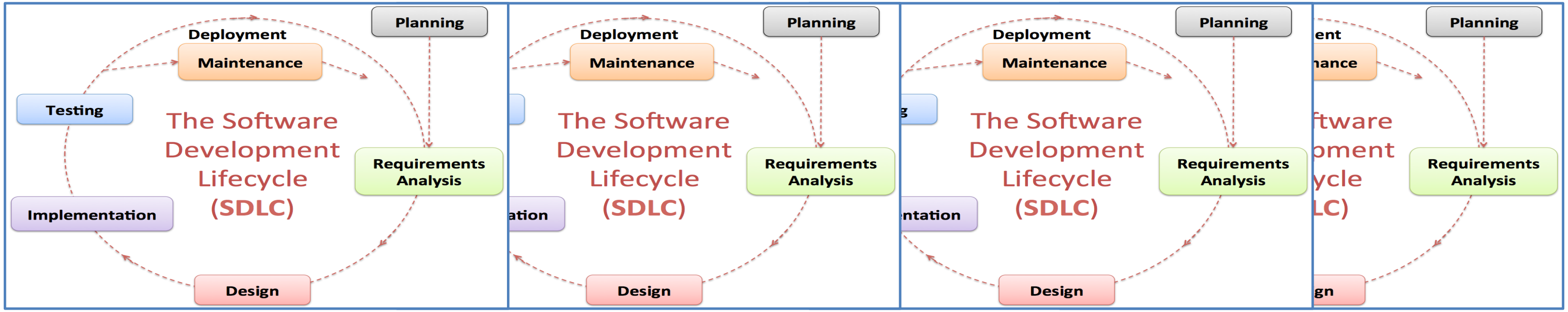
1 -12 months



Months/Years

Public Beta, RC Releases

Real life Software Development



Project Start

… … … … … … … Iterations/Sprints … … … … … …

The Software is beeing built and tested internally from beginning to the end (every day)

Alpha

Testing/Bug Fixing/
Refactoring/Redesign

Beta

Beta1

Beta 2

Beta 3

…

Eksternal Testing

Testing/Bug Fixing/
Refactoring/Redesign

RC

RC1

RC2

…

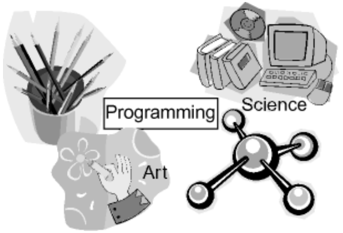
Eksternal Testing

RTM

Project Finished

What is Software Implementation?

Programming



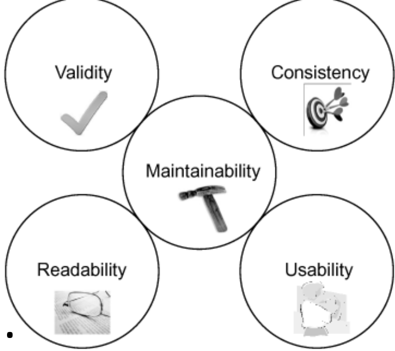
Programming can be considered as both **Art and Science**

- Programming is a **science** as it needs to follow a set of engineering principles and guidelines.
- It is also being classified as **art** as there are lots of possibilities of using creative and innovative minds.

Software Engineering (Saikat Dutt, et al.)

I would say those (Art & Science) are equally important in Programming

Programming Principles



- **Validity**: The program must give the correct result which is valid.
  - For example, let us consider a program intended to add two numbers say add (x, y). When we pass the value (4, 5), it should give the value 9 as output and when we pass the value (–4, 5), it should give the value 1 as output.
- **Consistency**: The program must do repeatedly what it intends to do. The program should give the output consistently.
  - For example, if add (4.2, 5.4) gives the output as 10, add (5.4, 4.2) also should give the output as 10.
- **Maintainability**: The program must be easily changeable (addition and modification) and should have proper documentations.
- **Readability**: The program must be easily readable so that it is easily maintainable.
- **Usability**: The program must be usable for the specific purpose without any trouble.

