Operating Systems
INF 3151, INF 4151

Administrative Introduction

My own OS

Who is helping you to learn

- Teachers:
  - Otto Anshus, otto@cs.uit.no
  - Vera Goebel, goebel@ifi.uio.no
  - Thomas Plagemann, plageman@ifi.uio.no
  - Guest lectures from Tore Larsen & Knut Omang
- PostDoc & PhD student:
  - Stein Kristiansen, steikr@ifi.uio.no
  - Fabrice Bigirimana, fabriceb@student.matnat.uio.no
- Teaching assistants ("gruppelærer"):
  - Andreas Gansen, andregan@ifi.uio.no
  - Joakin Lindquister, joakimjl@ifi.uio.no
  - Øystein Dale, oystedal@ifi.uio.no
  - Sighurhjörtur Snorrason, sigurhjs@ifi.uio.no
  - Henning Lund-Hanssen, hennilu@ifi.uio.no

2015 New Year New Approaches

2015 New Year New Approaches

- Reason: **resource problems**

- Two major changes:
  - Max number of 50 students
  - Design documents are mandatory, but do contribute to final grade (if all passed)
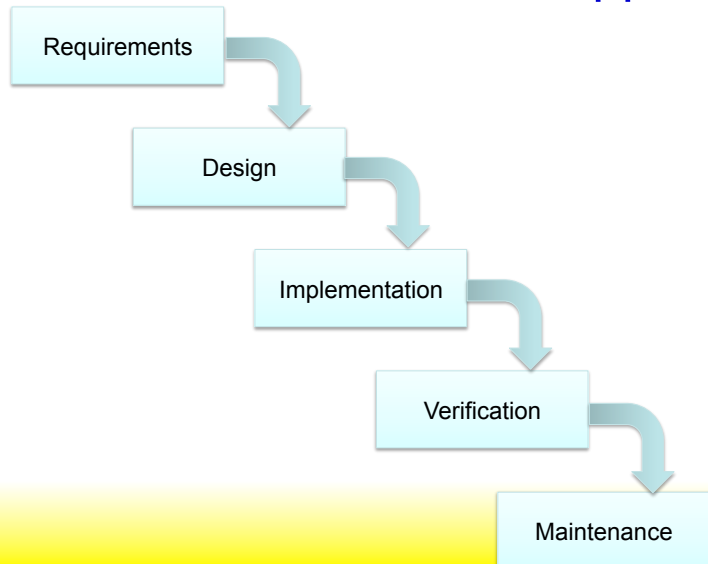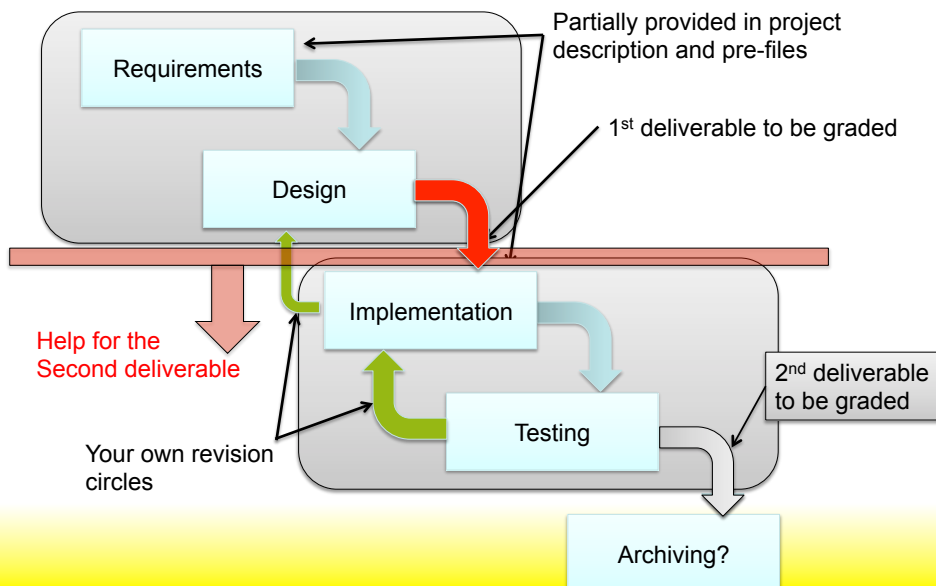
[Source: dagbladet.no]

Learning by doing

- Guided process to build your OS
  - First design! You propose, we give you feedback!
  - Afterwards implementation
  - In total six projects

- Grading based on your presentation
  - Design (one week)
  - Code (two weeks, except P3 which has only 1 week)
  - Deliverables through Devilry:
    - https://devilry.ifi.uio.no/
    - Deadline: Wednesday 12:00 local time SHARP!

Software Development –
The Classical Waterfall Approach

Requirements

Design

Implementation

Verification

Maintenance

Waterfall Approach and INF3151

Requirements

Partially provided in project
description and pre-files

1st deliverable to be graded

Design

Implementation

Help for the
Second deliverable

2nd deliverable
to be graded

Testing

Your own revision
circles

Archiving?

How to write an Effective Design Document
by Scott Hackett
http://blog.slickedit.com/2007/05/how-to-write-an-effective-design-document
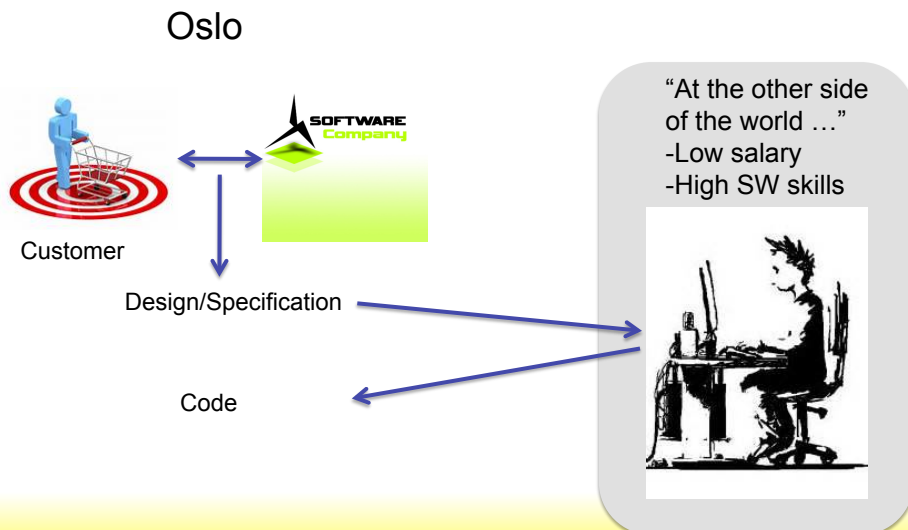
"If you fail to plan, then you plan to fail"

- Scott Hackett's blogg is related to design as part as software development in general
  - ➔ apply this to our setting!
- Why a design document?
  - Explain your design decisions and why they are good
  - Formalism and tools are not the most important – as long as your document conveys your explanations
  - Develop the design document for the audience, i.e., for your team mate and for us (to help you and to grade you)

How to write an Effective Des….. (cont.)

- What makes a good design?
  - It is good if:
    - it fulfills the requirements in a meaningful way
    - all design decisions are justified (give clear reasons)
    - documents benefits of design decisions
  - Diagrams are good and useful, but they must be explained in text!

More on design ….

Oslo

Customer

SOFTWARE Company

"At the other side of the world …"
-Low salary
-High SW skills

Design/Specification

Code

Design as process

- How to solve the assignments, i.e, develop a well working program?
- Read the assignment, study the pre-code, follow carefully the project presentation
- Think and discuss with your group mate
- Identify alternative approaches, e.g, important data structures, algorithms, etc.
- Evaluate the approaches and select the best
- Document the main results of this process in your design proposal

Design proposal

- Mandatory deliverable for each project
- Not more than 10 pages!
- Description of your plan of how to solve the problem and why in this way
- Answer all questions in the assignment and raised during the project presentation
- Typically this document contains:
  - Brief description of the different **alternatives** you have studied and why you selected which.
  - **Detailed textual description** of the proposed **data structures** and **algorithm(s)** to be used with supporting illustrations in form of **figures**, **flow diagrams**, or **pseudo code**. If you use standard data structures or algorithms, put your main emphasis on how they are applied to the problem.
  - Description of what functionality will be implemented in which file/function, and how these implemented parts will interact (will they work exclusively? Concurrent? Can they be interrupted? etc.) to attain the goal given in the problem description.
  - **Key details** such as why a particular mask value is chosen, how it is constructed, how and why a particular register is loaded with a particular value, etc.
- You will also get hints during the presentation of the project on what should be addressed in the design proposal ➔ Pay attention!

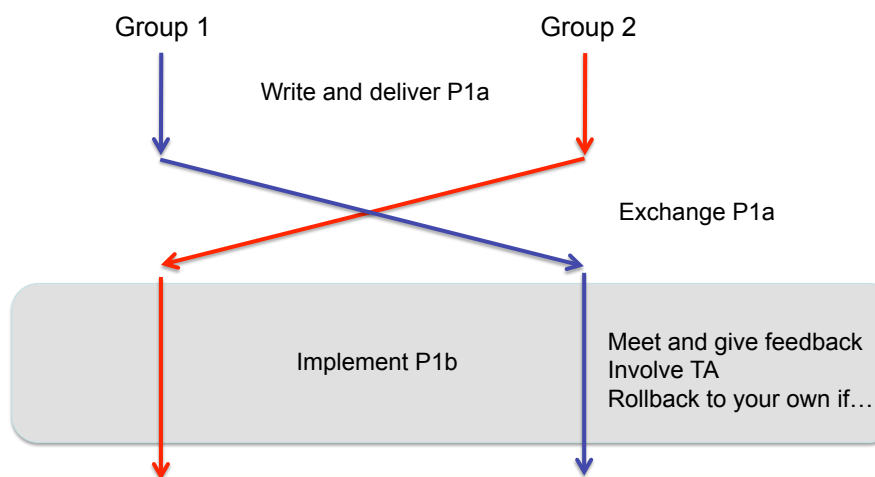# What do we do with the design proposal?

- Learn to make a design
- Give you early feedback
  - Oral presentation of your design proposal to your TA
  - TA gives you feedback whether you are on the right track or not
    - → saves you a lot of time
    - → helps you to get a better grade
  - However, the feedback addresses only what you have taken up in the design document
    -> the better the design the more specific the feedback (and vice versa)
- If not passed, you are out of the course

Design – The "*Smart*" Approach in the previous years

If I implement first everything and it works I can write afterwards the perfect design and get an "A"

Let's play outsourcing in P1

Group 1   Group 2

Write and deliver P1a

Exchange P1a

Implement P1b

Meet and give feedback
Involve TA
Rollback to your own if…

Grading of Exercises

- All TAs will give the same amount of support for students
Read: we help you to learn, but not to make shortcuts!
- (Additional help for "desperate" students might be reflected in the grade)

Grading of Exercises

Project	Grading
P0	none
P1a	passed / not passed
P1b	passed / not passed
P2a	passed / not passed
P2b	A - F
P3a	passed / not passed
P3b	A - F
P4a	passed / not passed
P4b	A - F
P5a	passed / not passed
P5b	A - F
P6a	passed / not passed
P7b	A - F

- Each mandatory deliverable is graded by a TA
- Each A – F deliverable is graded by a PhD student
- External censor controls randomly
- At the end of the term all grades are combined and eventually adapted

Group Lectures

- Class room (2 hrs. per week):
  - Catching up background
  - Diving into technical depth, e.g., how to …
  - Be active – ask & discuss!!
- Terminal room (2 hrs. per week):
  - TAs are always available for questions etc.
- Mandatory design review (one per project per team):
  - Present your design to the TA
  - Get feedback
- Approx. 20 students per TA
- Deliverables have to be prepared by teams of two students
- Teams of three students are not allowed because of grading
- Initial task: each team sends to Kristoffer Robin Stokke their user names for registration in Delivry

Exception Handling - I

- Sick leave:
  - Official certificate from a medical doctor
  - Oral examination about the missed deliverable
- Disagreement in a group:
  - Oral examination
- Cheating / fraud:
  - According to rules of Faculty of Mathematical and Natural Sciences
  - The declaration you sign is just to make you aware of the existing rules

The Big NoNo

- It is not allowed to distributed code from the assignments or to make it accessible to others (except lecturers and teaching assistants of the course) neither in paper form nor in electronic form. This is valid for the code developed from the students as well as code distributed as part of the assignments.
- All contraventions are regarded as fraud!
- You have to sign a corresponding declaration and deliver it in the next group lecture to your TA
    - Without it you will not get access to the next assignment!

The Big YesYes

- Start to work hard right from the beginning
- Be active in the group lectures
- Discuss with your partner
- Discuss with other students (but do not exchange code or the answers to the theory assignments!)

- Solving problems and understanding an OS can be a lot of fun!

Exercises "this week"

- Kick start into assembler and C
  - Work on a "simple" challenge
  - Strongly related to future challenges in the course
- You learn also about the programming environment to be used
- Start group work …
- When:
  - Friday 12:15-14:00
  - Manday 10:15-12:00, 14:15-16:00
  - Tuesday 14:15-16:00
  - Wednesday 08:15-10:00

Questions?

- Talk to us
- Take a look at the FAQ