https://www.halvorsen.blog

Software Engineering

Week Assignment

Project Kick-off & Planning

Hans-Petter Halvorsen

Week Assignment

1. Project Start: Define Teams & Roles
   - Select Project
   - Make CV (Overview of Skills, Education, Work, etc.)
2. Team **Brainstorming** (What/How?)
3. Create a **Software Development Plan** (SDP)
4. Development **Tools**
   - Install necessary Software
   - Get Started with **Azure DevOps**

Course Introduction

Hans-Petter Halvorsen

Table of Contents

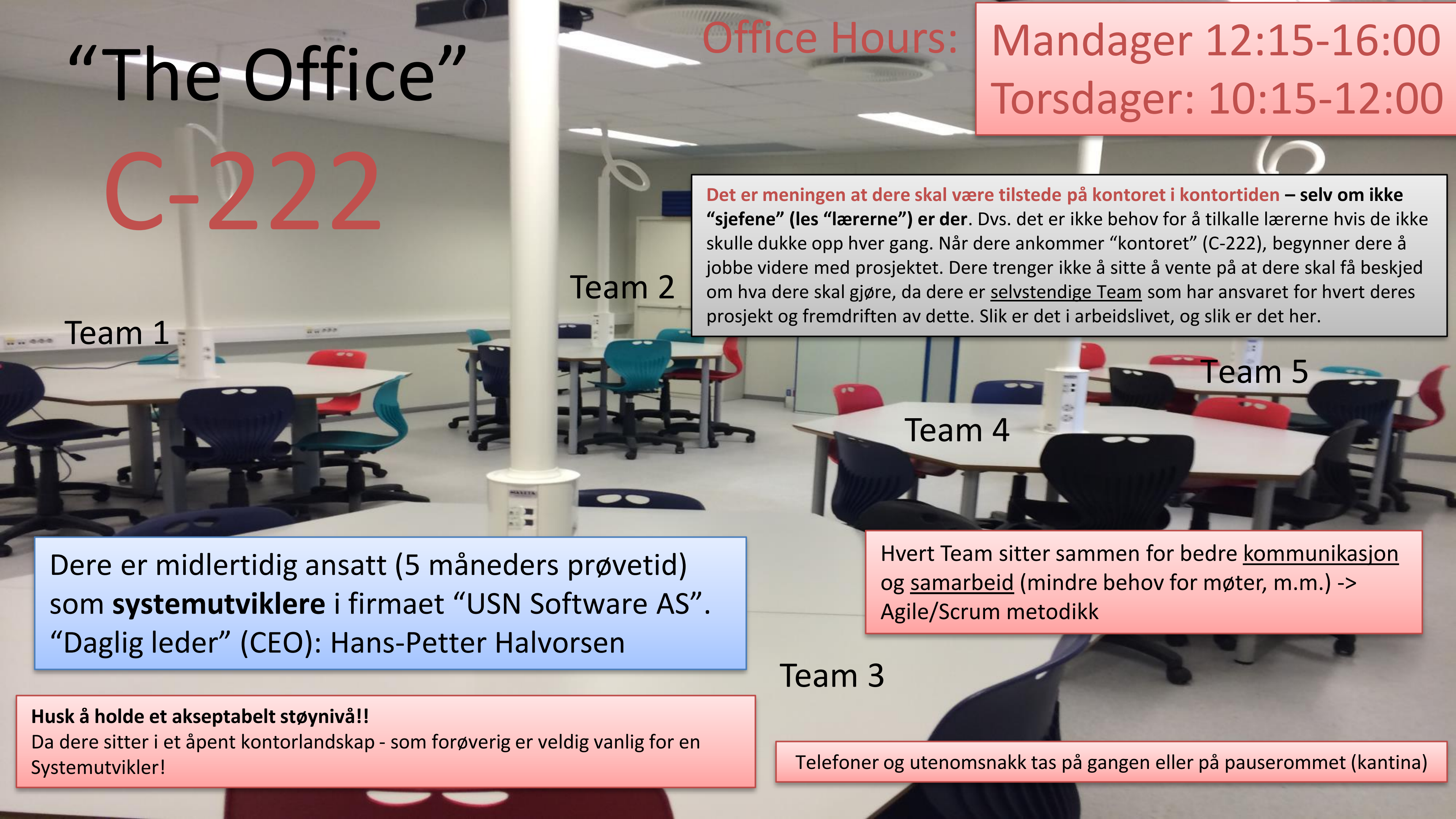
"USN Software"

- Dere har herved fått midlertidig ansettelse (5 måneders prøvetid) i firmaet "USN Software"
- Dere er ikke lenger studenter men <u>Systemutviklere</u>
- Dere "må" møte på jobb i arbeidstiden ("kjernetiden") som er Mandager 12:15-16:00 og Torsdager 10:15-12:00. Jobbing utover dette må påregnes enten på jobb eller hjemmekontor
- Arbeidsted er C-222 ("kontoret") - Dvs åpent kontorlandskap som jo blir mer og mer vanlig
- Prøv å endre "mindset" og "uvaner" dere har tilegnet dere som studenter
  - Møt forberedt på jobb. Møt i tide.
  - Man begynner ikke arbeidsdagen med 4 timers "forelesning" i industrien/næringslivet
  - I industrien/næringslivet planlegger man hva man skal gjøre dagen før/uka før, man planlegger møter i god tid, man stiller forberedt på møter, osv. Man bruker ofte litt av kvelden i forveien til å planlegge/forberede morgendagen
  - Overhold tidsfrister (dere lever av kunden). Det blir stilt krav til dere. Kvalitetssikring av arbeidet.
  - Kan man ikke komme på jobb (syk eller liknende), så si/meld fra til teamet/nærmeste leder
  - Ta ansvar for prosjektet og egen læring samt dine medarbeidere, dere er ikke bedre enn det "svakeste ledd"
  - Eksempel: <u>Ikke</u> skriv "Vi er 3 studenter på USN som har fått i oppdrag å lage ..."

"The Office"

C-222

Office Hours: Mandager 12:15-16:00
Torsdager: 10:15-12:00



Det er meningen at dere skal være tilstede på kontoret i kontortiden – selv om ikke "sjefene" (les "lærerne") er der. Dvs. det er ikke behov for å tilkalle lærerne hvis de ikke skulle dukke opp hver gang. Når dere ankommer "kontoret" (C-222), begynner dere å jobbe videre med prosjektet. Dere trenger ikke å sitte å vente på at dere skal få beskjed om hva dere skal gjøre, da dere er <u>selvstendige Team</u> som har ansvaret for hvert deres prosjekt og fremdriften av dette. Slik er det i arbeidslivet, og slik er det her.

Dere er midlertidig ansatt (5 måneders prøvetid) som **systemutviklere** i firmaet "USN Software AS". "Daglig leder" (CEO): Hans-Petter Halvorsen

Hvert Team sitter sammen for bedre <u>kommunikasjon</u> og <u>samarbeid</u> (mindre behov for møter, m.m.) -> Agile/Scrum metodikk

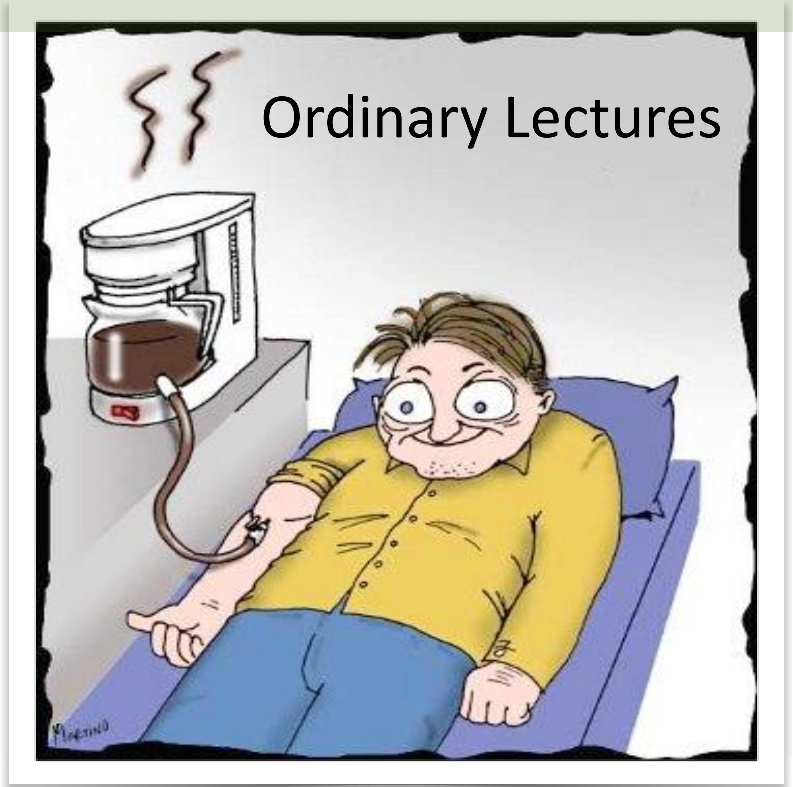
Husk å holde et akseptabelt støynivå!!
Da dere sitter i et åpent kontorlandskap - som forøverig er veldig vanlig for en Systemutvikler!

Telefoner og utenomsnakk tas på gangen eller på pauserommet (kantina)

Do you learn like this?

In this Course we apply:

- Authentic Teaching and Learning
- Flipped Classroom
- Project and Problem-based Learning (PPBL)
- Student-centric Teaching
- Education 3.0
- Programming based learning



Hard Work and doing Practical Work and Exercises are probably better ☺

Learning Process

$\sim\frac{1}{3}$

Preparations

Watch **Videos,** read **Textbooks, install Software**, etc.
(go through resources on web page for the week)

$\sim\frac{1}{3}$

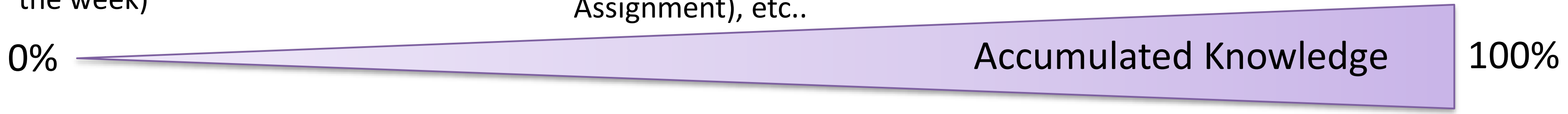
In Class

Reviews (Review of previous Week Assignment), **Project Work** (work with Week Assignment), etc..

$\sim\frac{1}{3}$

Finishing Work

Videos, Textboooks, Quizzes, Additional Resources, **Project Work (<u>complete Week Assignment</u>)**, etc.



Most of the learning takes place outside the classroom: Do not forget the benefits of being well prepared when you come to the classroom, as well as working with the material afterwards.
Learning takes place during preparations, and when you processes the material. The students need to work active with the topics and resources.

Teaching Outcome

“Traditional Lectures are passive teaching methods, where you can keep your attention for 10 minutes” Alf Inge Wang, Professor NTNU (The vendor of Kahoot)

Lectures – 5%

Reading– 10%

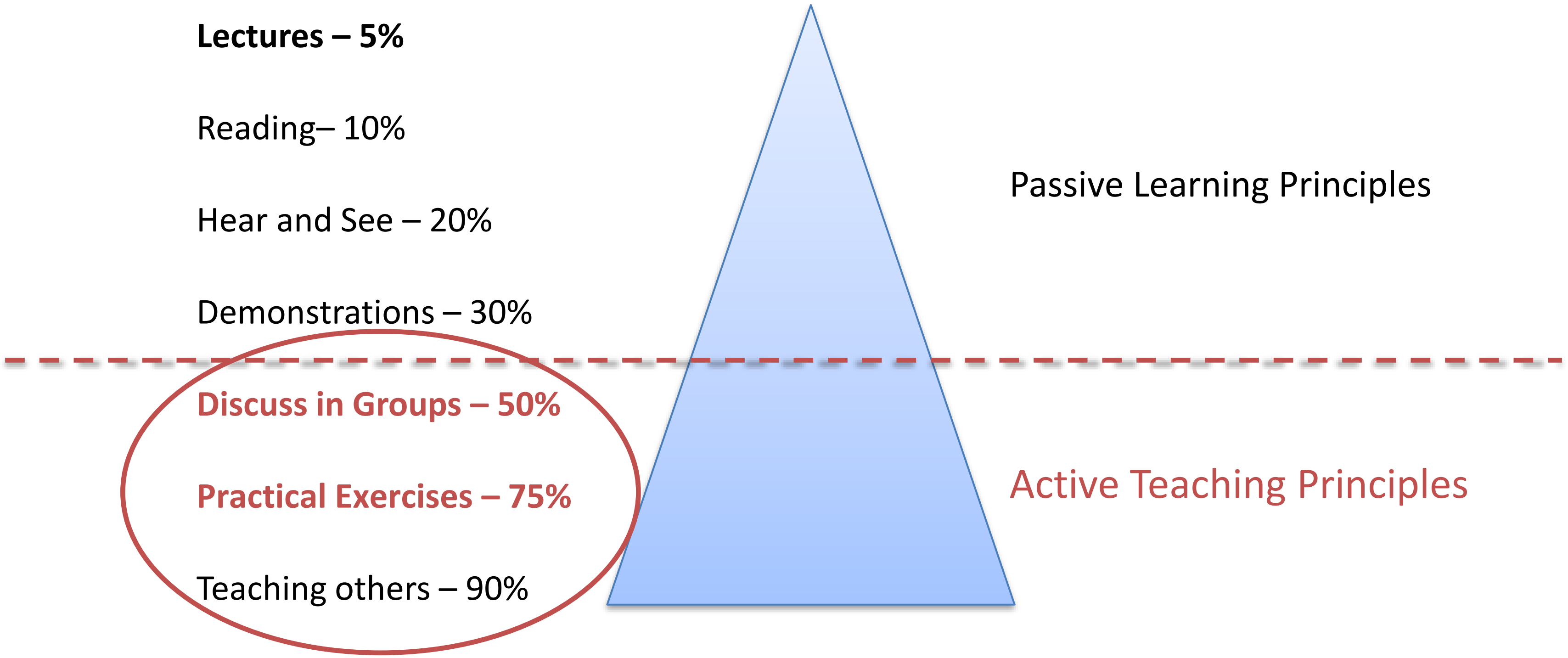
Hear and See – 20%

Demonstrations – 30%

Discuss in Groups – 50%

Practical Exercises – 75%

Teaching others – 90%



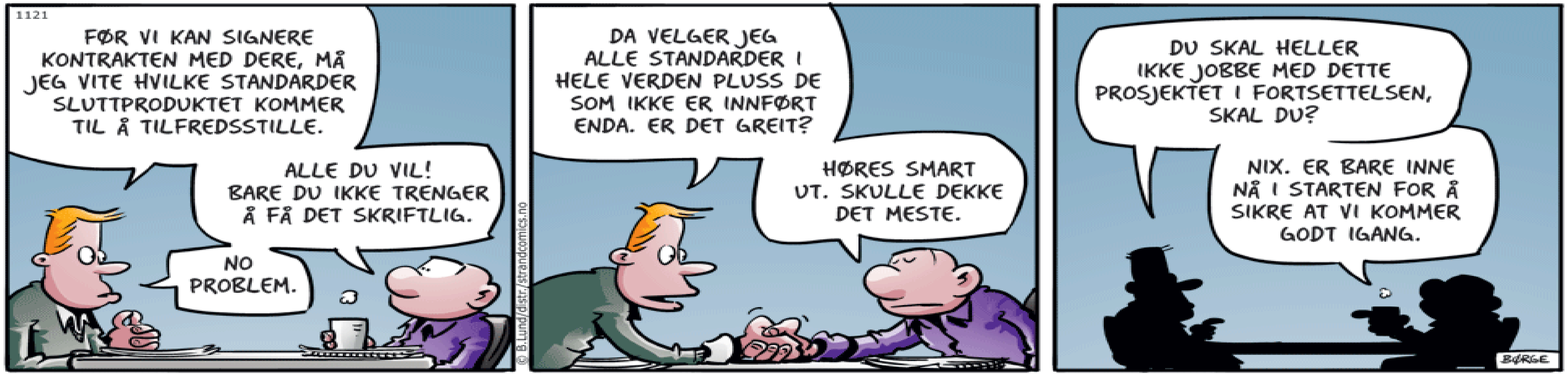
Always be prepared for Class!

(Just to be clear: This is NOT to be prepared!)

Project Kick-off & Planning

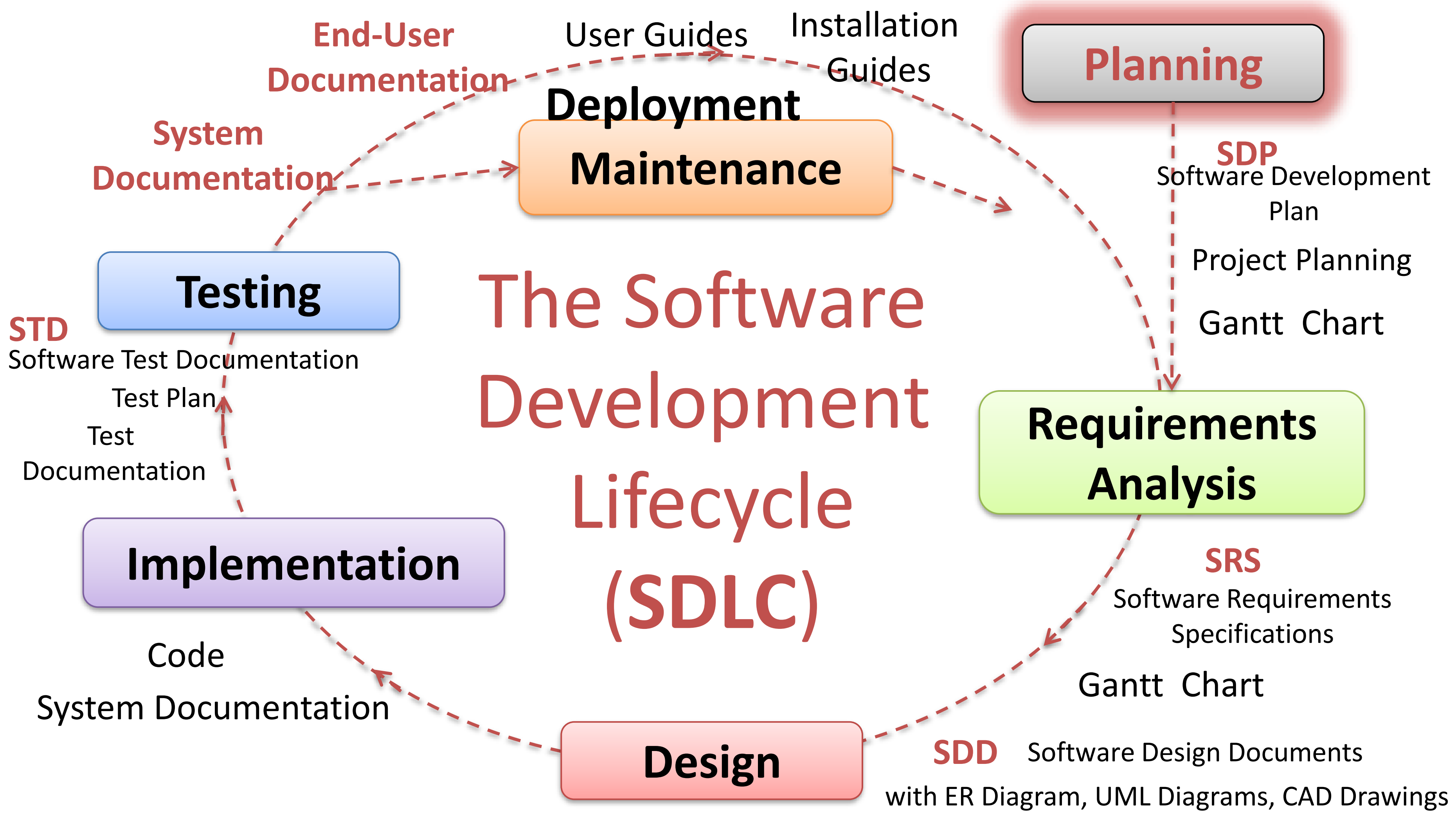
Hans-Petter Halvorsen

“Closing the Deal” – The Software Project can Start

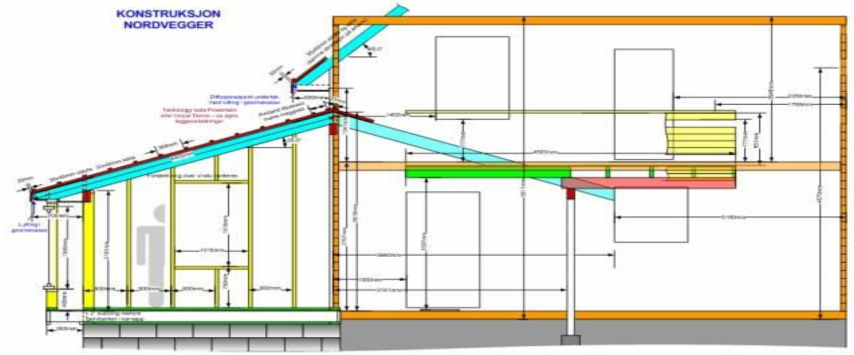

1121
FØR VI KAN SIGNERE KONTRAKTEN MED DERE, MÅ JEG VITE HVILKE STANDARDER SLUTTPRODUKTET KOMMER TIL Å TILFREDSSTILLE.
ALLE DU VIL! BARE DU IKKE TRENGER Å FÅ DET SKRIFTLIG.
NO PROBLEM.
© B.Lund/distr./strandcomics.no
DA VELGER JEG ALLE STANDARDER I HELE VERDEN PLUSS DE SOM IKKE ER INNFØRT ENDA. ER DET GREIT?
HØRES SMART UT. SKULLE DEKKE DET MESTE.
DU SKAL HELLER IKKE JOBBE MED DETTE PROSJEKTET I FORTSETTELSEN, SKAL DU?
NIX. ER BARE INNE NÅ I STARTEN FOR Å SIKRE AT VI KOMMER GODT IGANG.
BØRGE
B. Lund. *Lunch*. Available: http://www.lunchstriper.no, http://www.dagbladet.no/tegneserie/lunch/

You have chosen Project – Lets get started!

The Software Development Lifecycle (SDLC)



Requirements/Design



Plans made and approved

Alpha

Foundation finished, building structure started
A “proof” that you can do it, PoC (Proof of Concept)

Beta

Building structure finished,
Inside work on track

RC

Furniture, Flowers and
small adjustments missing

RTM

Ready for Sale or Move in

Level of complexity in your solution?

The 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**

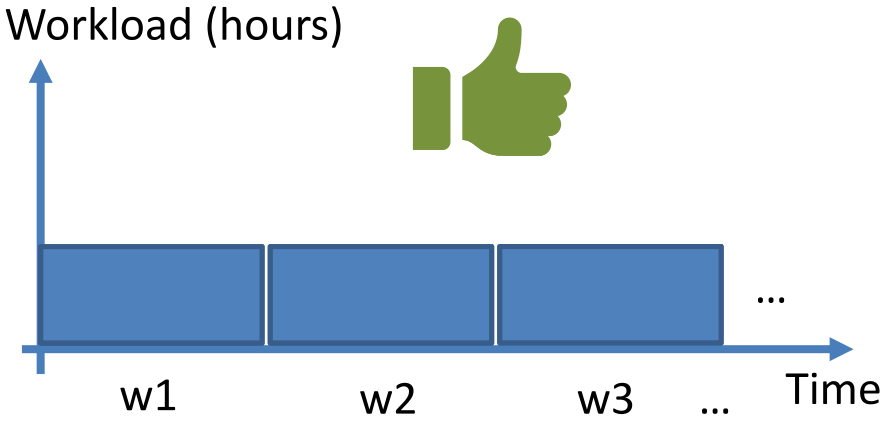
Conclusion:

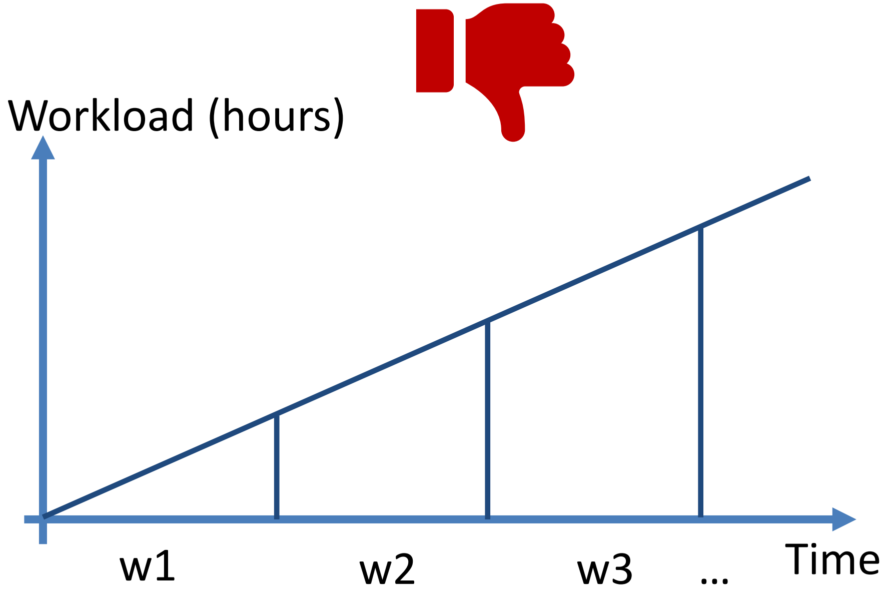
- Someone always tends to make things more complicated than necessary!
- The main goal in this Project and Course is to make a functional Prototype! – Not a fully working professional Product ready for sale
- Estimated Hours: 270+ hours/student