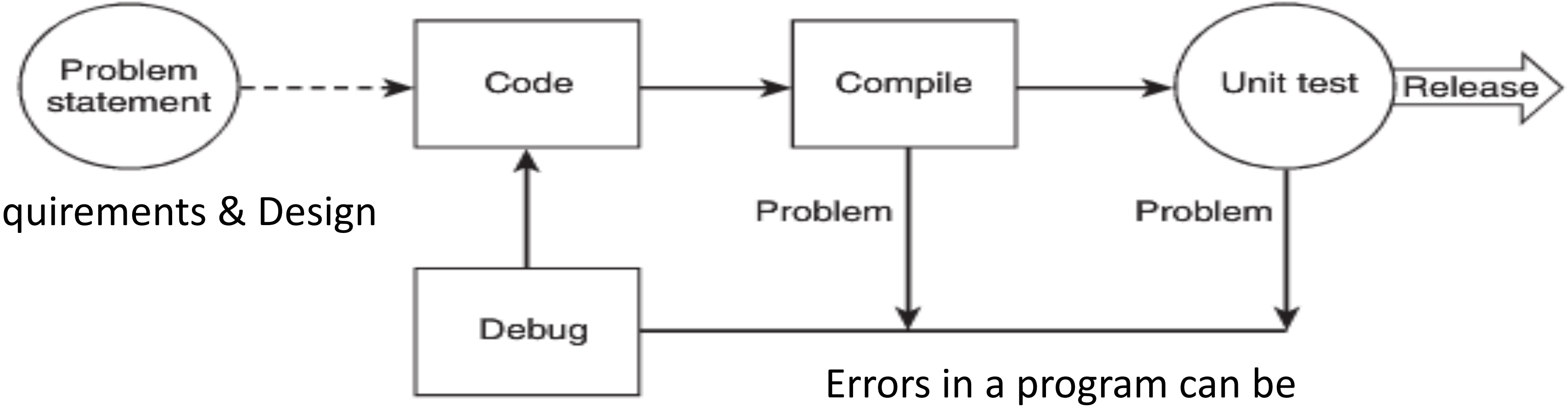
Software Engineering (Saikat Dutt, et al.)

Implementation

- The ultimate goal of most software engineering projects is to produce a working program.
- **The act of transforming the detailed design into a valid program in some programming language, together with all its supporting activities is referred to as *implementation*.**
- The implementation phase involves more than just writing code. Code also needs to be tested and debugged as well as compiled and built into a complete executable product (see next slide).
- We usually need to use a Source Code Control Tool in order to keep track of different versions of the code.

Software Implementation

Developers Perspective



Requirements & Design

Errors in a program can be broadly categorized into syntax and logic errors.

Design & Implementation

- **In many cases the detailed design is not done explicitly (in the Design Phase) but is left as part of the implementation**
- Doing the detailed design as part of the implementation is usually faster, but it may result in a less cohesive and less organized design, because the detailed design of each module will usually be done by a different person.
- **In small projects, the detailed design is usually left as part of the implementation (and in Agile/Scrum)**
- In larger projects, or when the programmers are inexperienced, the detailed design will be done by a separate person

Programming Languages

How many do you know about?

Java

Programming Languages

Perl

Kotlin

A few examples

Ruby

C/C++

C#

Visual Basic

Python

PHP

MATLAB

LabVIEW

Objective-C

Swift

Programming Languages compared with Cars:
http://crashworks.org/if_programming_languages_were_vehicles

Programming Languages

- What is a Programming Language?
- https://en.wikipedia.org/wiki/Programming_language

List of known Programming Languages:

- https://en.wikipedia.org/wiki/List_of_programming_languages

Integrated Development Environment (IDE)

IDE = Templates + Code Editor + Debugger + Compiler – Integrated in one unified Environment

- Programming Languages vs. IDE
- One IDE can handle multiple Languages

- C#, VB.NET, C++, ... → Visual Studio
- Java → Eclipse
- Objective-C/Swift → Xcode
- LabVIEW → LabVIEW

https://en.wikipedia.org/wiki/Integrated_development_environment

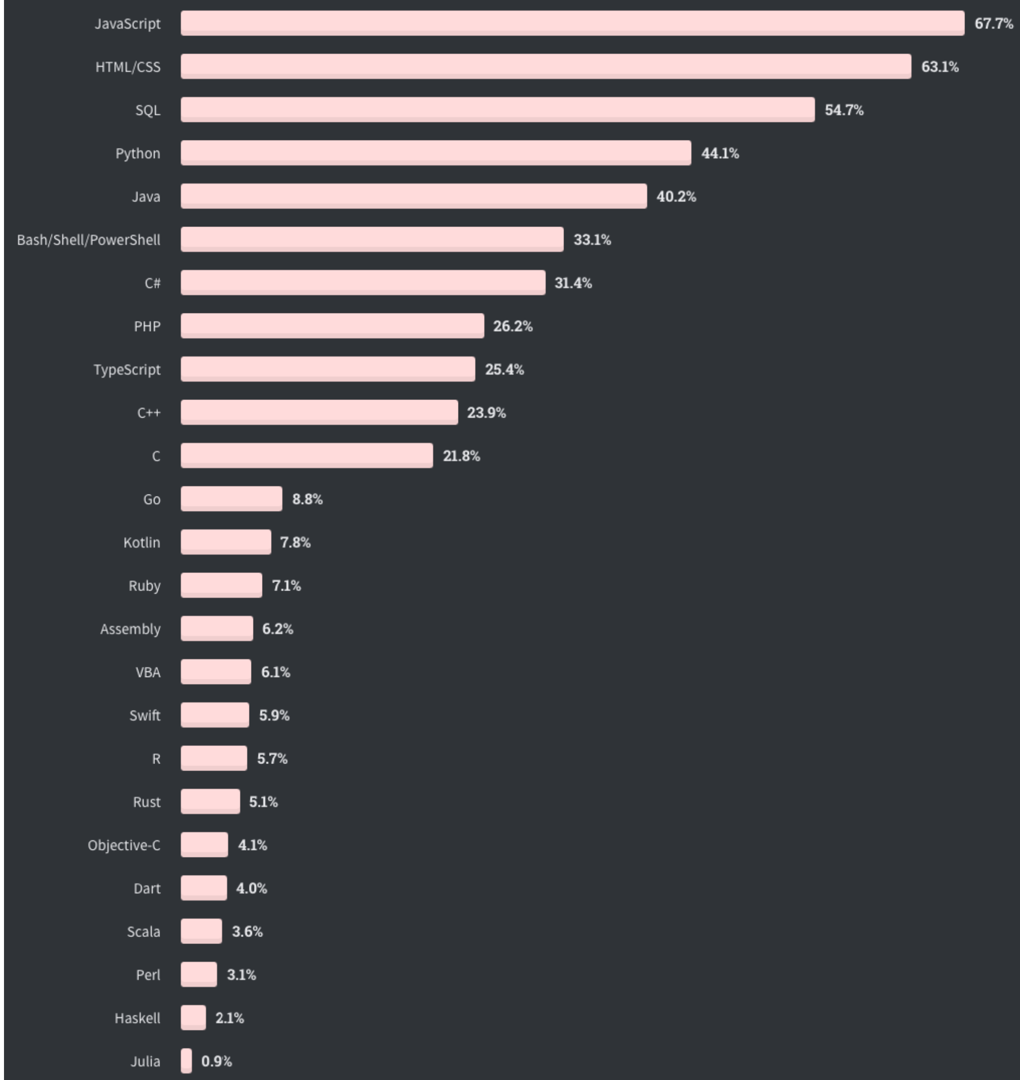
Programming Languages

Which Languages are most Popular?

Most Popular Programming Languages*

JavaScript, HTML/CSS, SQL, Python and C# are tools we are using at USN, and they are all on the top 10 list

*Stackoverflow Developer Survey 2020



Code

- Windows: 50 million code lines
- Google: 2 billion code lines

http://www.tek.no/artikler/sa-mye-kode-bestar-hele-google-av/192834

Code Review

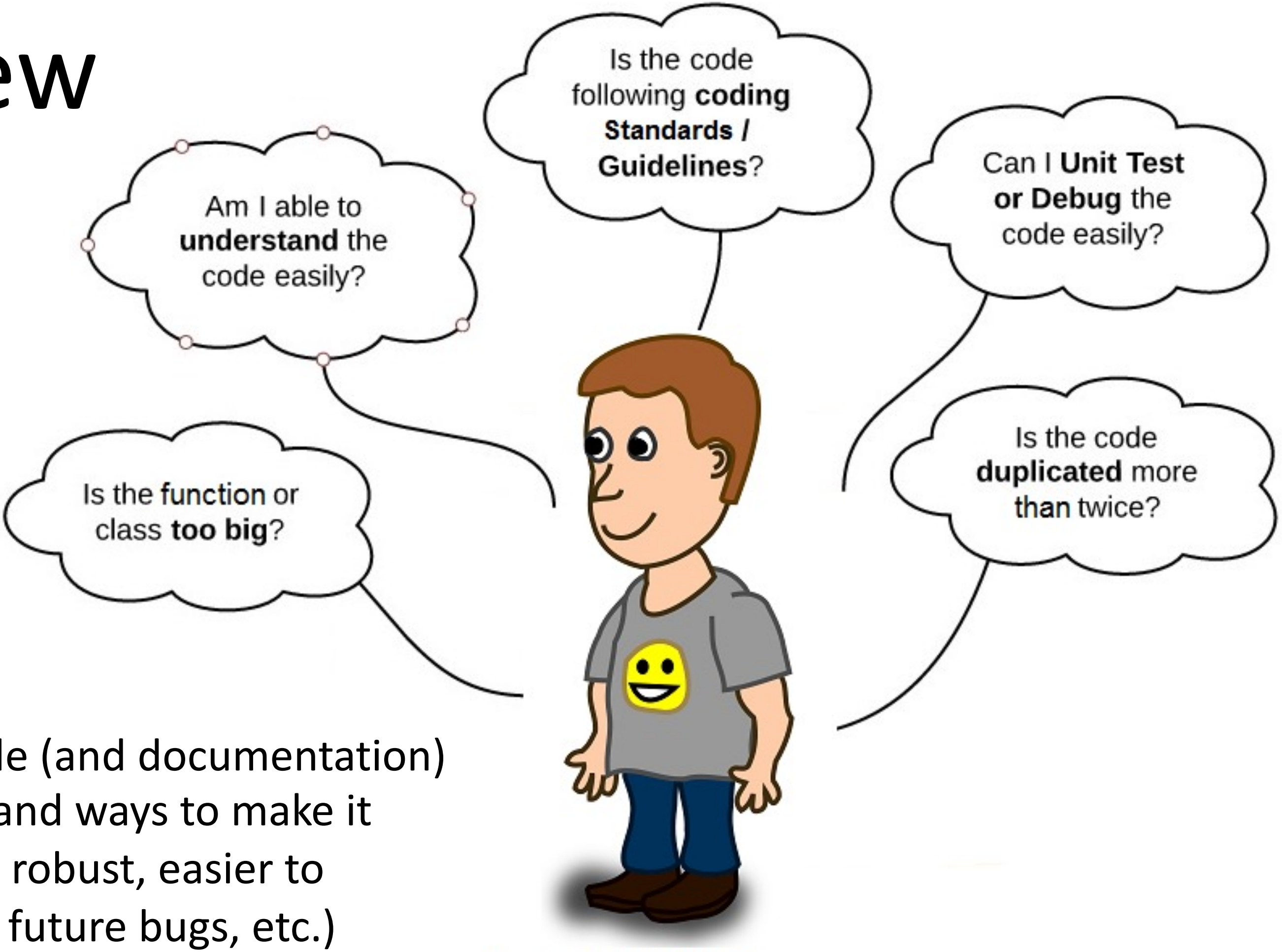
Code Review

- **Do a Code Review on the Code for one of the other Team members.** Focus in the Code Review:
  - The code is according to SRD (the code is according to the UML diagrams ,etc.)
  - Programming Style and Coding Guidelines are followed
  - Comments are used properly, etc.
  - Create a "**Code Review Checklist**" that you use during the Review
- Fill out the "Code Review Checklist". Upload to Teams/Azure DevOps.
- If you find bugs, they should be reported in Azure DevOps (Work Items)
- Go through and discuss the Code Review with the Developer

http://en.wikipedia.org/wiki/Code_review

See Next Slides for more details...

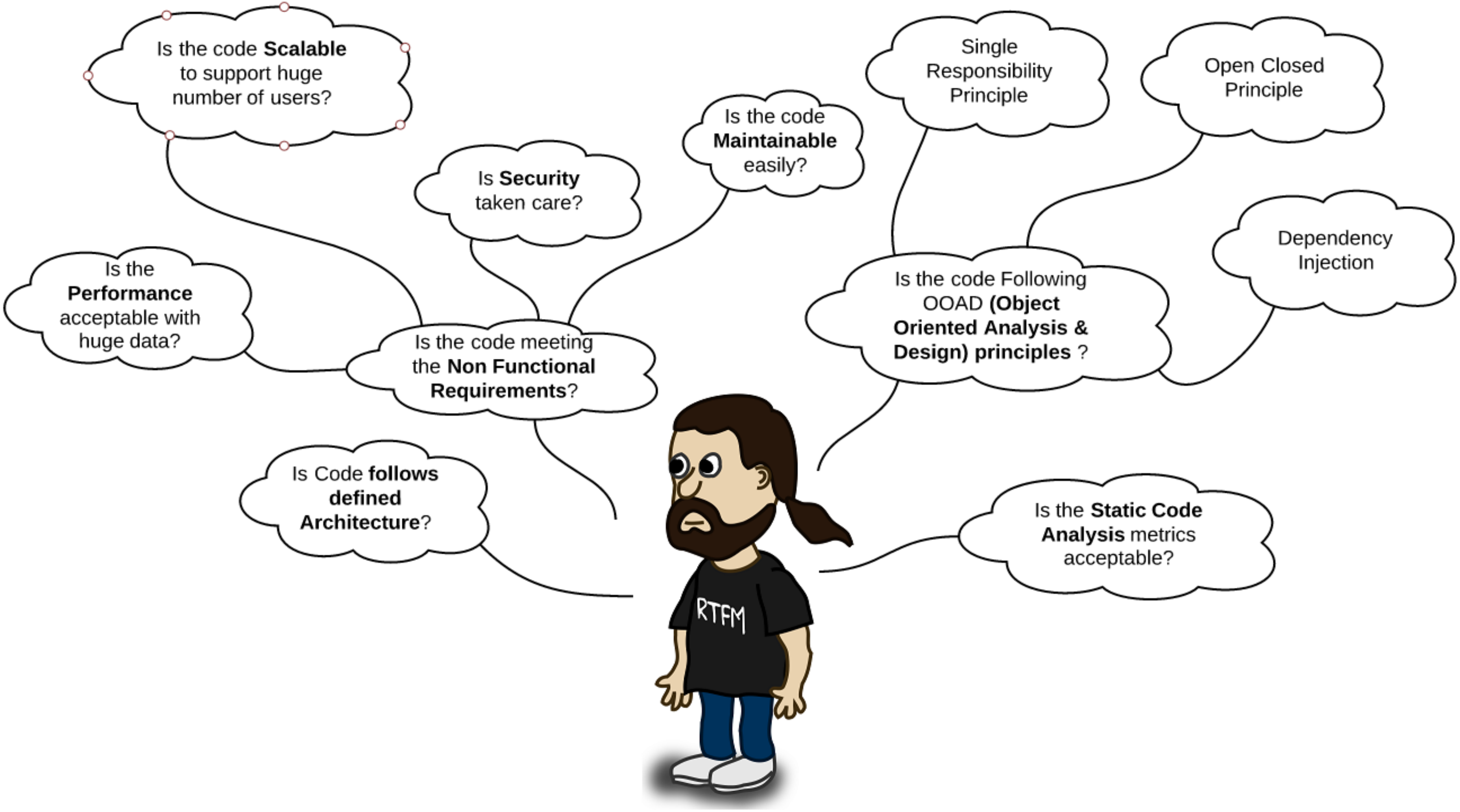
Code Review


Am I able to **understand** the code easily?
Is the code following **coding Standards / Guidelines**?
Can I **Unit Test or Debug** the code easily?
Is the function or class **too big**?
Is the code **duplicated** more than twice?

Basic Code Reviewer

<u>Purpose</u>: Go through the code (and documentation) in detail to find weaknesses and ways to make it better (more readable, more robust, easier to maintain in the future, avoid future bugs, etc.)

http://www.evoketechnologies.com/blog/code-review-checklist-perform-effective-code-reviews/



Is the code **Scalable** to support huge number of users?

Is the **Performance** acceptable with huge data?

Is **Security** taken care?

Is the code **Maintainable** easily?

Is the code meeting the **Non Functional Requirements**?

Is Code **follows defined Architecture**?

Single Responsibility Principle

Open Closed Principle

Dependency Injection

Is the code Following OOAD **(Object Oriented Analysis & Design) principles** ?

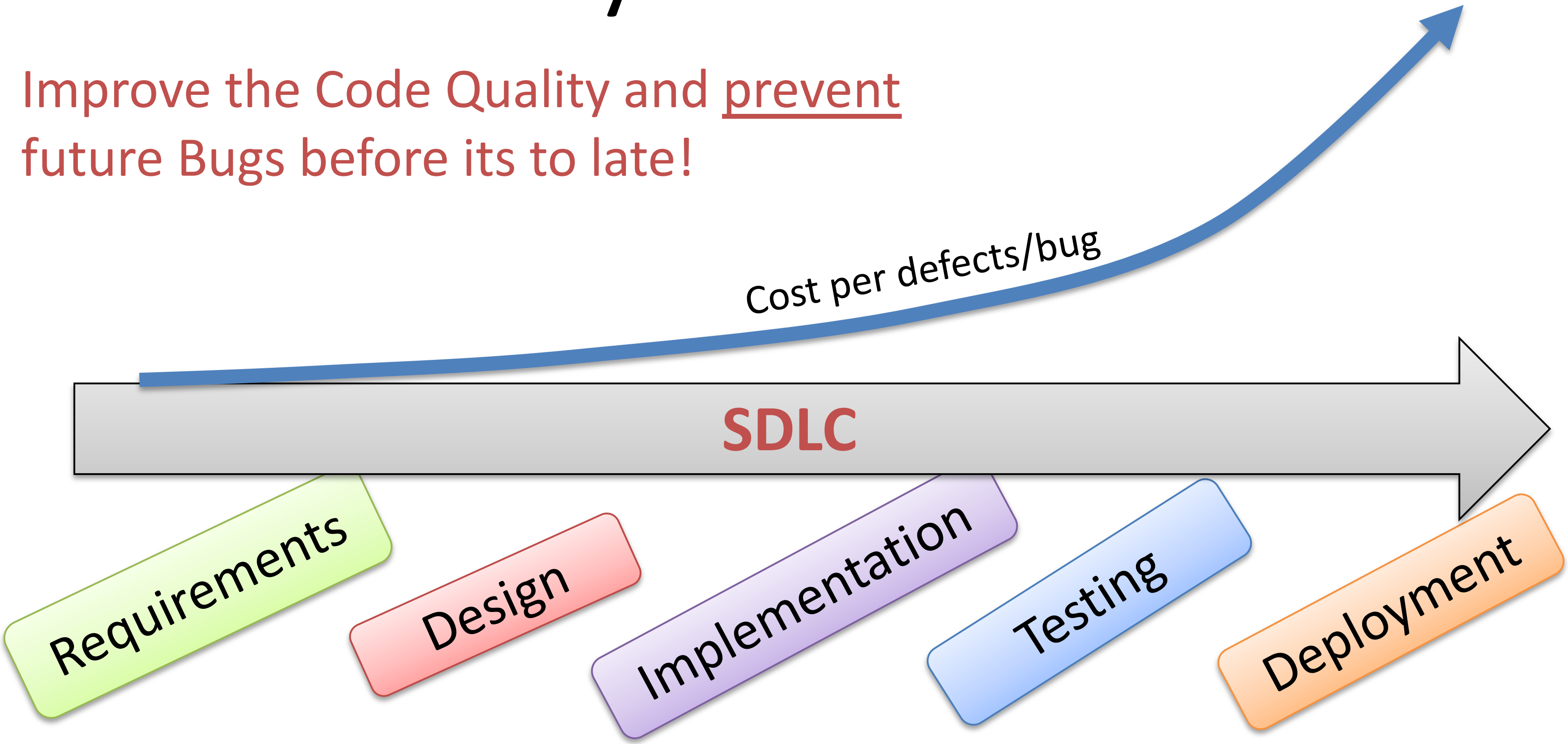
Is the **Static Code Analysis** metrics acceptable?

Expert Code Reviewer

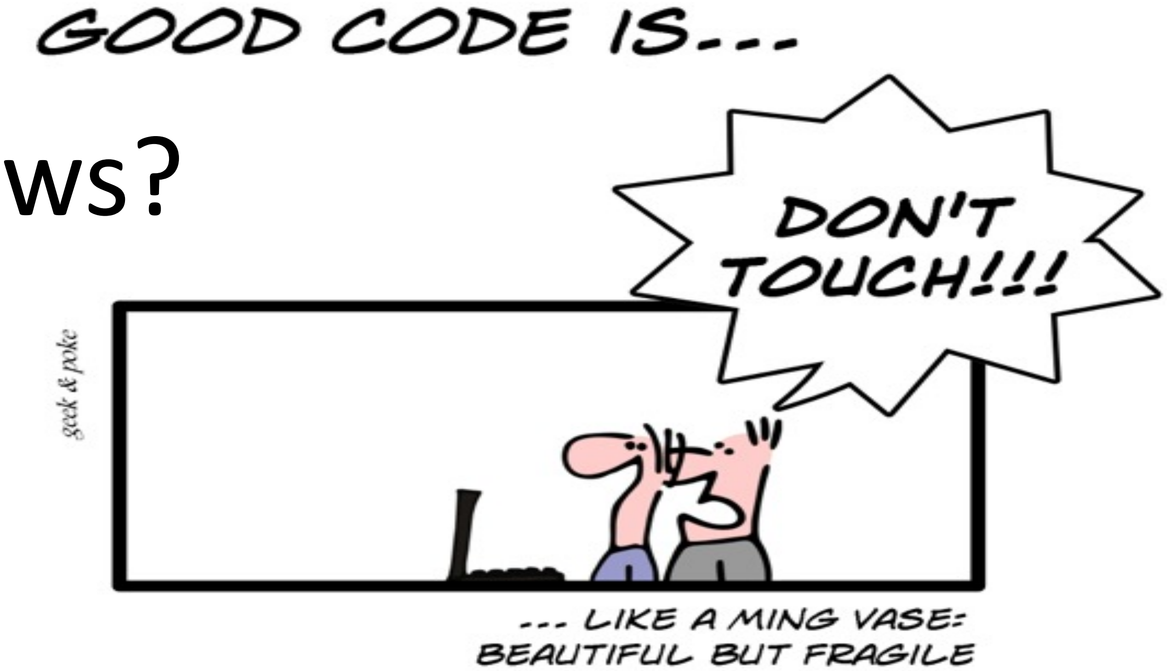
http://www.evoketechnologies.com/blog/code-review-checklist-perform-effective-code-reviews/

Why Do Reviews?

Improve the Code Quality and <u>prevent</u> future Bugs before its to late!



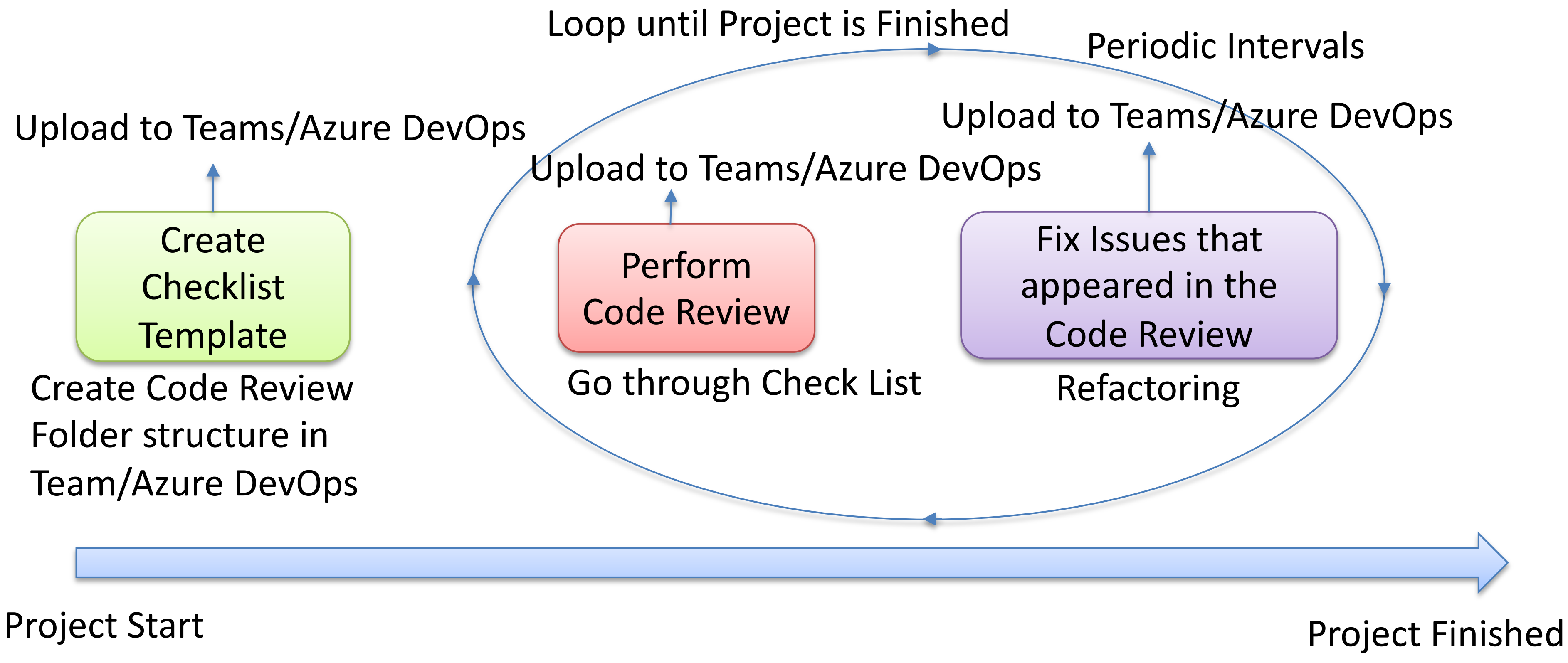
Why Do Reviews?



"If your code works, but you don't know why

– Then it does not work, you just don't know it yet"

Code Review Procedure



Loop until Project is Finished

Periodic Intervals

Upload to Teams/Azure DevOps

Create Checklist Template

Create Code Review Folder structure in Team/Azure DevOps

Upload to Teams/Azure DevOps

Perform Code Review

Go through Check List

Upload to Teams/Azure DevOps

Fix Issues that appeared in the Code Review

Refactoring

Project Start

Project Finished

Code Review Checklist Template Example

Use e.g., Excel to create a Code Review Checklist Template (e.g., with different Tabs for different Topics).

Date of Code Review: <Fill out>
Code Owner: <Fill out>
Code Reviewer: <Fill out>

Make the Code review Template available for everybody by the uploading it to Teams/Azure DevOps

Status	Description	Comments	Priority
	Are all variables properly defined with meaningful, consistent, and clear names?		
	Does the code completely and correctly implement the design?		
	Are all comments consistent with the code?		
	Are there any blocks of repeated code that could be condensed into a single Method?		
	... (if you find stuff that are not listed, just fill out more rows in the Excel sheet)		

+++ (you should have 20-50 Items in the Checklist)

Completed Code Review Checklist Example

Date of Code Review:	2016.02.25
Code Owner:	Nils Hansen
Code Reviewer:	Per Jensen

After the Code Review Checklist has been filled out by the Code Reviewer, the List should be uploaded to Teams/Azure DevOps

Fields filled out by the Code Reviewer

Status	Description	Comments	Priority
✅	Are all variables properly defined with meaningful, consistent, and clear names?	OK	
❌	Does the code completely and correctly implement the design?	No. GUI not according to SRD. Class “Customer” not according to UML. See SRD document.	2
✅	Are all comments consistent with the code?	OK	
❌	Are there any blocks of repeated code that could be condensed into a single Method?	See Code in Form1.aspx.cs, line 205-300	1
…	… (if you find stuff that are not listed, just fill out more rows in the Excel sheet)	…	

Implementation/Coding

- Programming Style and Coding Guidelines
- Comments
- Debugging
- Code Review
- Refactorization

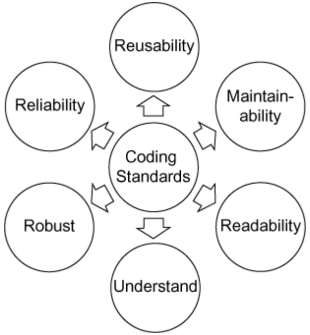
Coding Conventions/Programming Practices

- Naming Convention
  - File Names, Folder Names, Class and Methods Names, Uppercase/Lowercase, etc.
- Programming Principles and Best Practice
  - Variables, Database Communications, Public/Private/Global/Local, Comments, Code Structure
- Guidelines
  - Good developers always follow the coding standards and guidelines while writing the code.
  - Code written using the standards and guidelines are easy to review/understand/debug.
  - It is also easy to maintain and enhance the code if it follows the standards and guidelines.

Software Engineering (Saikat Dutt, et al.)

Coding Standards

- Reusability (easy to reuse parts of code as it is written in standard code)
- Maintainability (easy to identify bugs, easy to add new features)
- Readability (coding standard increases easy reading)
- Understand ability (easy to understand – this is not the same as readability, e.g., repetitive code may be is readable, but not understandable)
- Robustness (code that can handle unexpected inputs and conditions)
- Reliability (code i.e., unlikely to produce wrong results)



Software Engineering (Saikat Dutt, et al.)

Naming Convention

- **Camel Notation**
  - For variables and parameters/arguments
  - Examples: “myCar”, “backColor”
- **Pascal Notation**
  - For classes, methods and properties
  - Examples: “ShowCarColor”
- **Hungarian Notation**
  - For controls on your user interface we either use “Pascal notation” or “Hungarian notation”, but stick to one of them!
  - Examples: “txtName”, “lblName”
- Acronyms
  - Examples: “DBRate”, “ioChannel”, “XmlWriter”, “htmlReader”

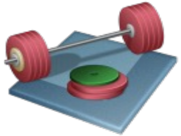
Code Review Checklist and Guidelines for C# Developers

- □ **Naming conventions** to be followed always. Generally for variables/parameters, follow Camel casing and for method names and class names, follow Pascal casing.
- □ **Code Reusability**: Extract a method if the same piece of code is being used more than once or you expect it to be used in future.
- □ Make sure that there shouldn't be any **project warnings**.
- □ **Code Consistency**. Use same type of variables, etc.
- □ **Code Readability**: Should be maintained so that other developers understand your code easily.
- □ **Methods**: Make sure that methods have less number of lines of code. Not more than 30 to 40 lines.
- □ **Unit Testing**. Write developer test cases and perform unit testing to make sure that basic level of testing is done before it goes to QA testing.
- □ **Avoid nested for/foreach loops** and nested if conditions as much as possible.
- □ Understand thoroughly the **OOPs concepts** and try implementing it in your code.
- □ Avoid straightaway ***copy/pasting of code*** from other sources. It is always recommended to hand write the code even though if you are referring to the code from some sources.
- □ **Peer code reviews**. Swap your code files/pages with your colleagues to perform internal code reviews.

Google "Code Review Checklist" and you will find lots of Examples!

http://www.codeproject.com/Articles/593751/Code-Review-Checklist-and-Guidelines-for-Csharp-De

Software Development Plan



- As mention earlier, many of these things should be included in the Software Development Plan (SDP)
- Make sure to update the SDP if you lack information about Coding Conventions, Programming Practices, Code Reviews, etc.
- Information where you find the Code Review Check List(s) and where to put the actual Reviews should be part od SDP

Refactoring

Refactoring

- Based on the Code Review, you should Refactoring (Improve) your Code

See Next Slides for more details...

Refaktorering (omstrukturering)

- Se etter forbedringsmuligheter og implementer dem selv om det ikke er umiddelbart behov for dem
- Koden blir mer forståelig og enklere å endre, og mindre behov for dokumentasjon (mer vedlikeholdbar)
- Noen endringer krever at arkitekturen omstruktureres (kostbart)
- Eksempler på refaktorering:
  - Reorganisering av klassehierarki for å fjerne duplisert kode
  - Lag og ta i bruk feks et Klassebibliotek for kode som deles av flere moduler
  - Forbedre navn på attributter og metoder
  - Erstatte kode med kall til metoder i et programbibliotek

Refactoring - Symptoms

- **Coding Style and Name Conventions** not followed
- Proper **Commenting** not followed
- **Duplicated code** (clearly a waste).
- Long **method** (excessively large or long methods perhaps should be subdivided into more cohesive ones).
- Large **class** (same problem as long method).
- Switch statements (in object-oriented code, switch statements can in most cases be replaced with polymorphism, making the code clearer).
- Feature envy, in which a method tends to use more of an object from a class different to the one it belongs.
- Inappropriate intimacy, in which a class refers too much to private parts of other classes.

=> Any of these symptoms (and more) will indicate that your code can be improved. You can use refactoring to help you deal with these problems.

Essentials of Software Engineering: Frank Tsui; Orlando Karam; Barbara Bernal, 3 ed., Jones & Bartlett Learning

Update Code Review Checklist

Date of Code Review: 2016.02.25
Code Owner: Nils Hansen
Code Reviewer: Per Jensen
Refactoring Date: 2016.03.10

Change Status

Fields filled out/Changed by the Code Owner

Status	Description	Comments	Priority	Refactoring – What have been done to improve it?
✓	Are all variables properly defined with meaningful, consistent, and clear names?	OK		
✗	Does the code completely and correctly implement the design?	No. GUI not according to SRD. Class “Customer” not according to UML. See SRD document.	2	Describe what you have done
✓	Are all comments consistent with the code?	OK		
✗	Are there any blocks of repeated code that could be condensed into a single Method?	See Code in Form1.aspx.cs, line 205-300	1	Describe what you have done
…	… (if you find stuff that are not listed, just fill out more rows in the Excel sheet)	…		

Pair Programming

Pair Programming

- Pair Programming is used in Agile Development and especially eXtreme Programming (XP)
- **Work together (for a short period) 2 and 2 and test out Pair Programming**
- What do you think of this method? Pros and Cons? – Make the Pros/Cons List together (PowerPoint with 2-3 slides).

See Next Slides for more details...

Parprogrammering

- To programmerere utvikler kode sammen:
- **– Fører:**
  - kriver på tastaturet
- **– Navigatør**
  - observerer arbeidet til føreren og ser etter feil og svakheter
  - ser etter alternativer
  - noterer ting som må gjøres
  - slår opp referanser
- Kan brukes uavhengig av smidige metoder

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog