TDT 4242 Advanced Software Engineering

- Spring 2022

About Jingyue Li (Bill)

- Jingyue Li (Bill)
  - Master (Computer science) in China
  - Architect: IBM China Ltd.
    - Banking solutions
  - PhD and Post-Doc (Software engineering) at IDI, NTNU
  - Principal researcher: DNV Research & Innovation
    - Software engineering
    - Dependability of critical software systems
  - Teaching experience:
    - Software architecture, IT projects, software security and data privacy, advanced software engineering, empirical software engineering
    - Started teaching this course from 2017
  - Jingyue.li@ntnu.no

About Daniela

- Co-Responsible teacher: Daniela Cruzes
- Master (Computer science) in Brazil
- Dr. Ing (Software engineering) in Brazil
- Post-Doc in USA and at IDI
- Tester: CPqD.
- Billing solutions for Telecom.
- Software Security Officer: FARA.
- Transport Intelligent Systems
- Senior Research Scientist: SINTEF
- Professor of Software Engineering:
- NTNU
- Lead Security Researcher : VISMA

- Email: dcruzes@ntnu.no

About Anh Nguyen-Duc

- Occupation & Experience
  - Assoc. Prof. at NTNU, Assoc. Prof. At USN
  - Entrepreneurs (Muml, VVN, etc), Startup mentors (VVNOR, Klikkit, Mitanbud, Comartek AS, etc)
- Research interest:
  - Software startups & software ecosystem
  - Secure software development, Static analysis
  - AI for SE, SE for AI, AI ethics
- Education
  - Master (Software Engineering) in Sweden & Germany
  - PhD and Post-Doc (Software engineering) at IDI, NTNU
- Teaching experience:
  - Customer driven project (TDT4290), Software engineering (TDT4140), IT project (IT2901), Web development & HCI (WEB1000), Software process (SYS1000), Project management (PRO1000, MIS405), Green IT (MSM4103), Product Process and People (MS4102)
- Contact: anhn@ntnu.no or angu@usn.no

Outline

- ➢ Knowledge to learn to be a good software engineer
- What advanced SE knowledge to be covered in this course?
- How is this course organized?

Goals of the software methodology – Ensure value creation

- Ensure economic gain
- Ensure personal gain
- Ensure organizational (business) gain
- Ensure societal gain

- Phases of value achievement
  - Plan for value achievement. Business case and value realization plan
  - Perform change
  - Implement change and realize the value

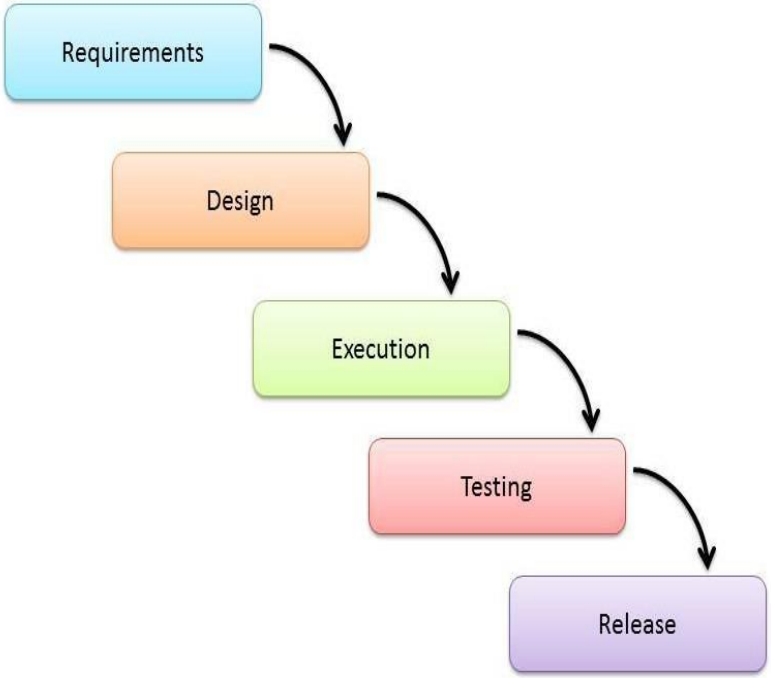
Categories of software based on functions

- Application software (web, mobile, IoT, control SW)
- System software (OS, DB, …)
- Computer programming tools (compiler, MATLAB, etc.)
- …

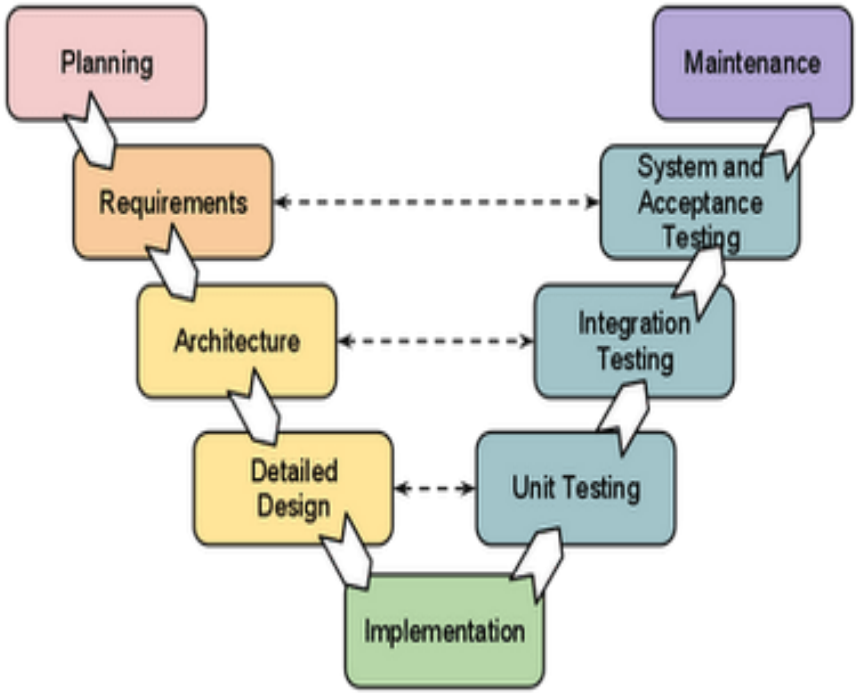
Project types

- In-house development:
  - The same company makes requirements, develops, and uses the system
- Custom development:
  - One company is customer, and
    - Another company analyzes requirements and develops system
    - Another company analyzes requirements, yet another develops
  - May include outsourcing to subcontractors
- Mass-market software
  - Project to buy or rent package (SaaS)
  - Project to develop an adaptation of package (e.g., ERP-system)
- Reuse of resources in digital or software ecosystems

Typical software development lifecycle models*



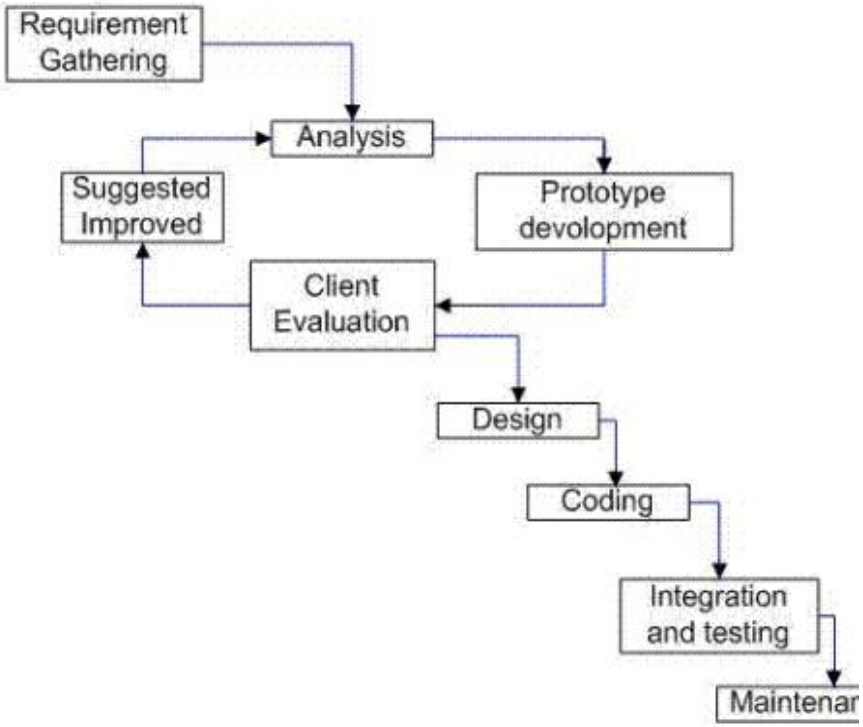
Waterfall model



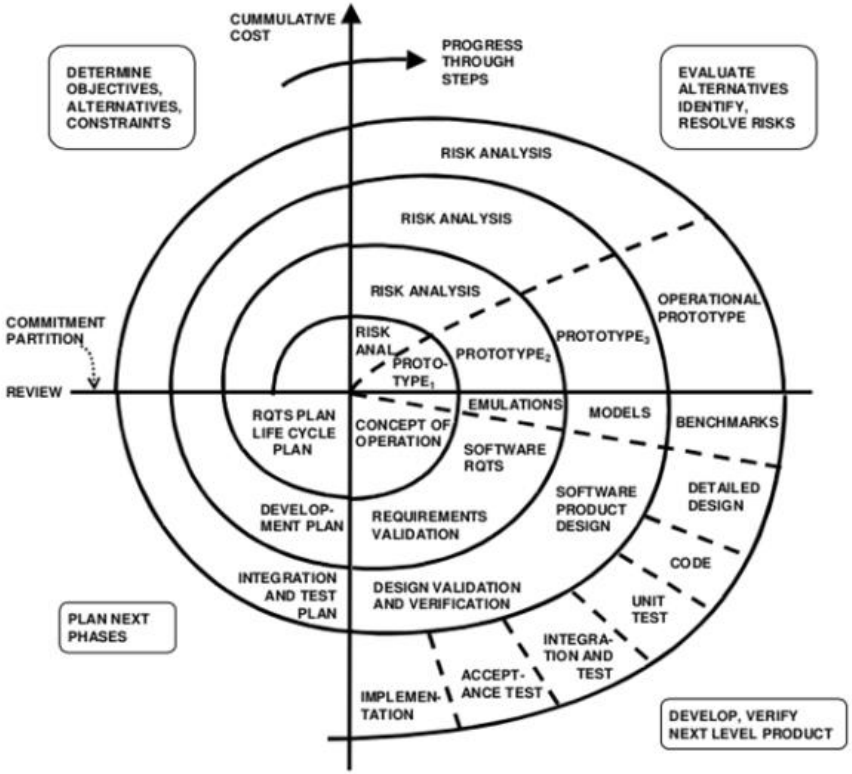
V-model

* https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

Typical software development lifecycle models (cont')*



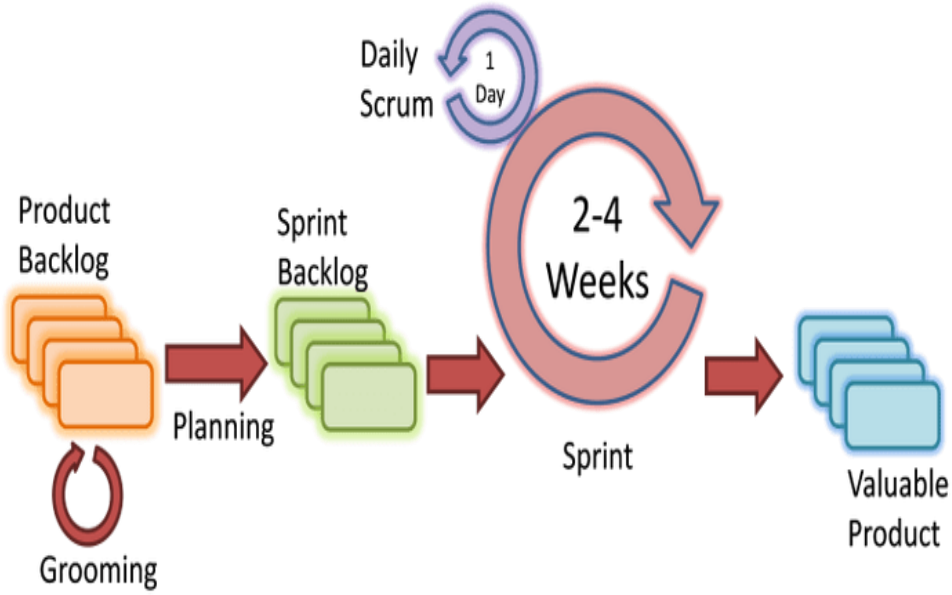
Evolutionary Prototyping Model



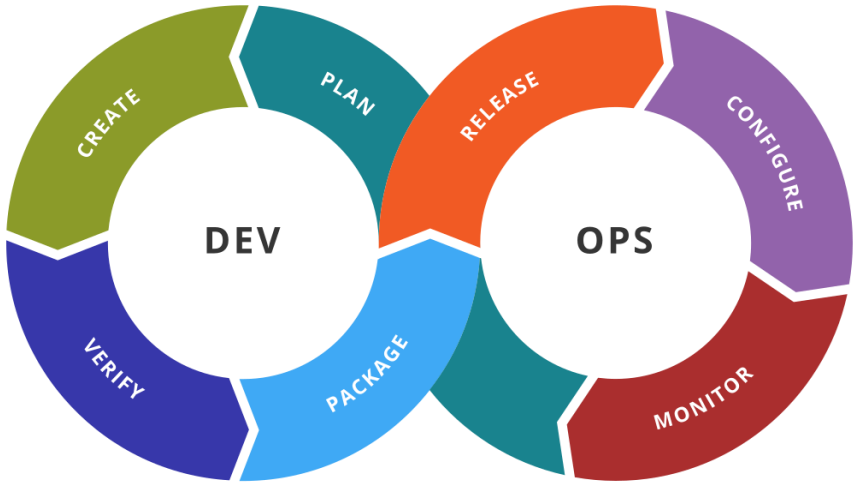
Spiral Model

* https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

Typical software development lifecycle models (cont")*



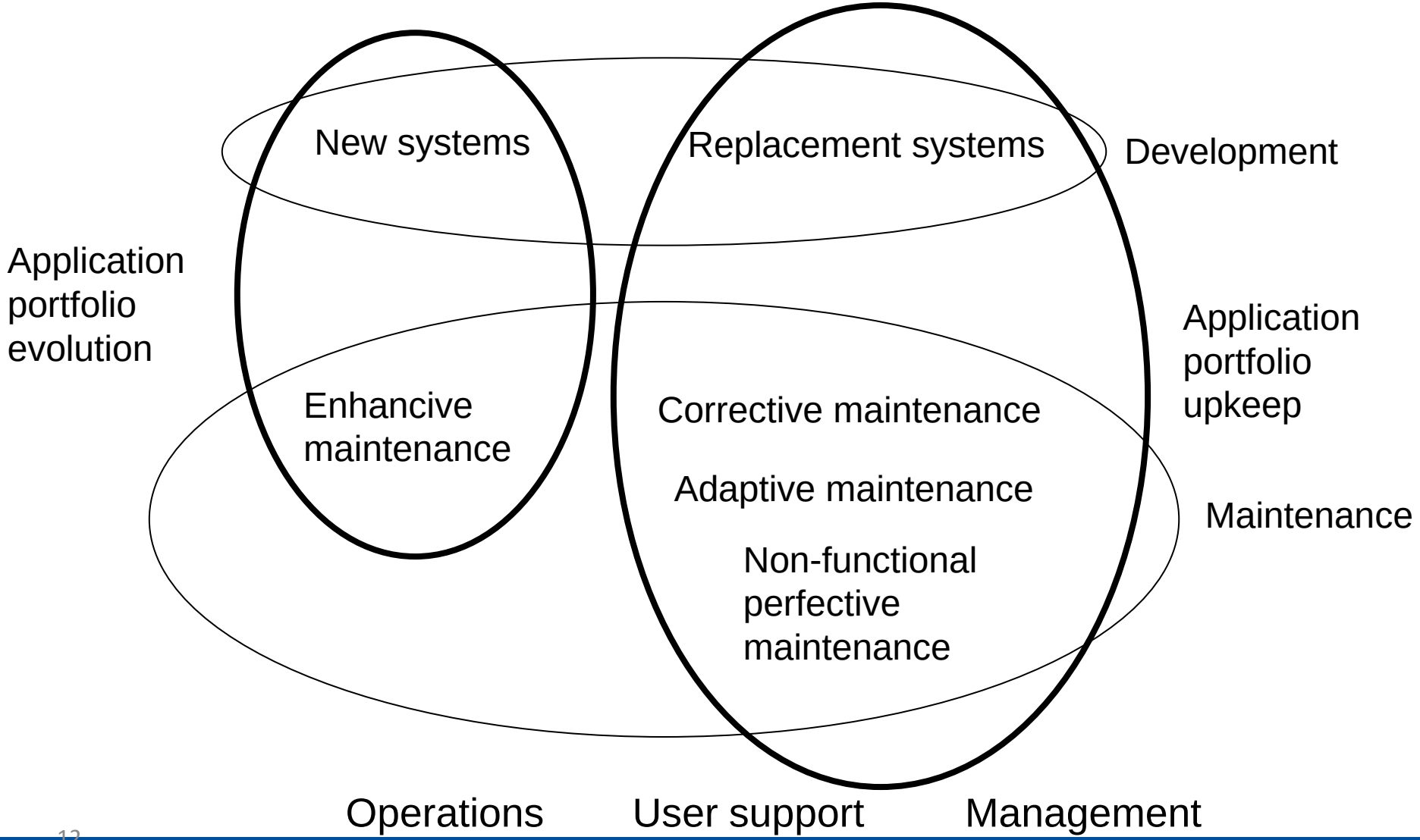
Agile Model



DevOps

* https://melsatar.blog/2012/03/15/software-development-life-cycle-models-and-methodologies/

Tasks within development and maintenance of software systems



Software Engineering Body of Knowledge (SWEBOK)* covered

- Software requirements
- Software design
- Software construction
- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process

*https://www.computer.org/web/swebok

Software Engineering Body of Knowledge (SWEBOK) covered (Cont')

- Software engineering models and methods
- Software quality
- Software engineering professional practice
- Software engineering economics
- Computing foundations
- Mathematical foundations
- Engineering foundations

Related courses

- TDT4140 Software Engineering
- TDT4250 Advanced Software Design
- TDT4290 Customer-driven project

Outline

- Knowledge to learn to be a good software engineer
- ➢ What advanced SE knowledge to be covered in this course?
- ➢ How is this course organized?

High-level course modules

- High-quality requirements
- High-quality and effective testing
- High-quality and maintainable code
- Advanced topics
  - DevOps
  - Cost and defect estimation
  - Software startup
  - Software eco-systems

Tentative lecture plan

High-quality requirements

Week	Date	Lecture content	Speaker
3	17.01.22	Requirement engineering 1	Anh Nguyen Duc
4	24.01.22	Requirement engineering 2	Anh Nguyen Duc

Challenges in Requirements Engineering

- Importance of getting requirements right:
  1/3 budget to correct errors originates from requirements

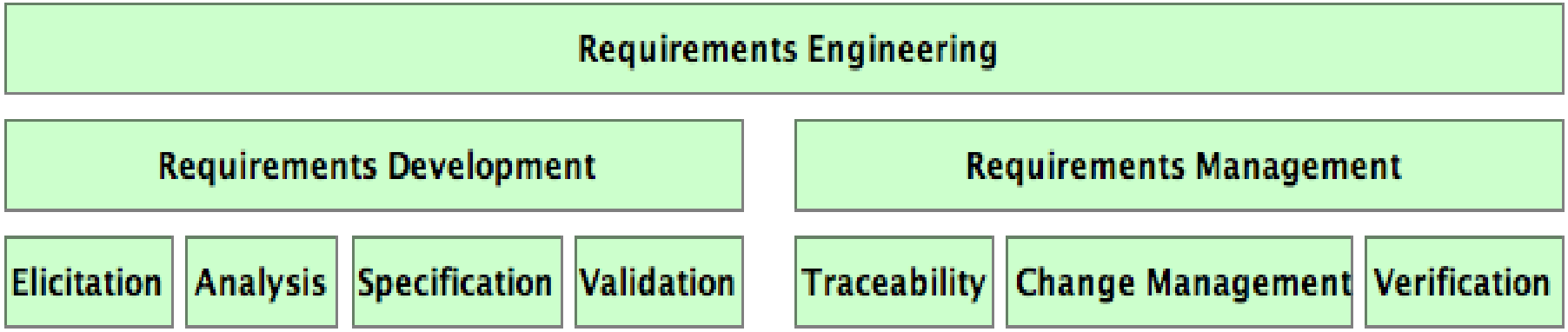
Phase in which fixed	Relative Cost
Requirements	1
Design	3 – 6
Coding	10
Development Testing	15 – 40
Acceptance Testing	30 – 70
Operations	40 – 1000

Challenges in Requirements Engineering (Cont')

- Why projects are canceled:

Factors	Percentage of Responses
Incomplete Requirements	13.1%
Lack of User Involvement	12.4%
Lack of Resources	10.6%
Unrealistic Expectations	9.9%
Lack of Executive Support	9.3%
Changing Requirements	8.7%
Lack of Planning	8.1%
Didn't need it any longer	7.5%
Lack of IT Management	4.3%
Technology Illiteracy	9.9%

Requirements Engineering process



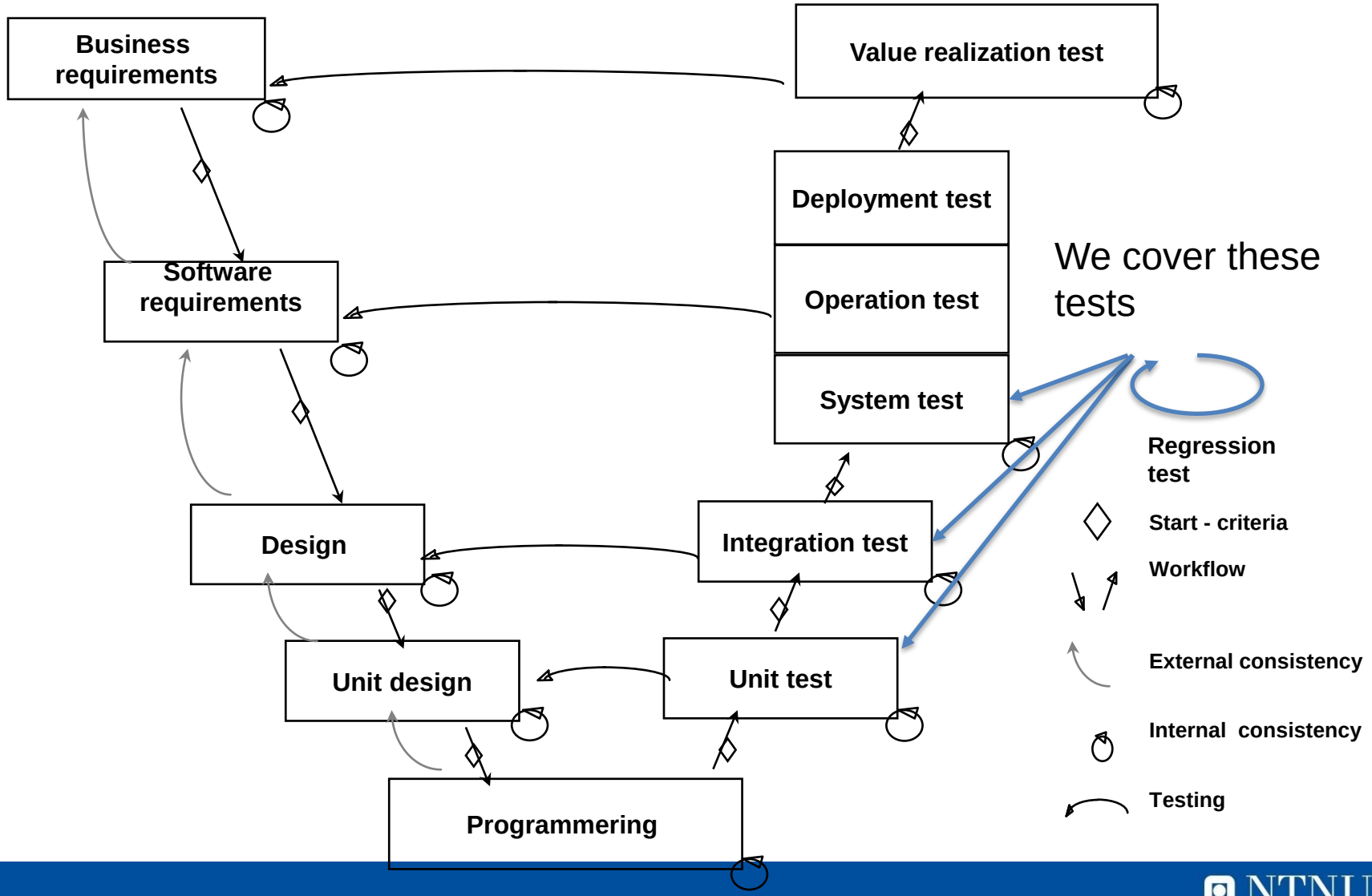
- Use cases
- User stories
- Goal-oriented RE approaches

Characteristics of good software requirements specification*

- Complete
- Unambiguous
- Consistent
- Correct
- Verifiable
- Traceable
- Ranked for importance and/or stability
- Modifiable

* IEEE Std 830-1998

Requirements matched by testing – the V-model



IEC 61508 – Test requirements

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
1	Probabilistic testing	C.5.1	---	R	R	**R**
2	Dynamic analysis and testing	B.6.5 Table B.2	R	HR	HR	**HR**
3	Data recording and analysis	C.5.2	HR	HR	HR	**HR**
4	Functional and black box testing	B.5.1 B.5.2 Table B.3	HR	HR	HR	**HR**
5	Performance testing	Table B.6	R	R	HR	**HR**
6	Model based testing	C.5.27	R	R	HR	**HR**
7	Interface testing	C.5.3	R	R	HR	**HR**
8	Test management and automation tools	C.4.7	R	HR	HR	**HR**
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	R	R	HR	**HR**
10	**Formal verification**	**C.5.12**	**---**	**---**	**R**	**R**

Tentative lecture plan

High-quality and effective testing

Week	Date	Lecture content	Speaker
6	07.02.22	Control flow and data flow testing	Jingyue Li
7	14.02.22	Functional testing with Black Box	Jingyue Li
8	21.02.22	System and acceptance testing and Regression testing	Jingyue Li
9	28.02.22	Testing in Practice	Daniela Cruzes

Tentative lecture plan

High-quality and maintainable code

Week	Date	Lecture content	Speaker
10	07.03.22	Code review and code refactoring	Jingyue Li
11	14.03.22	Technical Debt process in Industry	Guest lecture from Visma

Code review, analysis, and refactoring

Refactoring Code

- Making code better, may not be faster
- Better structured, better built
- More readable, more understandable
- Easy to work with
- Easy to add new features
- Easy to spot and fix bugs
- Keeping code in control
- Improving existing code

Tentative lecture plan

Advanced topics

Week	Date	Lecture content	Speaker
5	31.01.22	DevOps and Git Lab	Nora Thomas from Vipps.no
12	21.03.22	Cost and defect estimation	Anh Nguyen Duc
13	28.03.22	Software startup Software eco-systems	Anh Nguyen Duc

DevOps

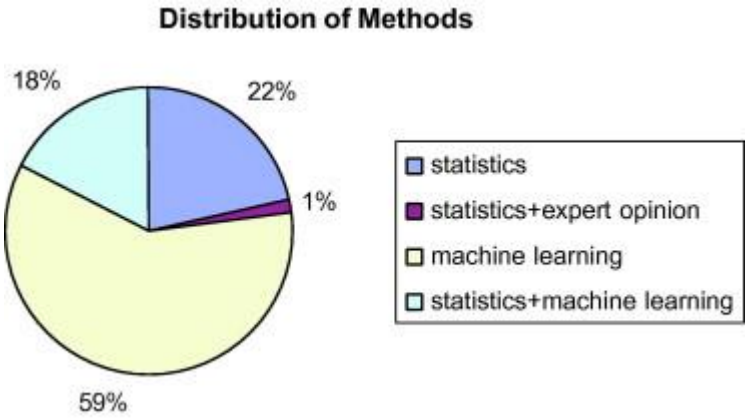
- DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.
- Integrated development and operations
- Supporting quick releases

Cost and defect estimation - Cost

- Why?
  - 55% total number of project runs out of budgets
  - Almost every software project experiences some sort of project delays
  - Estimation is needed for better planning and preventive actions
- Purpose of software estimation
- How
  - Expert judgement: Delphi method, focus group, planning poker, etc
  - Regression-based method: COCOMO models, Putnam model, and function points
  - Simulation-based
  - Machine learning based approach: Bayesian Network, Deep Learning, etc
- Where to apply:
  - Waterfall vs. Agile
  - Within-company vs. cross-company
  - Private vs open source dataset

Cost and defect estimation - Defect

- Defects vs. fault, error, breach, failure
- Software defect can be very costly!
- Defect prediction is a cost-effective to deal with defect
  - Informed static analysis
  - Bug triaging
  - Faulty vs. non-faulty classification
- Approaches
- Metrics:
  - Project-level metrics: project type, budget, duration
  - Team level metrics: engineer's competence
  - Code level metrics: LOC, CK metric suite, code smell, number of functions, number of interfaces, number of class, code complexity, etc
  - Product level metrics: number of component, cohesion, etc
  - Collaboration level metrics: people-to-people interaction, people-to-file interaction
- Advanced defect prediction methods
  - Regression-based approaches
  - Support Vector Machine
  - Machine Learning (supervised/ unsupervised)
  - Decision Trees
  - Naïve Bayes
  - Deep Learning



Software startup

- Airbnb, Uber, Spotify, Skype, Kahoot, AppSumo, Vipps, Mitandbud, etc are used to be startups
- Startup companies are unique:
  - Little or no operating history
  - Limited resources
  - Multiple influences
  - Dynamic technologies and markets
- A temporary organizational state with special way of working, engineering practices and activities can be done different than in established contexts

Software startup

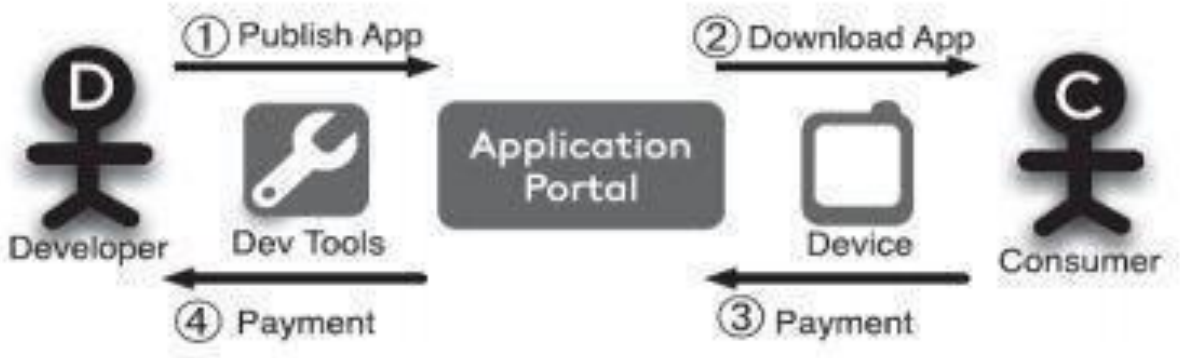
- Understanding failure in startups
- Reasoning about software startup engineering
  - Problem-solution fit and Product-market fit
  - Startup way of working
  - Technical debt
  - Influences of multiple stakeholders
  - Competence and financial situations
- Components of software startups
  - Organization
  - Competence
  - Product
  - Way of working
  - Startup ecosystem

Software startup

- Processes, framework, practices for software startup engineering
  - Lean approach to developing product and businesses
  - Minimum Viable Product
  - Requirement Engineering
  - Minimum Viable User Experience
  - Design thinking
  - Customer journey
  - Growth hacking
- Continuous experimentation in startups
  - Goal-driven hypotheses
  - Build-measure-learn loops
  - Hypothesis driven engineering

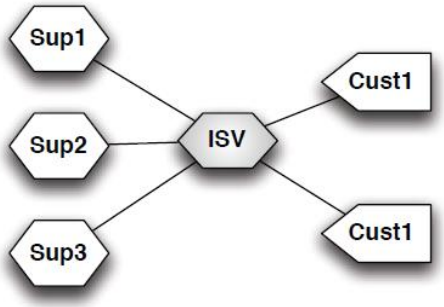
Software ecosystem

- "*a set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them*" Messerschmitt and Szyperski (2003)
- Key elements:
  - A central hub: an owner of the platform
  - A platform: Usually technological platform
  - Niche players: who generate value from the platform
- Example: app stores, open source projects, etc

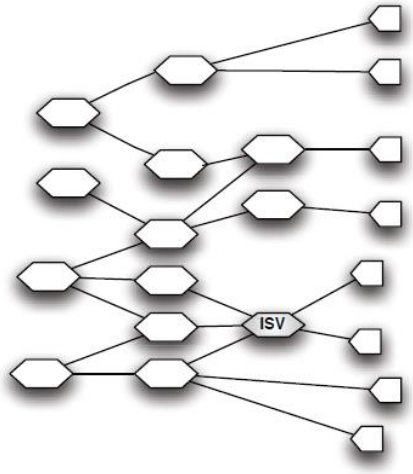


Software ecosystem

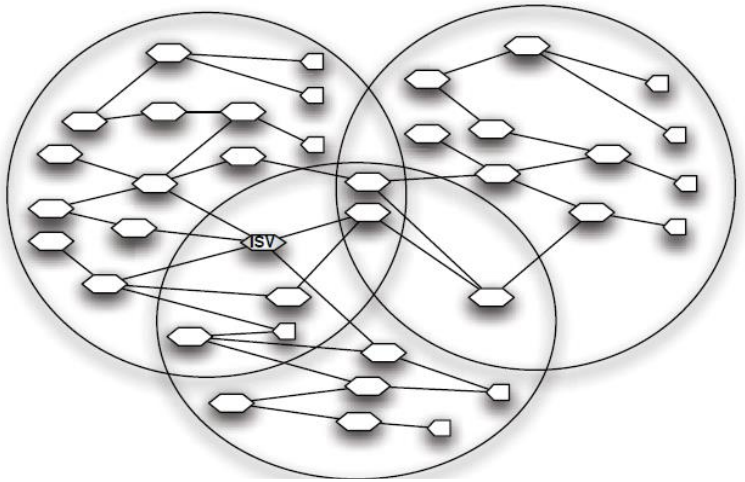
- Scope of ecosystem



A. A hub between suppliers and customers



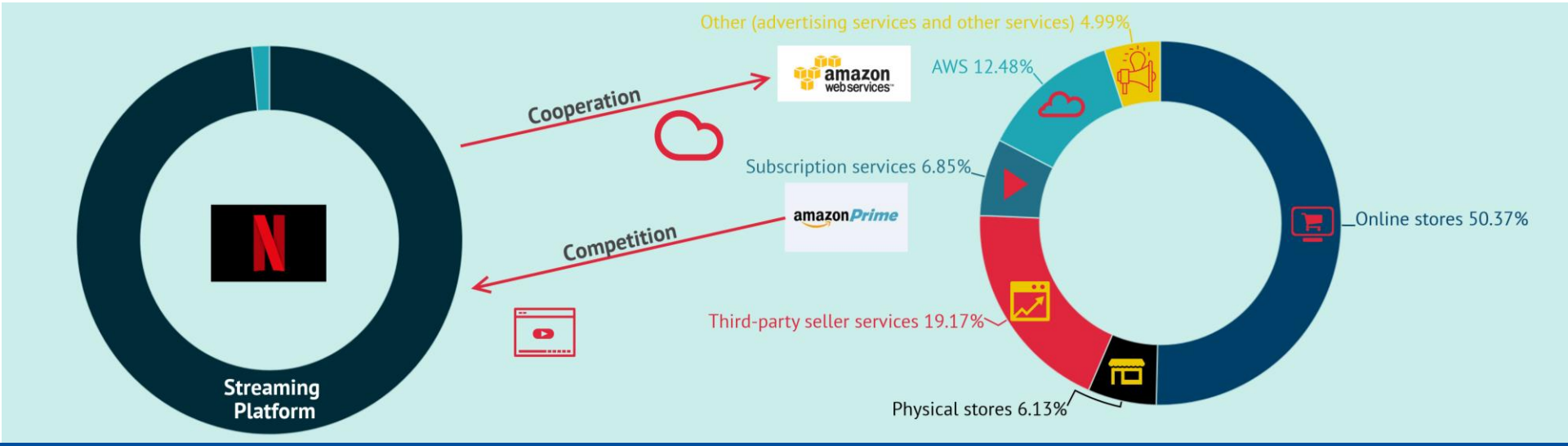
B. The hierachy of hubs



C. Dynamic relationships among hubs, suppliers and customers

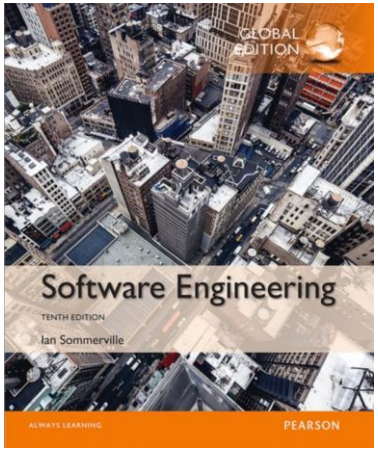
Software ecosystem

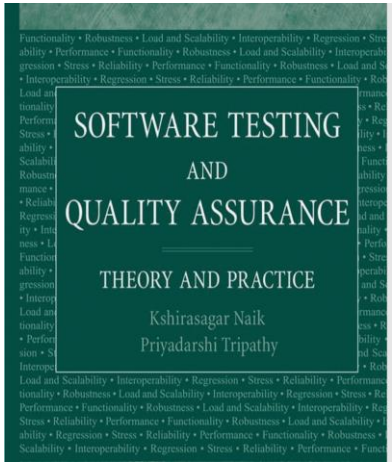
- Scope of ecosystem
- Healthiness of ecosystem
- Collaboration vs. Competition
- Sustainability

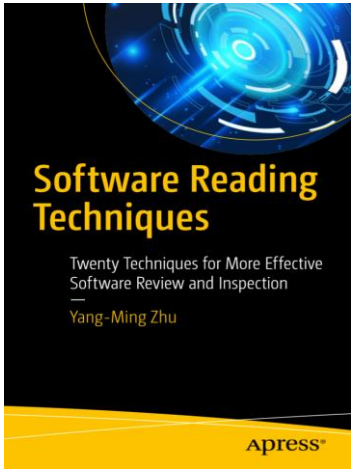


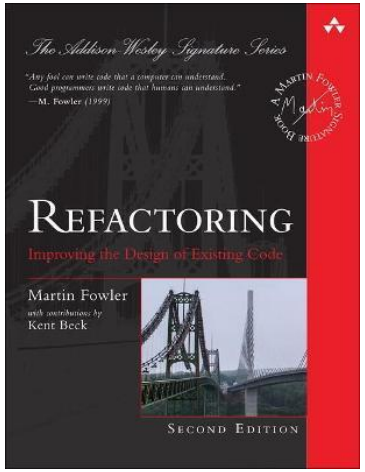
Curriculum

- Journal and conference articles (will be uploaded to blackboard)
- Some chapters from books (accessible in NTNU library)









(1st/2nd edition)

Teaching goals

- Apply goal-oriented requirement engineering method and requirement boilerplate to identify and model requirements;
- Identify and correct typical requirements quality issues;
- Apply different testing and code review, analysis, and refactoring strategies;
- Explain the industrial state of the practice methods of verifying and validating high-assurance software-intensive system;

Teaching goals (Cont')

- Understand the state of the practice agile methods, such as DevOps, and apply corresponding tools;
- Explain the issues and solutions of distributed and collaborative software development and maintenance;
- Explain key issues and solutions to managing and operating large and complex software-intensive systems.

Evaluation and grading

- Exercises: 50%
  - Exercise 1: 10%
  - Exercise 2: 25%
  - Exercise 3: 15%
- Exam (3rd June): 50%
- You have to pass both!
  - You have to hand in **all** the exercises **and** get a passing grade to be allowed to take the exam
  - If you fail the exam, you will fail the course

A: 89–100 points
B: 77–88 points
C: 65–76 points
D: 53–64 points
E: 41–52 points
F: 0–40 points

Exercises	Exam	Final grade
Fail	Fail	Fail
Fail	Pass	Fail
Pass	Fail	Fail
Pass	Pass	Combine exercise and exam grades

Exercise 1

- The purpose is to let students practice
  - Software requirement elicitation methods
  - Evaluating software requirements quality

- You need to
  - Based on an existing medium-sized web application, collect and understand new requirements you get from a peer group
  - Document the new requirements in high quality

Exercise 2

- The purpose is to let students practice
  - DevOps tools
  - Extending existing code
  - Software testing methods
- We will ask you extending the existing system and test the existing code
- You need to
  - Apply DevOps tools
  - Develop new code
  - Run tests manually and write test scripts

Exercise 3

- The purpose is to let students practice
  - Code review
  - Code analysis
  - Code refactoring

- You need to
  - Do manual code review of the existing code
  - Refactor the code based on your review and analysis results

Exercise tasks and schedule

The exercise introduction lectures are digital via Zoom on Tuesday 08:15 AM - 10:00 AM.
The exercise introduction lecture schedule is in the following Table.

ID	Weeks	Exercises schedule		
		Introduction lecture	Start	Deliverable
1	3-7	18 Jan. 08:15 - 10:00	18 Jan.	• Final report of exercise 1. 14 February at 23:59 (Monday), Week 7
2	7-12	15 Feb. 08:15 - 10:00	15 Feb.	• Final report of exercise 2. 21 March at 23:59 (Monday), Week 12
3	12-14	22 March 08:15 - 10:00	22 March	• Final report of exercises 3. 4 April at 23:59 (Monday), Week 14

In other weeks without exercise introduction lectures, we will have TAs online via the same Zoom link to answer questions.

Exercise groups

- 1-2 students in each group
- To form a group
  - Send an e-mail with a list of the names and e-mails of the group members to tdt4242@idi.ntnu.no
- If you don't have a group
  - Send an e-mail to tdt4242@idi.ntnu.no
  - You will be assigned to a group
- Deadline: 1 Feb.

Deadline for exercises

All exercises will have a deadline for delivery.

This deadline may only be exceeded after agreement with the

- course responsible
- teaching assistant (email: tdt4242@idi.ntnu.no)

If no such agreement exists, we will deduct 20% points on the grade for any obligatory exercise for each week it is delayed.

Reference group

- A group of 2-3 students that have a special duty to provide feedback about the course during the semester
- We'll have 2-3 meetings where we can discuss content and form of lectures and assignments
- The group should be formed during the first 3 weeks, so please nominate yourself (or others)
- More information (In Norwegian) → http://www.ntnu.no/utdanningskvalitet/roller.html#Studentene

About you

I need to know a little more about you to adapt my teaching focuses and exercises!