Grunnleggende Prosjektkrav

- **Visual Studio/C#**
- Systemet skal være **Modulbasert** **Warning!** Helt grunnleggende for å få en god karakter!
  - Vær så snill ikke lag et kjempestort/avansert program som gjør alt mulig rart – Del opp i Moduler! Det blir så mye enklere å lage, vedlikeholde og bruke! Typisk er det også forskjellig type brukere som vil bruke de forskjellige modulene.
  - Hvert team-medlem bør fortrinnsvis ha ansvaret for hver sin Modul/Applikasjon!
- **SQL Server** Database (<u>installeres lokalt på hver enkelt PC</u>)
- Minst en modul må være laget vha **ASP.NET Core**
- Dere må selv lage **Kravspesifikasjonen** til systemet, både **HVA** som skal lages og **HVORDAN** (Design) det skal lages

How to Work

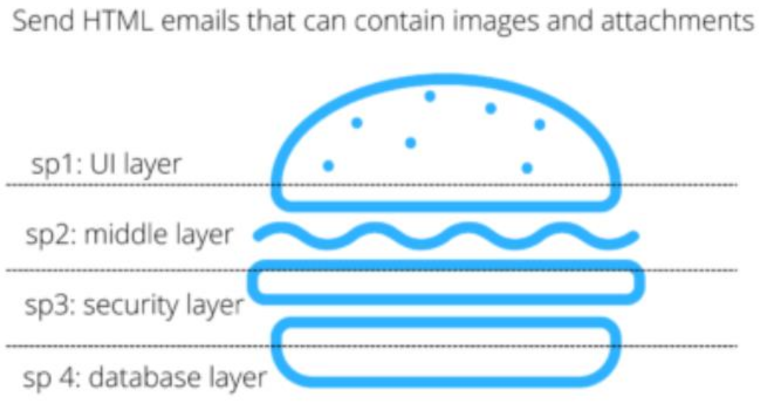




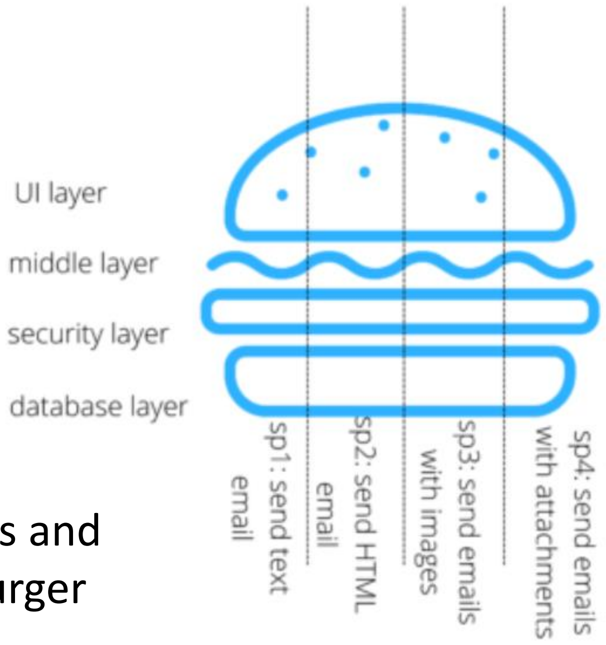
Ferdigstill forrige ukes planer og oppgaver slik at dere ikke får etterslep og må gjøre forrige ukes arbeid i tillegg til denne ukas arbeid. Da blir det mer og mer å gjøre for hver uke.

Plan-driven vs. Agile

Would you rather eat layers (left image) or slices (right image) of a burger?



(Plan-driven/Waterfall)



(Agile)

Important Agile principle: Working software, documents and product at all times, which is illustrated with the hamburger to the right

https://allthingsagile.co/post/agile-principle-7/

Project Start: Define Roles and Teams

Hans-Petter Halvorsen

Table of Contents

Define Teams and Roles

- Project Start: Organize yourself into Teams in the classroom
- Max 5 Teams
- 3-4 students in each Team (3 is optimal!!)
- Select a Project from the Project List

See Next Slides for more details...

Teams and Roles

- Customer/Stakeholders
- Project Manager
- Software Architect
- UX Designer
- Developer
- Tester
- etc.

Each of you should be and practice these different roles in this project

Customer/Stakeholders

Collaboration!

Software Architect

Software Tester

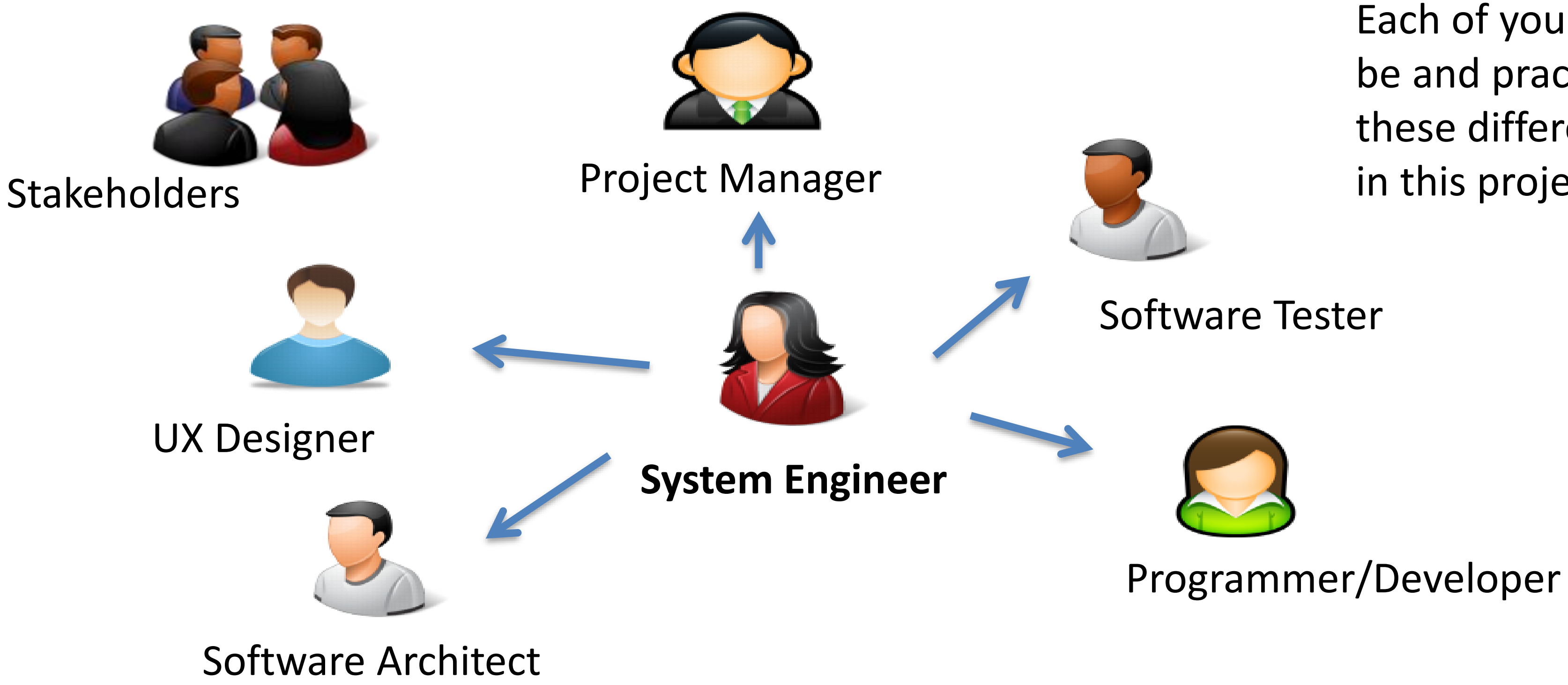
Project Manager

UX Designer

Programmer

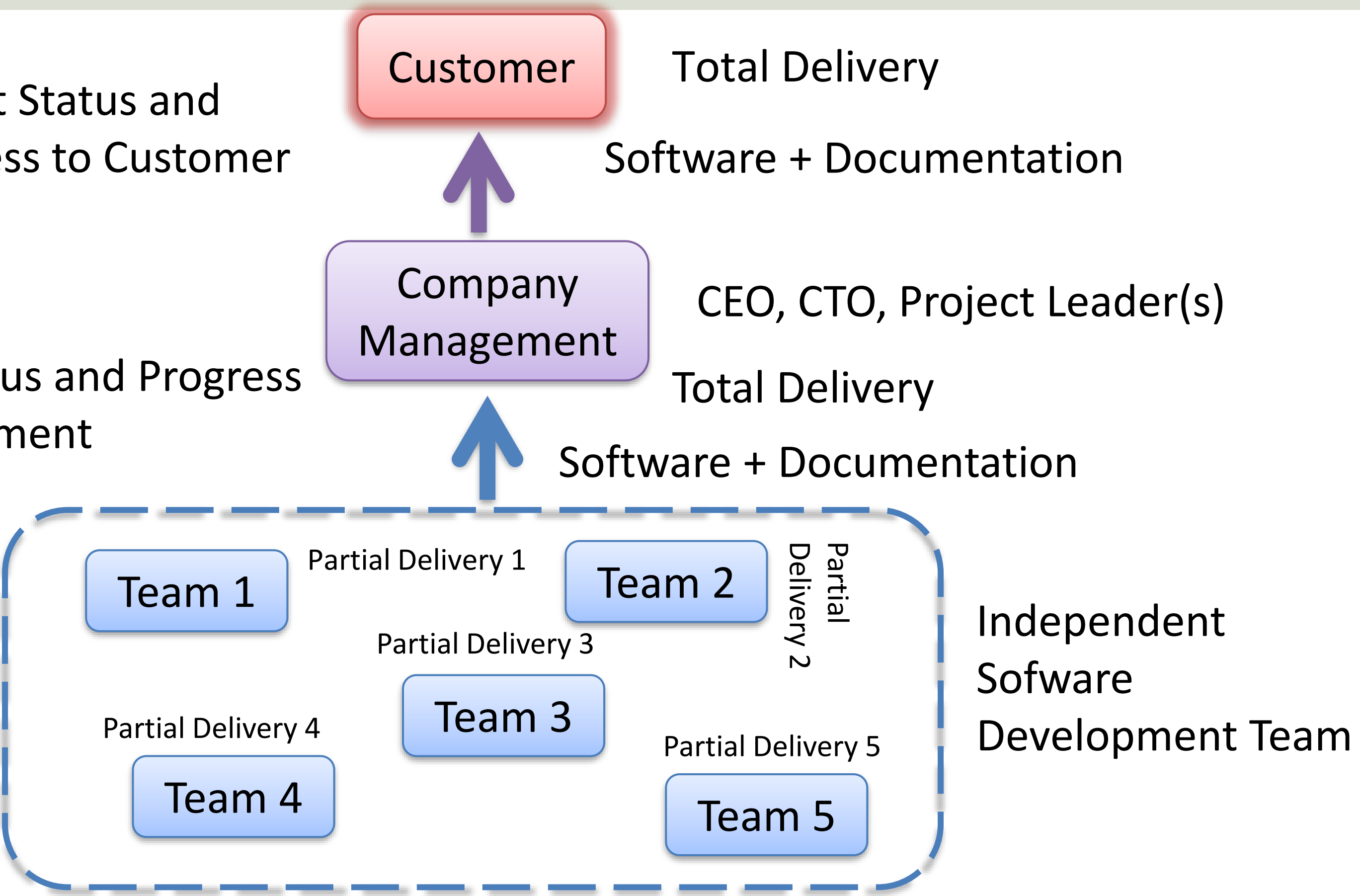
Software Teams

Each of you should be and practice these different roles in this project

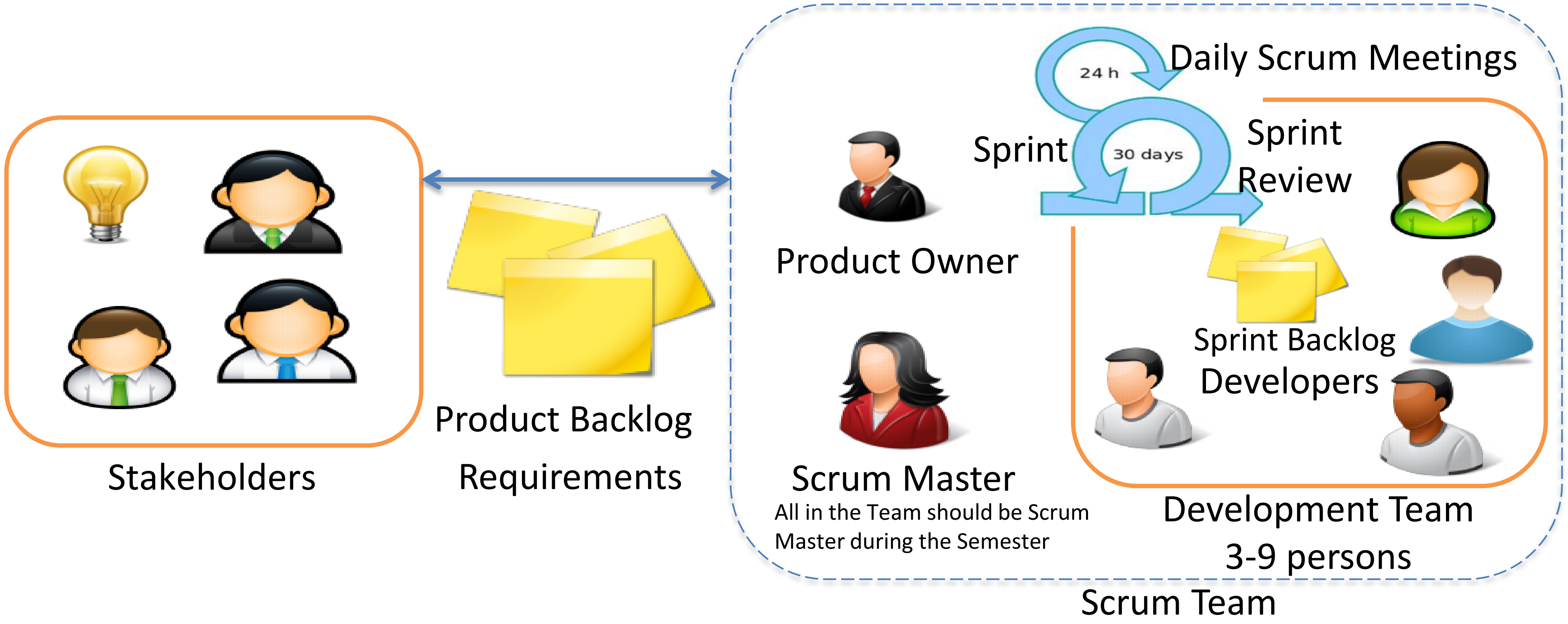


A System Engineer is a general person that could be a Programmer, Architect, Designer, Tester in different phases in the project, or he could be a tester in one project and a programmer in another project – all in one person. That is usually the case in small companies, while in larger companies these roles (designer, tester, programmer) could be a full-time job.

Project Organization



Scrum Teams and Roles



You will learn more about Agile Software Development (and Scrum) later

Team Skills/CV

- In order to get to know each other, each member should write a **CV**. Upload the document to Teams.
  - Background (former education, work experience, etc.)?, skills within Programming and Software Engineering? Project Work, etc.?
- Get an overview of the skills inside the team
- Use the **CV** just created as a foundation
- Who is best suited for the different parts that shall be solved in the project?
- The CVs may also be included as an appendix in the SDP.

Brainstorming

Brainstorming

- Perform a Brainstorming session inside your Team,
- i.e., find out **what** you shall do and **how** to do it
- The "big picture" – not (unnecessary) details yet. Make a basic System Sketch
  - What shall the System do?
  - Which Modules does the system consists of?
  - Main Functionality?
  - Hardware?
  - Language?

Make sure to include results from the Brainstorming into the SDP document

- In general, Don't think to complicated
  - Someone always tends to make things more complicated than necessary!
  - The main goal in this Project and Course is to make a functional Prototype! – Not a fully working professional Product ready for sale (Time Frame 270h/student)
  - 80% of the users only use 20% of the features

See Next Slides for more details...

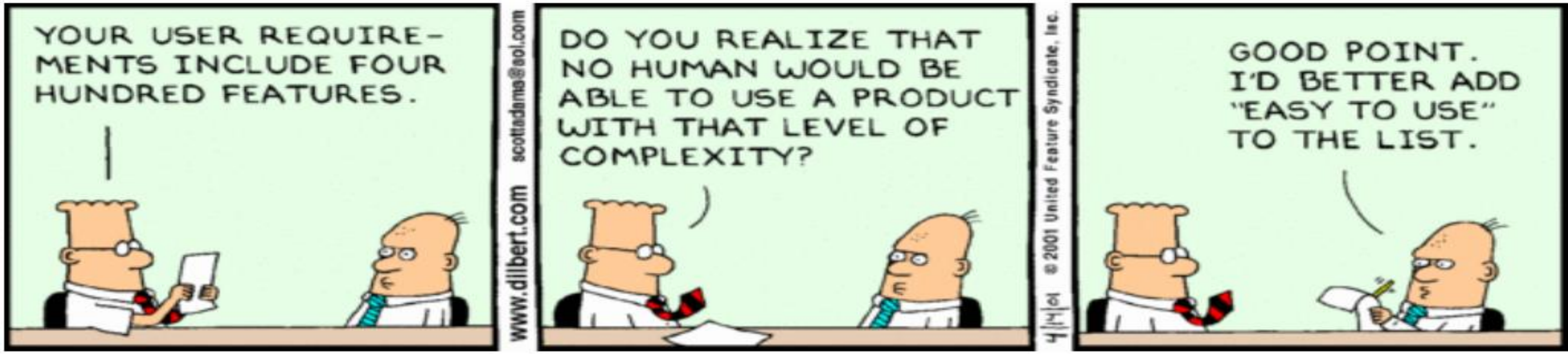
Brainstorming

The Brainstorming should include:

- High-level **Software Requirements**
  - What your system/applications should do
- High-level **Software Design**
  - How your application shall work and operate
  - What is the purpose with the application
  - Simple user interface design sketches, flow charts, etc.

Note! This will be a first draft, more details will be/should be added later.

High-Level Requirements vs. Detailed Requirements


YOUR USER REQUIREMENTS INCLUDE FOUR HUNDRED FEATURES.
DO YOU REALIZE THAT NO HUMAN WOULD BE ABLE TO USE A PRODUCT WITH THAT LEVEL OF COMPLEXITY?
GOOD POINT. I'D BETTER ADD "EASY TO USE" TO THE LIST.
www.dilbert.com scottadams@aol.com
4/14/01 © 2001 United Feature Syndicate, Inc.

High-Level Requirements

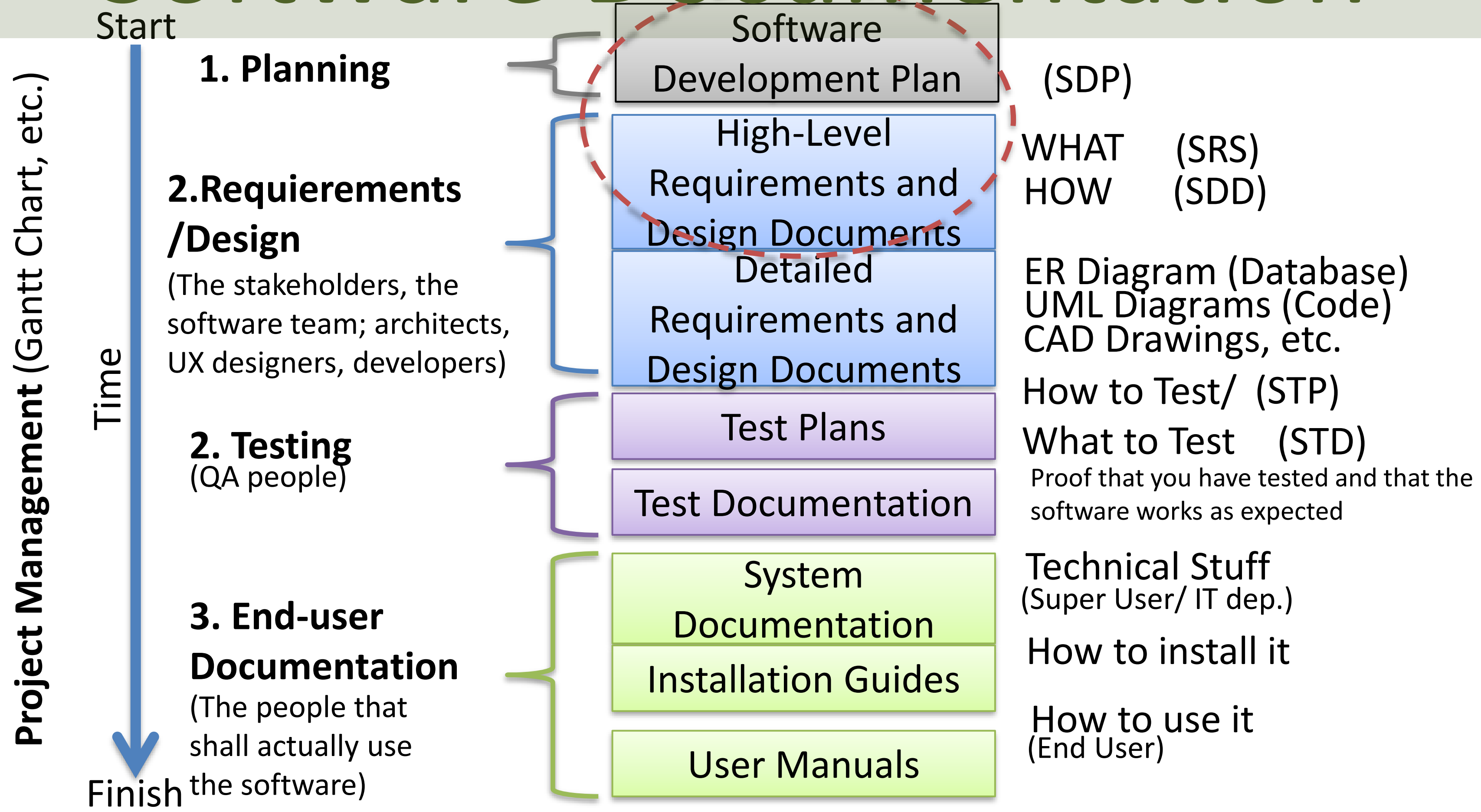
- **WHAT** should the system do?
- Who should use the system
- What is the purpose with the system?
- Performance
- What parts should the system consists of
- What Platforms should be used (PC, Tablet, Web?, ...)
- etc.

- Use Words and Figures in order to describe these Requirements

Detailed Requirements

- What Platforms should be used (Windows, iOS, ...) in more detail
- Tools and Languages
- Software Architecture ()
- Frameworks (.NET, ASP.NET, ...)
- Detailed GUI design sketches
- UML Diagrams
- ER (Database) Diagrams
- CAD Drawings
- etc.

Software Documentation



1. Planning

Software Development Plan (SDP)

2.Requierements /Design
(The stakeholders, the software team; architects, UX designers, developers)

High-Level Requirements and Design Documents

WHAT (SRS)
HOW (SDD)

Detailed Requirements and Design Documents

ER Diagram (Database)
UML Diagrams (Code)
CAD Drawings, etc.

2. Testing
(QA people)

Test Plans — How to Test/ (STP)

Test Documentation — What to Test (STD)
Proof that you have tested and that the software works as expected

3. End-user Documentation
(The people that shall actually use the software)

System Documentation — Technical Stuff (Super User/ IT dep.)

Installation Guides — How to install it

User Manuals — How to use it (End User)

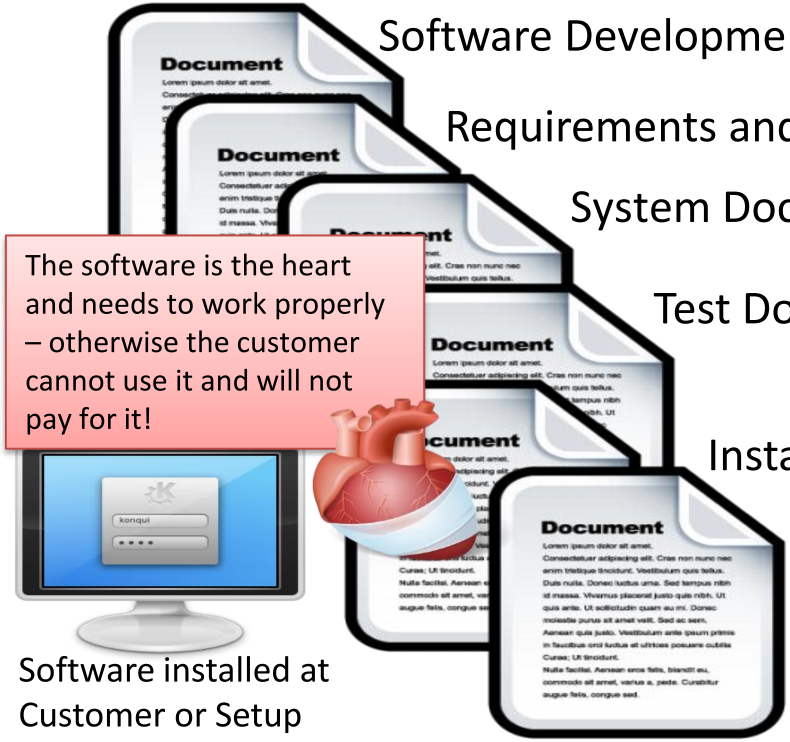
"Traditional School Project" vs. "Real Life Software Development Project"

<u>One</u> Report with "everything"

<u>One</u> document written by 3-6 students <u>together</u>

Does the system works? Hopefully – but never used or tested by the reader, Customer (or the students?)

We shall do it like this in this course!



Software Development Plan

Requirements and Design Documents

System Documentation

Test Documents

Installation Guides

User Guides

The software is the heart and needs to work properly – otherwise the customer cannot use it and will not pay for it!

Software installed at Customer or Setup available for download

Many documents written by different people and intended for different people

Software Development Plan

Hans-Petter Halvorsen

Table of Contents

Software Development Plan (SDP)

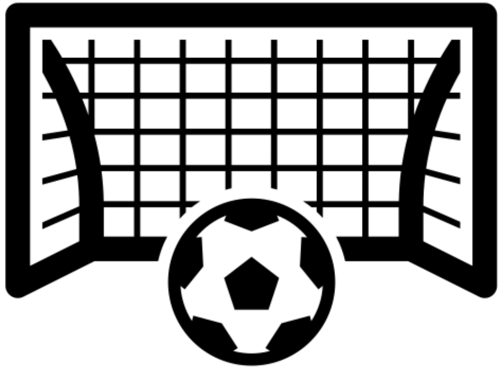
- **Create a Software Development Plan** (e.g., 8-10 pages)

The SDP should be places in Teams

- By spending time on the SDP now, you will save lots of time later!
- Also referred to as the "Communication Plan" or just "Project Plan"
- The **SDP** is a document that describes the Project, Resources, Communication, Schedule (e.g., Gantt chart), Tools, etc.

See Next Slides for more details...

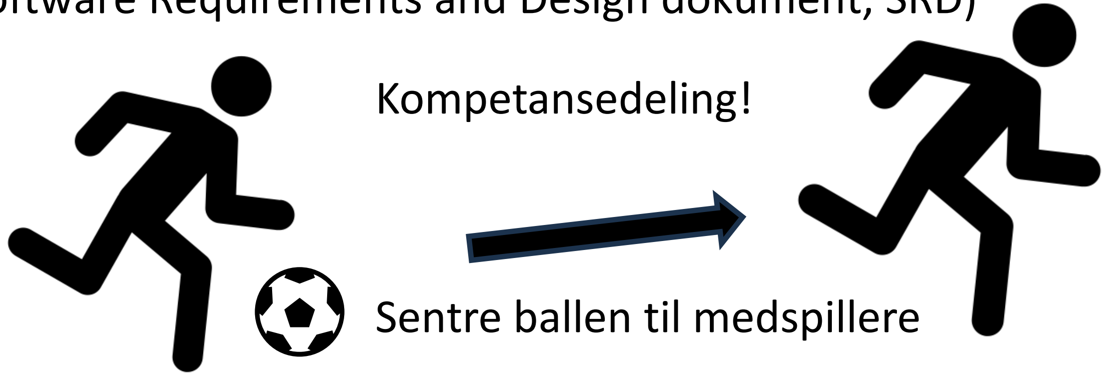
Systemutvikling = Fotball



Systemutvikling er Teamarbeid hvor man gjør hverandre gode

Laget består av individer som må samarbeide for å lykkes

Laget på lage en god plan for å vinne kampen (Softwareutvikling: Software Development Plan, SDP og Software Requirements and Design dokument, SRD)



Kompetansedeling!

Sentre ballen til medspillere

Det hjelper ikke å ha Ronaldo på laget hvis ikke spillerne klarer å samarbeide og gjøre hverandre gode. De må kommunisere godt og fordele arbeidsoppgaver, vite hvordan de skal spille, osv.

Fotballspillerne har forskjellige roller (spiss, forsvar, kaptein, ..), i Software Engineering har vi Programmerer, Tester, Arkitekt, Scrum Master, Prosjektleder, ..

Software Development Plan (SDP)

- A Software Development Plan (SDP) is all about the Internal Communication within the Development Team and how it Communicates with rest of the Organization, the Customers, etc.
- The purpose of the SDP is to communicate to team members and stakeholders how the software/project shall be caried out
- Examples of contents: Project Plan, Project Organization, Software and Tools that shall be used in the project management and development, etc.

Document Location?

- We will use **Microsoft Teams** for sharing and collaboration on Documents under construction
  - Here you can easily collaborate on writing documents in real-time
- We will use **Azure DevOps** to store and share Project Planning and Source Code
  - We should also share Release/Final Documents (Word files, Excel files, Visio files, etc.) in Azure DevOps

Software Development Plan (SDP)

Example 1

Note! Sommerville use the name “Project Plan” instead of SDP

An SDP normally include the following sections:

1. **Introduction**: This briefly describes the objectives of the project and set out the constraints (e.g., budget, time, etc.) that affects the management of the project. Simple **System Sketch**
2. **Project Organization** (Team Description) This section describes how the development team is organized, the people involved and their roles in the team. Software Process Model Description (Scrum, XP, Waterfall, ...), etc.
3. **Risk Analysis** (Hva kan gå galt? Og hvordan kan vi unngå det? Evt en en “bredskapsplan” hvis det skulle skje)
4. **Hardware and Software Resource Requirements**
5. **Work Breakdown** (WBS, Work Breakdown Structure): Break down the project in into activities and identifies milestones
6. **Project Schedule**: Shows dependencies between activities, the estimated time required to reach each milestone, allocation of people to activities. (5) and (6) is typically done in a **Gantt Chart** (created in e.g., Microsoft Project)
7. **Monitoring and Reporting Mechanisms**: Definition of the Management Report that sho when they should be produced, etc.
8. **Tools** that you are using

Ch. 23 Project Planning

I. Sommerville, *Software Engineering*: Pearson, 2015

Software Development Plan (SDP)

Example 2

Appendix A

A. Product Description
B. Team Description
C. Software Process Model Description
D. Project Definition
E. Project Organization
F. Validation Plan => Basic Test Plan
G. Configuration/Version Control
H. Tools

Depending on the size and complexity of the project, the plan itself may take several hours to several days or weeks to develop

For more details, see "Essentials of Software Engineering", Frank Tsui; Orlando Karam; Barbara Bernal, 3 or 4 ed., Jones & Bartlett Learning

Software Development Plan (SDP)

Example 2 with Details

A. **Product Description**: Describe the product and the client in general

B. **Team Description**: Describe the strengths/skills needed for the team members of this product. Give overview of Team members and their skills and roles.

C. **Software Process Model Description**: Describe the model (e.g., iterative Scrum, XP, or modified water- fall) to be used for this project. Include justifications for the process model choice.

D. **Project Definition**: Describe the users and the user environment. Include novice/expert descriptions. Consider creating different personas with different needs and motivations. For the user environment include the software used previously, other software used in conjunction, and the look and feel of the contemporary software genre. Can include use cases of the product, workflow diagrams, and/or business flow.

E. **Project Organization**: Include work breakdown structure (WBS) of the project: the schedule of the team's tasks; dependencies of the tasks; estimated time for each task; and PERT and Gantt charts with critical time, budget, and BID to the client with signature required.

F. **Validation Plan**: Create some draft input and output screens as low-level prototype to validate the initial understanding of the product. => Basic Test Plan (but with few details)

G. **Configuration/Version Control**: Specify the process and attributes for version control of all project and product artifacts.

H. **Tools**: Provide a list of major system, subsystem, and tools required for development.

Coding Conventions

- Coding Styles and Guidelines
- Pascal, Camel-case, lower-case or upper-case?
- How to define Variables, GUI components, etc.
- C#, SQL syntax, Table format, etc.
- It is important that all developer follow the same conventions
- Etc.

Note! All this information should be part of **SDP**

Gantt Chart

- You may create a simple draft of the Gantt Chart using MS Project this week.
- Important milestones, deadlines and available human resources should be part of the Project Plan (see the course schedule)
- It should be included in the Software Development Plan (SDP).
- More details will/should be added in the Next Week Assignment after the Requirements Analysis

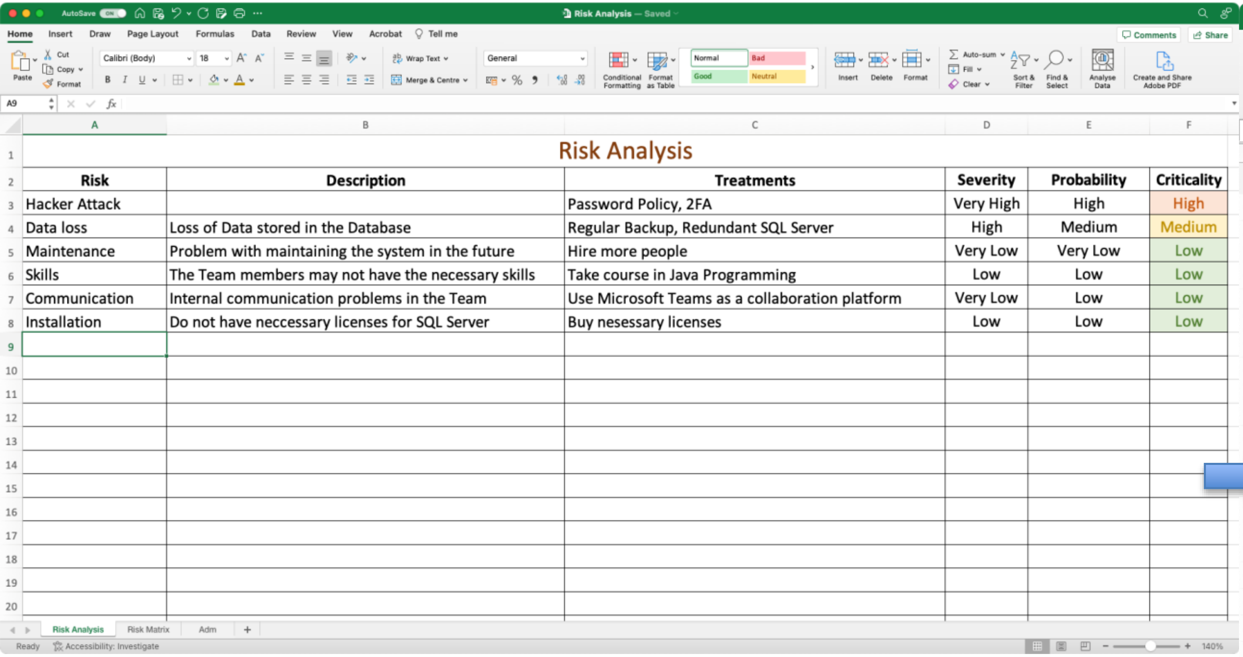
Risk Analysis

Identify and get an overview of the Risks in this Project. Risk Analysis and Management is the identification, evaluation, and prioritization of risks.

3 Steps:

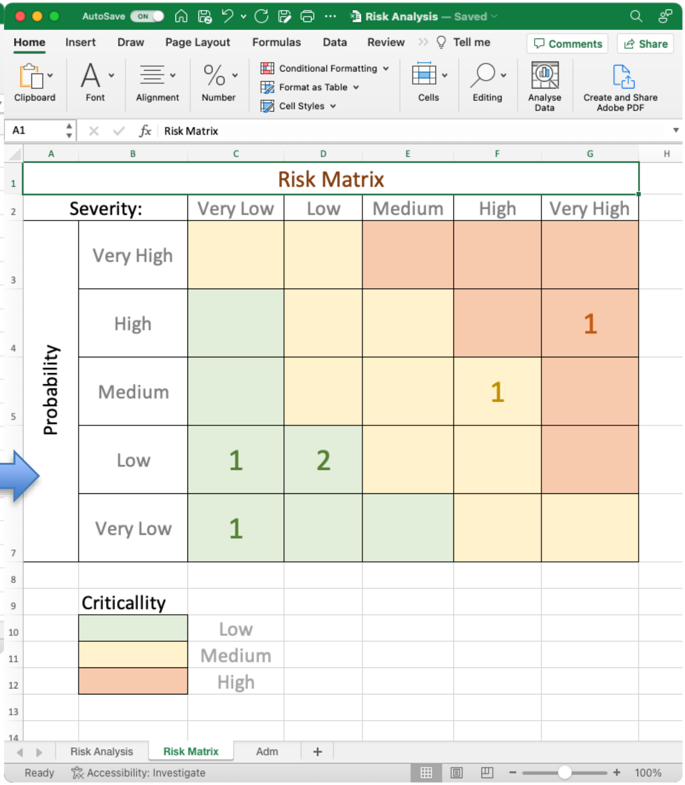
- **Step 1: Identify potential Risks**
  - The purpose is to create a comprehensive list of potential events that can trigger consequences that do not match our goals or wishes.
- **Step 2: Risk Analysis**
  - The Risk Analysis includes an assessment of both the causes and sources of the potential risks, the consequences they may have, as well as the likelihood that these consequences will occur.
  - Place your Risks in a Risk Matrix where you have Severity vs Probability.
- **Step 3: Risk Evaluation**
  - What shall you do if they occur or how can you avoid them? Make a list of Treatments for the most critical Risks that you have identified and analyzed.

Risk Analysis Example



Risk Analysis

Risk	Description	Treatments	Severity	Probability	Criticality
Hacker Attack		Password Policy, 2FA	Very High	High	High
Data loss	Loss of Data stored in the Database	Regular Backup, Redundant SQL Server	High	Medium	Medium
Maintenance	Problem with maintaining the system in the future	Hire more people	Very Low	Very Low	Low
Skills	The Team members may not have the necessary skills	Take course in Java Programming	Low	Low	Low
Communication	Internal communication problems in the Team	Use Microsoft Teams as a collaboration platform	Very Low	Low	Low
Installation	Do not have neccessary licenses for SQL Server	Buy nesessary licenses	Low	Low	Low



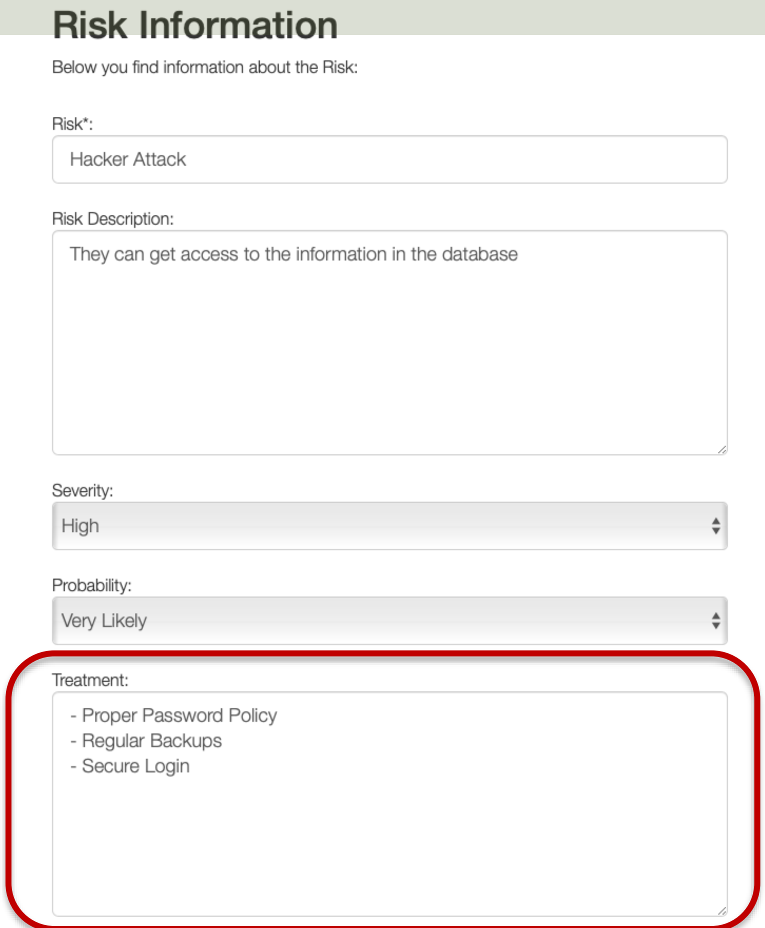
Risk Matrix

Probability	Severity:	Very Low	Low	Medium	High	Very High
	Very High	🟨	🟨	🟧	🟧	🟧
	High	🟩	🟨	🟨	🟧	🟧 1
	Medium	🟩	🟨	🟨	🟨 1	🟧
	Low	🟩 1	🟩 2	🟨	🟨	🟧
	Very Low	🟩 1	🟩	🟩	🟨	🟨

Criticallity

🟩	Low
🟨	Medium
🟧	High

Risk Analysis Example



Risk Information

Below you find information about the Risk:

Risk*:	Hacker Attack
Risk Description:	They can get access to the information in the database
Severity:	High
Probability:	Very Likely
Treatment:	- Proper Password Policy - Regular Backups - Secure Login

A Risk Analysis Item should typically contain the following:

- Risk **Name**
- Risk **Description**
- **Severity** (e.g., High, Medium, Low)
- **Probability** (e.g., Very Likely, Likely, Not Likely)
- Description of **Treatment**, i.e., how to avoid (or minimize it) or/and what to do if it occurs

What shall you do if they occur or how can you avoid or minimize it?

Document Structure/Layout/Formatting

- Use what you have learned in previous projects when it comes to writing documents and reports
- Typically, you have a Title Page, Table of Contents, Introduction Chapter, etc.
- Each Chapter should typically start on a New Page
- Use proper Referencing
- Use proper Figure-labeling and Figure-numbering. Make sure to refer to and explain the Figure in the text (the same goes for Tables and Equations)
- Make sure to include Page Numbering (e.g., in the Footer). The Chapter Name could, e.g., be in the Header
- Use same Layout, Fonts, Colors for all Documents
- All Documents should have a good structure and a “professional look and feeling”
- Etc. (All this should be available in **SDP**, either directly or link to a template, etc.)

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

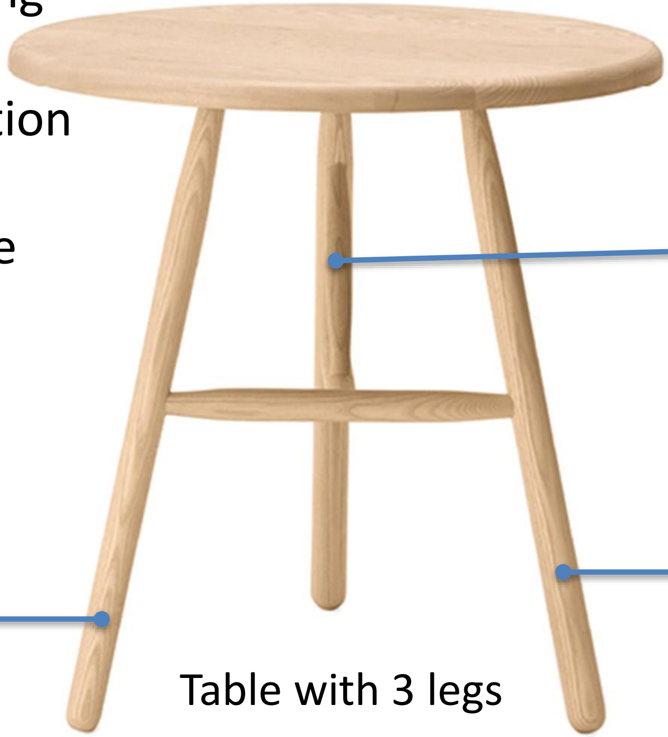
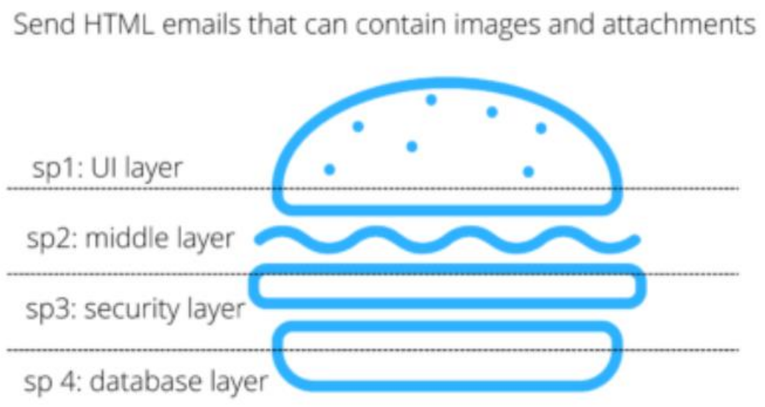


Table with 3 legs

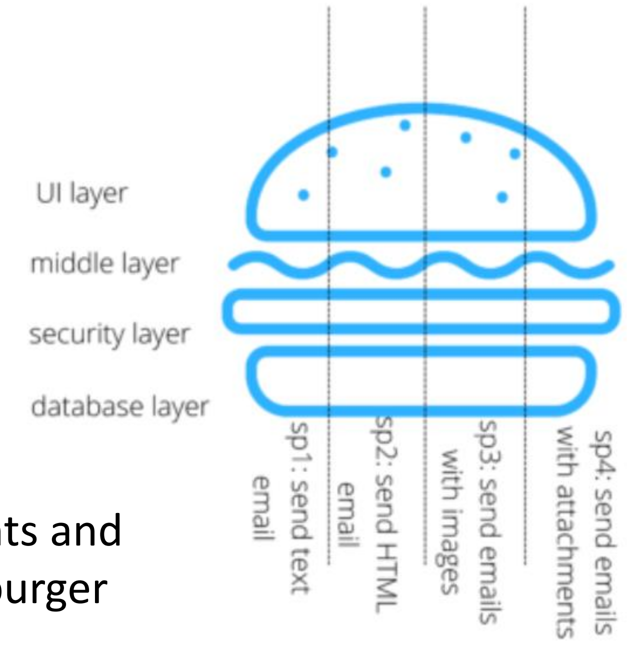
If you remove one of the legs, the table will fall apart

Plan-driven vs. Agile

Would you rather eat layers (left image) or slices (right image) of a burger?



(Plan-driven/Waterfall)



(Agile)

Important Agile principle: Working software, documents and product at all times, which is illustrated with the hamburger to the right

https://allthingsagile.co/post/agile-principle-7/

Development Tools

Hans-Petter Halvorsen

Table of Contents

Development Tools

1. **Microsoft Teams**
2. Install necessary Software:
    1. Microsoft **Visual Studio**
    2. **SQL Server** (“SQL Server Express is recommended)
    3. Microsoft **Visio** Professional
    4. Microsoft **Project** Professional
    5. **erwin** Data Modeler Academic Edition
    6. **Microsoft Teams** (Part of Office 365)
    7. ++ (Need more Software/Tools/Programming Languages later)
3. Get Started with **Azure DevOps**
    1. Create an Account and a New Team Project
    2. Add Team members, give Access to the Supervisors (Stakeholders)
    3. Create a good Folder Structure for your Documents and Code
    4. Create Iterations & Areas

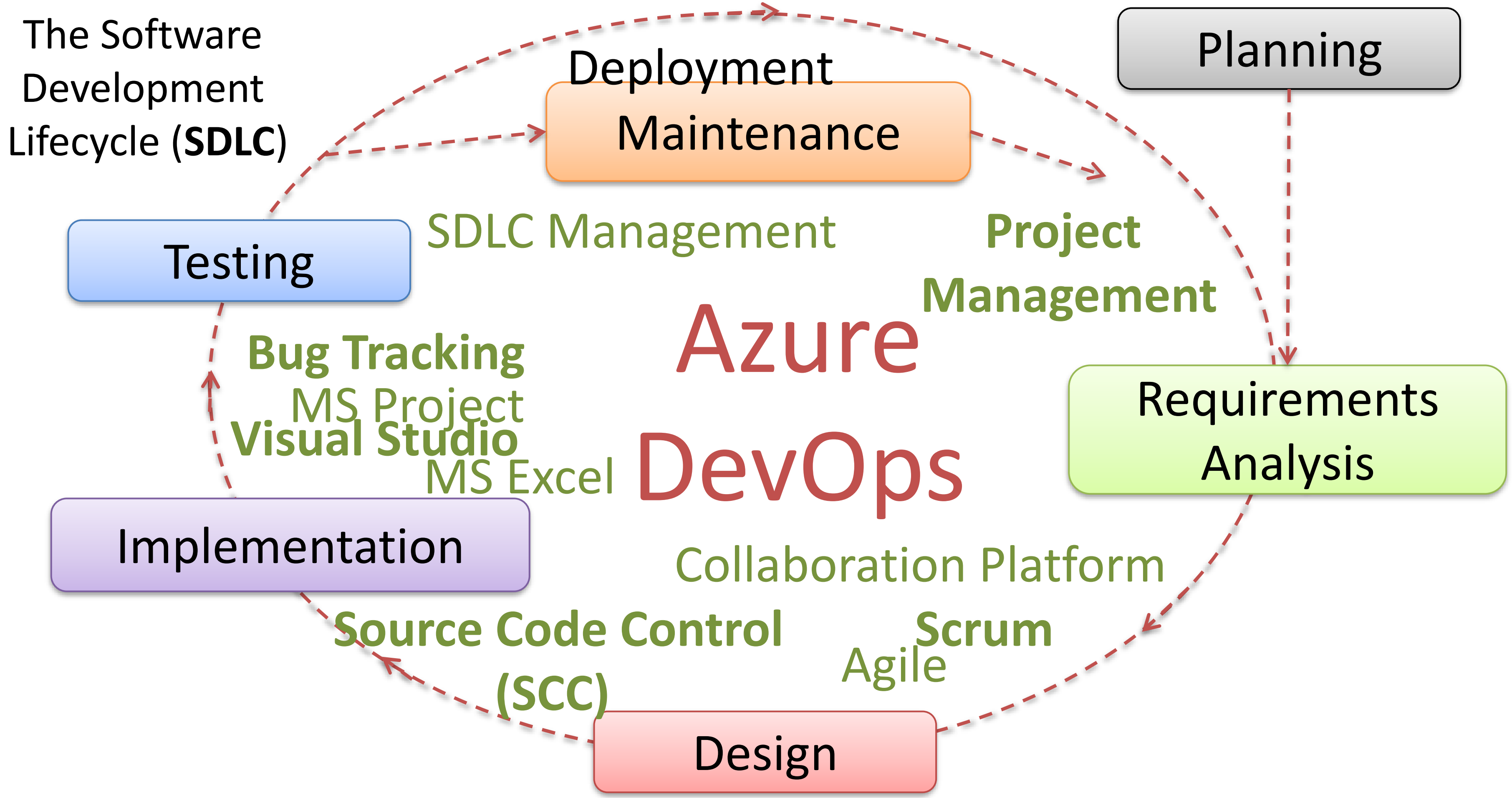
Azure DevOps

- We will use Azure DevOps as our software collaboration platform for our software lifecycle management (SDLC).
- Azure DevOps will be our main tool in addition to Microsoft Teams and Visual Studio
- Azure DevOps is located here: https://dev.azure.com

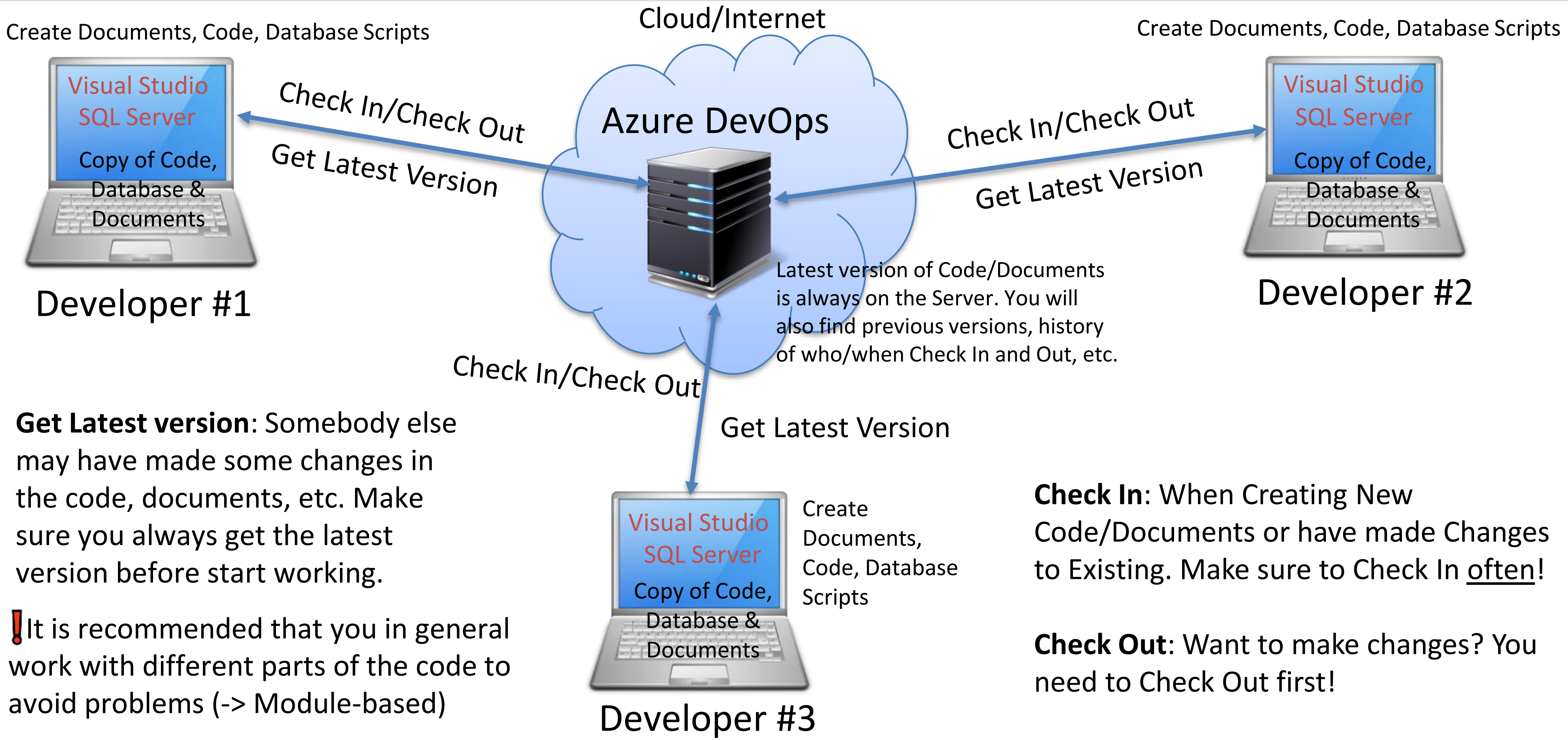
What is Azure DevOps?

- Azure DevOps is an Application Lifecycle Management (**ALM**) system,
  - i.e., the system takes care of all aspects in software development
  - from planning, requirements, coding, testing, deployment and maintenance.
  - Agile and Scrum workflow are included
- Azure DevOps is a Source Code Control (**SCC**), Bug Tracking, Project Management, and Team Collaboration platform
- Tightly integrated with **Visual Studio** as Microsoft is the vendor of both Visual Studio and Azure DevOps

Azure DevOps is an Application Lifecycle Management (**ALM**) System



Use Azure DevOps in Visual Studio



Cloud/Internet
Azure DevOps
Create Documents, Code, Database Scripts
Visual Studio
SQL Server
Copy of Code, Database & Documents
Developer #1
Check In/Check Out
Get Latest Version
Create Documents, Code, Database Scripts
Visual Studio
SQL Server
Copy of Code, Database & Documents
Developer #2
Check In/Check Out
Get Latest Version
Latest version of Code/Documents is always on the Server. You will also find previous versions, history of who/when Check In and Out, etc.
Check In/Check Out
Get Latest Version
Visual Studio
SQL Server
Copy of Code, Database & Documents
Create Documents, Code, Database Scripts
Developer #3

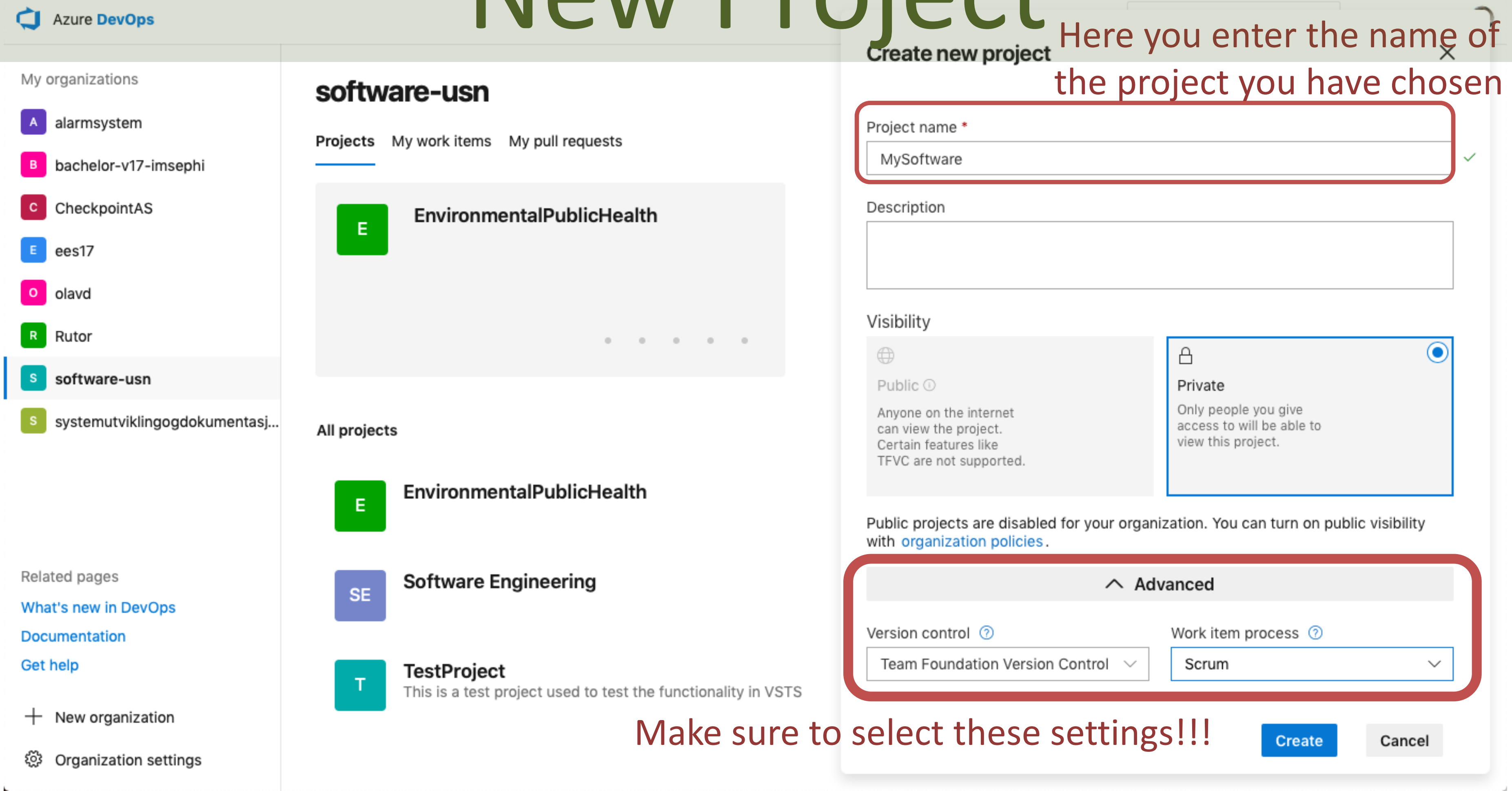
Get Latest version: Somebody else may have made some changes in the code, documents, etc. Make sure you always get the latest version before start working.

❗It is recommended that you in general work with different parts of the code to avoid problems (-> Module-based)

Check In: When Creating New Code/Documents or have made Changes to Existing. Make sure to Check In often!

Check Out: Want to make changes? You need to Check Out first!

New Project



Azure DevOps

My organizations

- alarmsystem
- bachelor-v17-imsephi
- CheckpointAS
- ees17
- olavd
- Rutor
- software-usn
- systemutviklingogdokumentasj...

Related pages

What's new in DevOps

Documentation

Get help

New organization

Organization settings

software-usn

Projects | My work items | My pull requests

EnvironmentalPublicHealth

All projects

EnvironmentalPublicHealth

Software Engineering

TestProject
This is a test project used to test the functionality in VSTS

Create new project

Project name *: MySoftware

Description:

Visibility

Public: Anyone on the internet can view the project. Certain features like TFVC are not supported.

[✓ Private]: Only people you give access to will be able to view this project.

Public projects are disabled for your organization. You can turn on public visibility with organization policies.

Advanced

Version control: Team Foundation Version Control

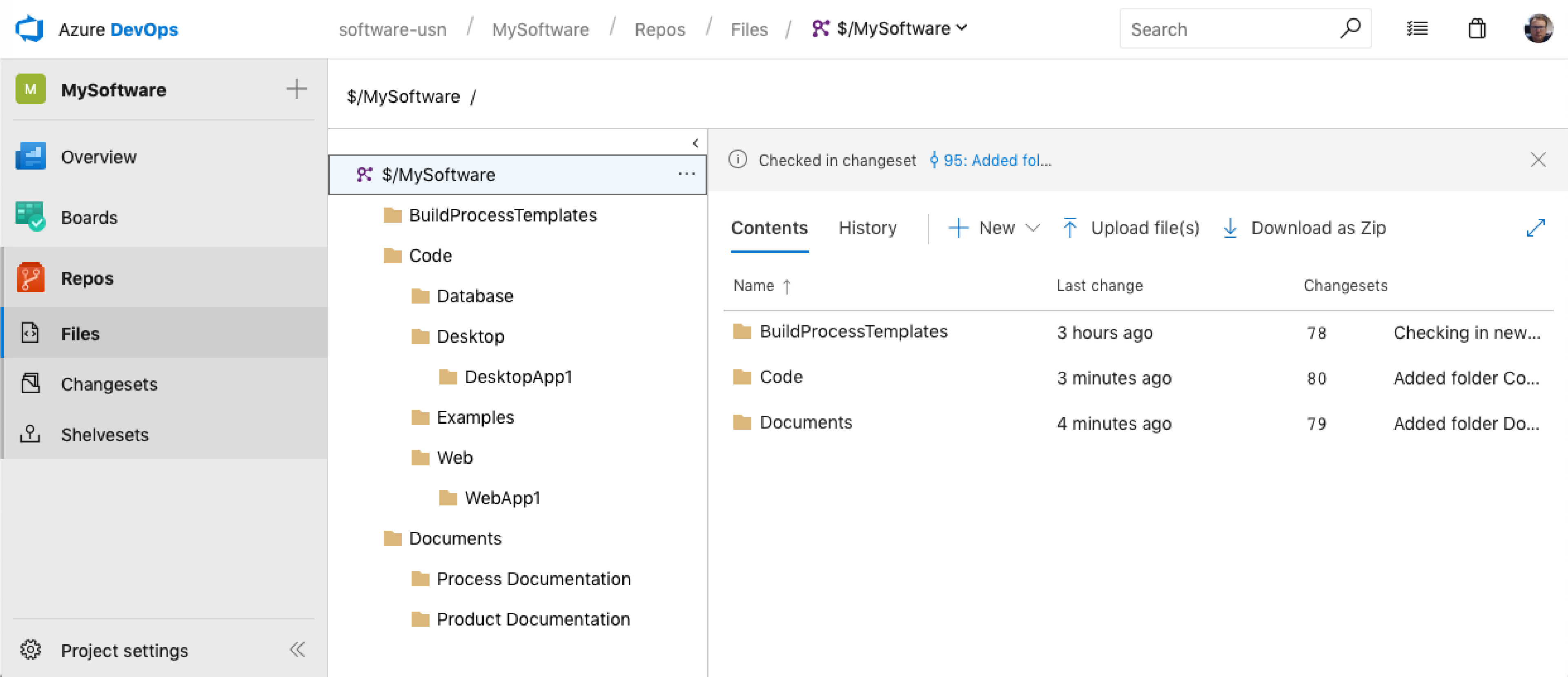
Work item process: Scrum

Create | Cancel

Here you enter the name of the project you have chosen

Make sure to select these settings!!!

Folders and Files



Azure DevOps | software-usn / MySoftware / Repos / Files / $/MySoftware

MySoftware: Overview, Boards, Repos, Files, Changesets, Shelvesets, Project settings

$/MySoftware /

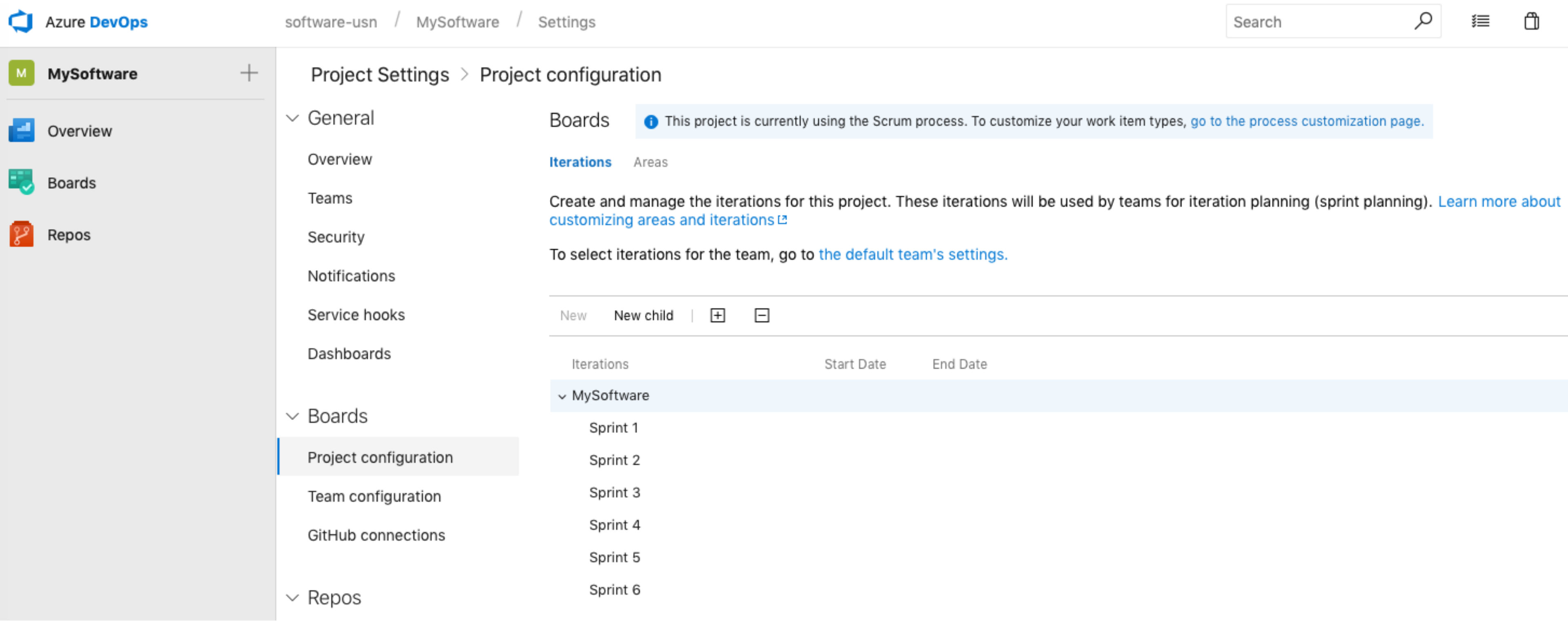
- $/MySoftware
  - BuildProcessTemplates
  - Code
    - Database
    - Desktop
      - DesktopApp1
    - Examples
    - Web
      - WebApp1
  - Documents
    - Process Documentation
    - Product Documentation

Checked in changeset 95: Added fol...

Contents | History | New | Upload file(s) | Download as Zip

Name ↑	Last change	Changesets	
BuildProcessTemplates	3 hours ago	78	Checking in new...
Code	3 minutes ago	80	Added folder Co...
Documents	4 minutes ago	79	Added folder Do...

Configure Iterations/Sprints



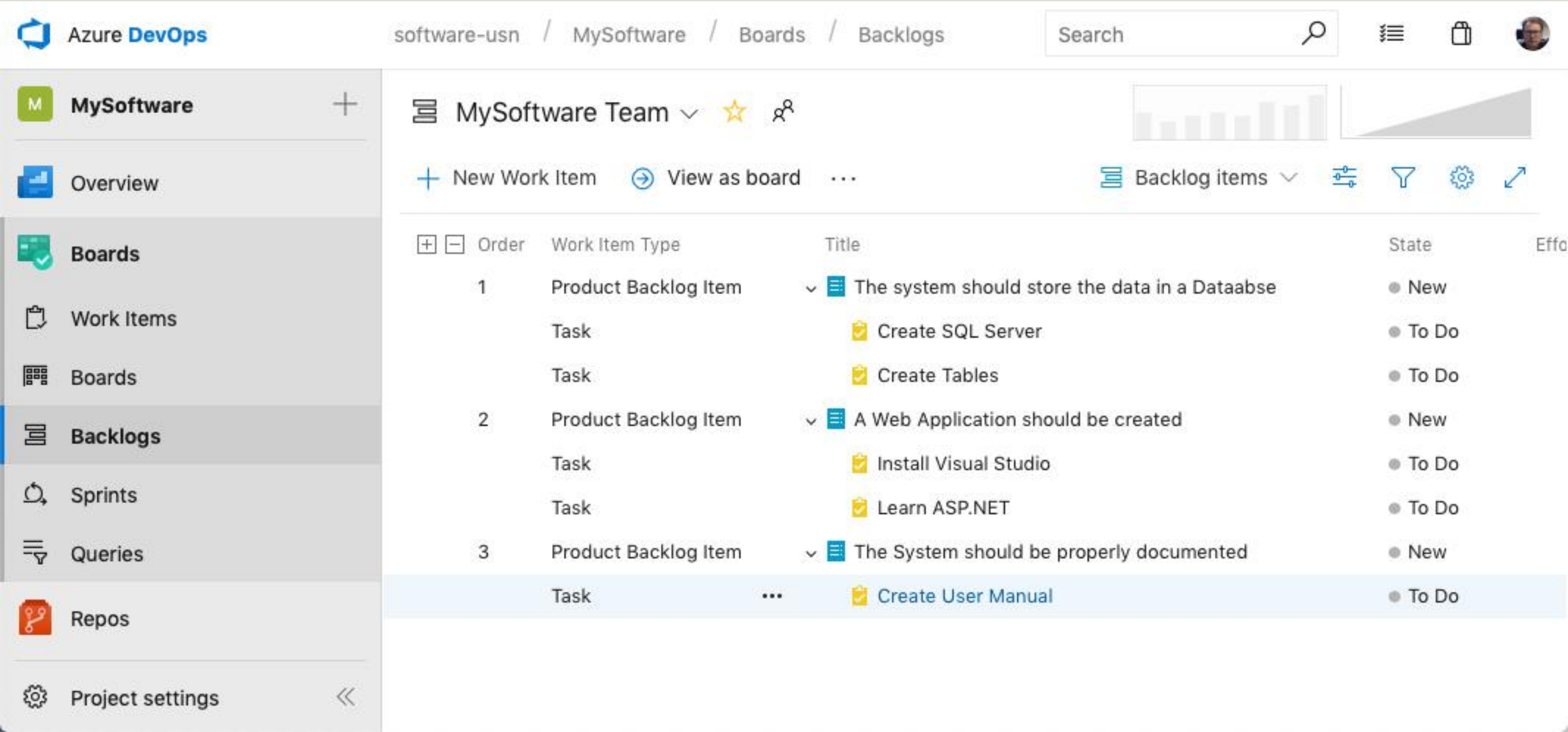
This project is currently using the Scrum process. To customize your work item types, go to the process customization page.

Create and manage the iterations for this project. These iterations will be used by teams for iteration planning (sprint planning). Learn more about customizing areas and iterations

To select iterations for the team, go to the default team's settings.

Iterations	Start Date	End Date
MySoftware		
Sprint 1		
Sprint 2		
Sprint 3		
Sprint 4		
Sprint 5		
Sprint 6		

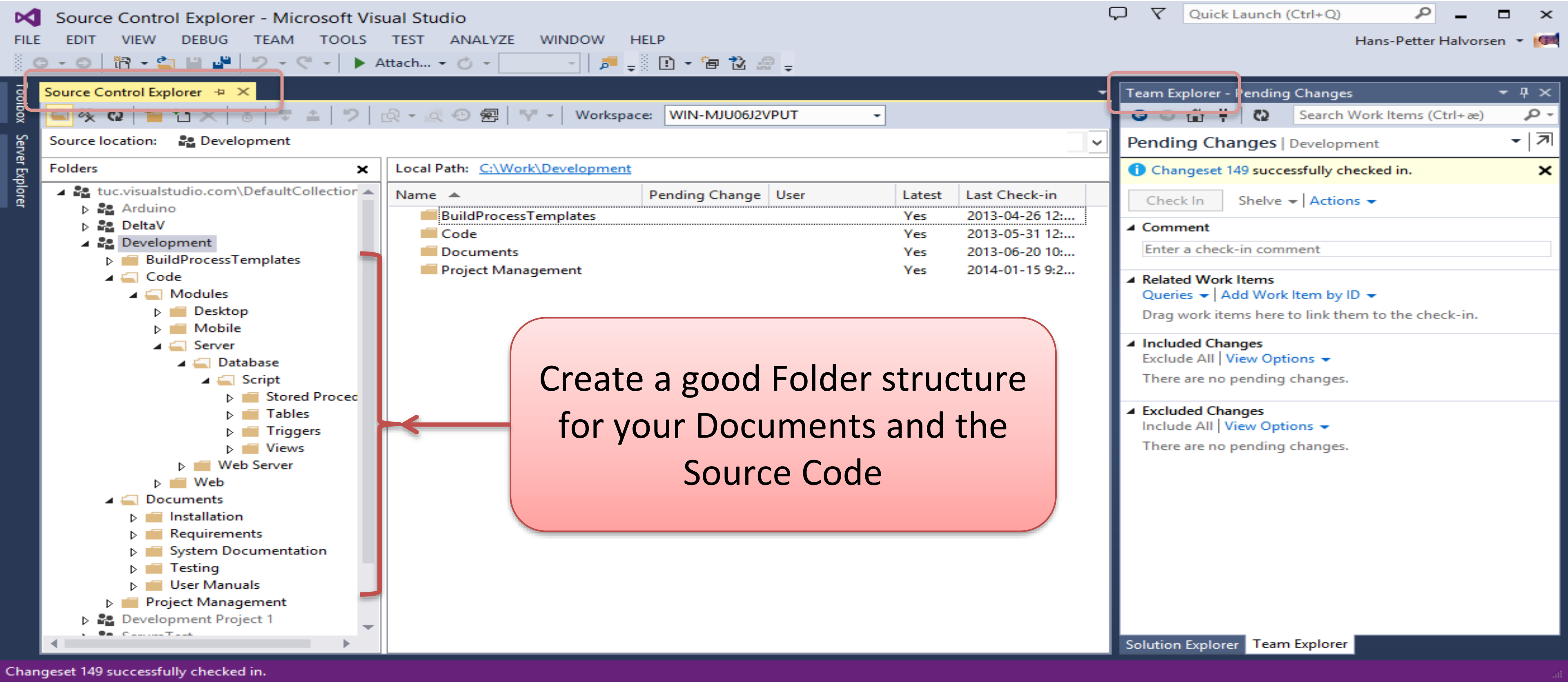
Product Backlog Items and Tasks



Order	Work Item Type	Title	State	Effo
1	Product Backlog Item	The system should store the data in a Dataabse	New	
	Task	Create SQL Server	To Do	
	Task	Create Tables	To Do	
2	Product Backlog Item	A Web Application should be created	New	
	Task	Install Visual Studio	To Do	
	Task	Learn ASP.NET	To Do	
3	Product Backlog Item	The System should be properly documented	New	
	Task	Create User Manual	To Do	

Azure DevOps in Visual Studio

Open the **Source Control Explorer**



Source Control Explorer - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Hans-Petter Halvorsen
Attach...
Source Control Explorer
Workspace: WIN-MJU06J2VPUT
Toolbox
Server Explorer

Source location: Development

Folders

- tuc.visualstudio.com\DefaultCollection
  - Arduino
  - DeltaV
  - Development
    - BuildProcessTemplates
    - Code
      - Modules
        - Desktop
        - Mobile
        - Server
          - Database
            - Script
              - Stored Proced
              - Tables
              - Triggers
              - Views
          - Web Server
        - Web
    - Documents
      - Installation
      - Requirements
      - System Documentation
      - Testing
      - User Manuals
    - Project Management
  - Development Project 1

Local Path: C:\Work\Development

Name	Pending Change	User	Latest	Last Check-in
BuildProcessTemplates			Yes	2013-04-26 12:...
Code			Yes	2013-05-31 12:...
Documents			Yes	2013-06-20 10:...
Project Management			Yes	2014-01-15 9:2...

Create a good Folder structure for your Documents and the Source Code

Team Explorer - Pending Changes

Search Work Items (Ctrl+æ)

Pending Changes | Development

Changeset 149 successfully checked in.

Check In   Shelve | Actions

Comment

Enter a check-in comment

Related Work Items

Queries | Add Work Item by ID

Drag work items here to link them to the check-in.

Included Changes

Exclude All | View Options

There are no pending changes.

Excluded Changes

Include All | View Options

There are no pending changes.

Solution Explorer   Team Explorer

Changeset 149 successfully checked in.

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog