Software Engineering I (02161)

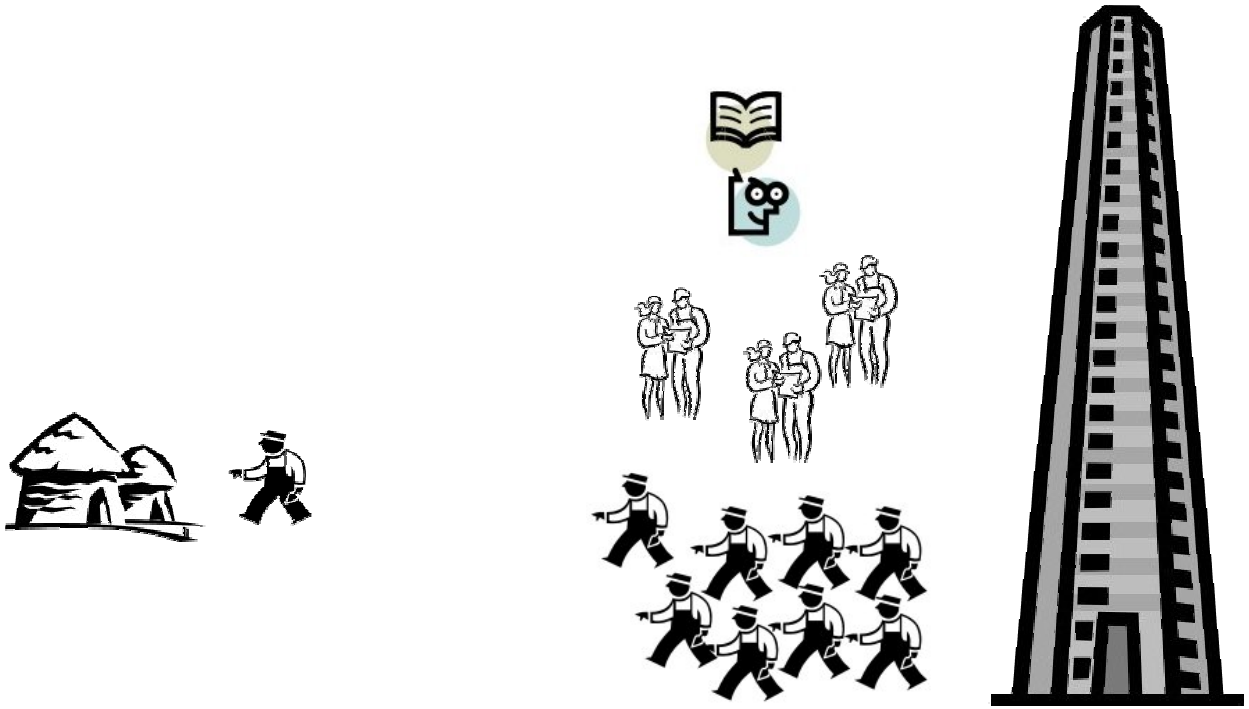
Week 8: Software Developement Process, Version Control, Introduction to the Project

Hubert Baumeister

Informatics and Mathematical Modelling
Technical University of Denmark

Spring 2010

Software Development Challenges

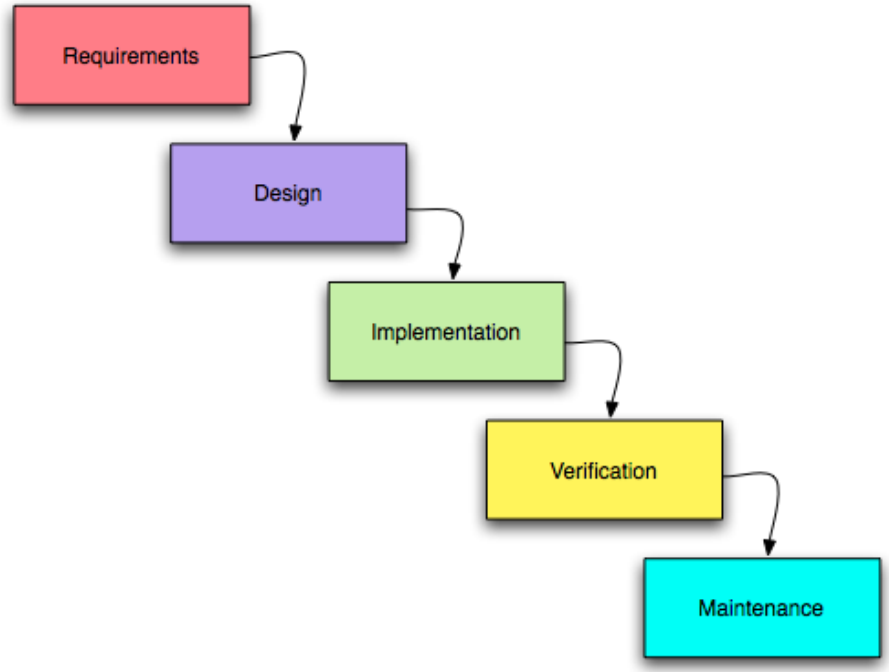


- Challenges of Software Devevlopment
  - on time
  - In budget
  - No defects
  - Customer is satisfaction

Software Development Process

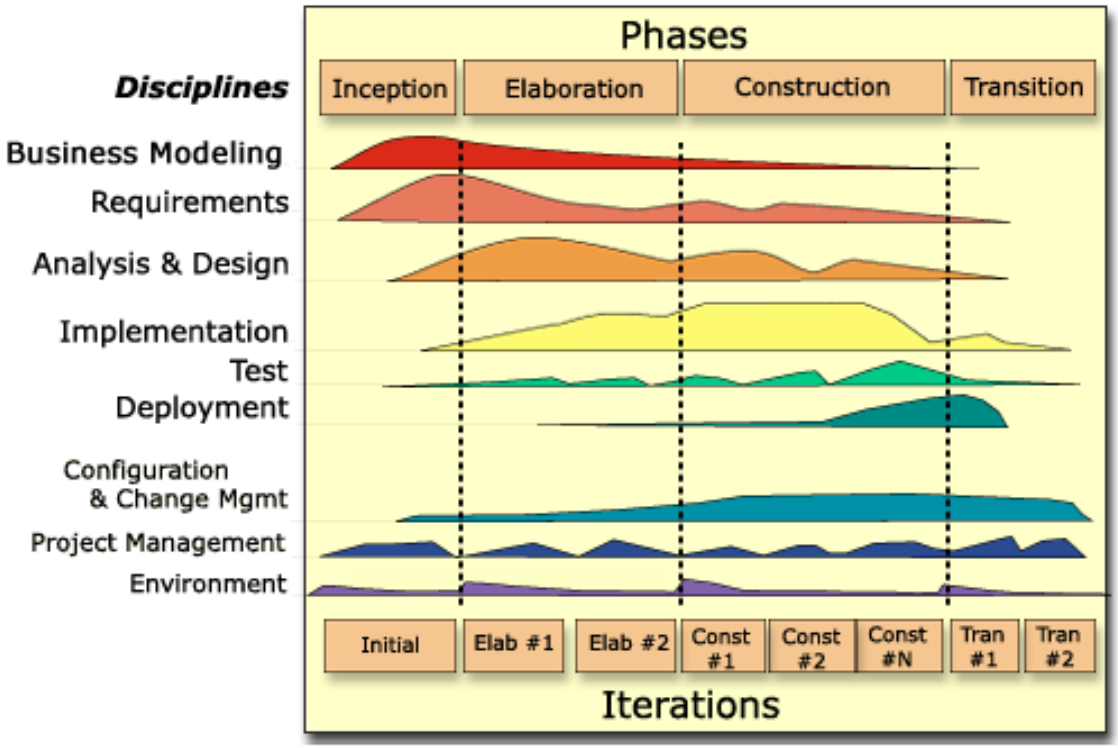
- Activities in Software Developement
  - Understand and document what kind of the software the customer wants
    → Requirements Analysis
  - Determine how the software is to be built
    → Design
  - Build the software
    → Implementation
  - Validate that the software solves the customers problem
    → Testing
    → Verification
    → Evaluation: e.g. User friendlieness
- Each of the steps has its associated set of techniques
- However, the techniques can be applied in different orders
  → Different software development processes
  - e.g. Waterfall and Iterative processes (e.g. Rational Unified Process (RUP), agile processes: Extreme Programming (XP), SCRUM, Feature Driven Development, Lean . . .)

Waterfall process



- Strict waterfall: An activity has to terminate before the next activity begins
  - → No feedback possible from the later activities
  - → Takes too long time for the system to be build, which does not allow the customer to give feedback

Iterative Processes: E.g. Rational Unified Process



- Inception, Elaboration, Construction, Transition corresponds to Plan the project, understand the problem, build the solution, test the solution, maintain the solution
  - All activities occur throughout the project
  - After each iteration, the customer sees the product and gives feedback

Agile Processes and Lean Software Development

- eXtreme Programming (XP), Scrum, Feature Driven Development (FDD), Lean Software Development
- Iterations getting shorter than with, e.g., RUP
- New set of techniques: Pair programming, customer on site, user stories, test-first and test-driven development, . . .
- Focus on marketable features
  - **A feature of the software that is relevant for the customer**
  - User stories (XP), Backlog (Scrum), Features (FDD), ...
  - Corresponds roughly to use case scenarios
- Lean Software Development
  - Apply values and principles from Lean Production to Software Development

What is Lean Production?

- History
  - The Toyota Way (set of management principles)
  - Toyota Production System (set of production principles)
  - Womack and Jones in the 1980's
    - Too books introducing Lean to the western world
    - *The Machine that Changed the World* and *Lean Thinking*

Goals

1. Produce value for the customer
2. Remove waste
3. Create flow

- Results
  1. Shorter production time
  2. Better products
  3. Less expensive products

Lean Principles

1. Identify customer value
2. Identify the value stream
3. Create flow
4. Pull
5. Perfection

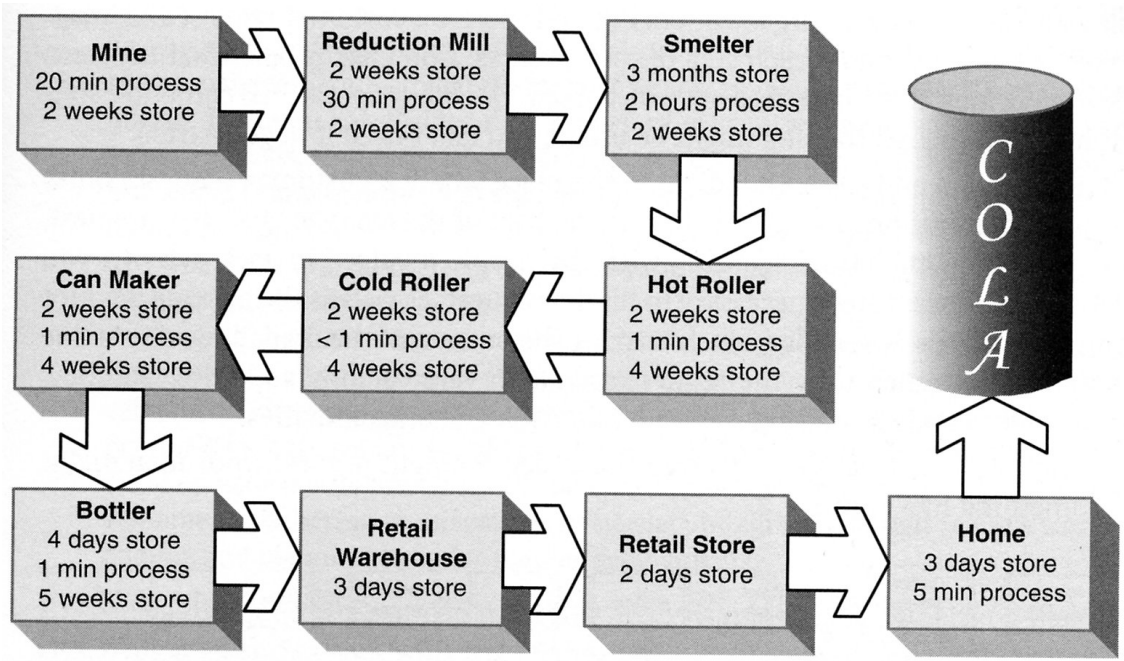
Lean Principle: Identify value

Def. Waste

Anything that absorbs resources but does not produce value

- Getting and keeping in touch with customer value is essential
  - → Requirements Engineering
    - Use Cases
    - Domain model
    - Glossary
    - ...
  - → XP practice: Customer on site

Identify Waste: Value Stream Map



- It takes in average 319 days to produce a Cola can while processing time is 3 hours (= 0.004%)

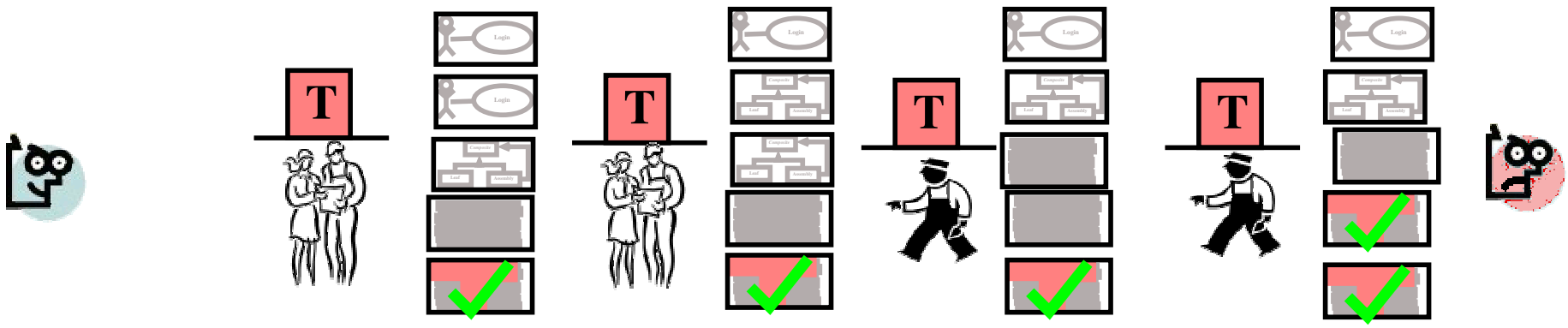
Figure from Lean Software Development by Mary and Tom Poppendieck, 2003

Types of Muda (Waste) in Software Systems

1. Defects in products that require rectification
2. Extra features (features the customer did not ask for)
3. Partially done work / technical debt
4. Unnecessary activities
5. Unnecessary movement of people (e.g. to find information)
6. Unnecessary hand offs / Task switching
7. Waiting by employees for tools or upstream activities
8. Software which does not meet the needs of the customer

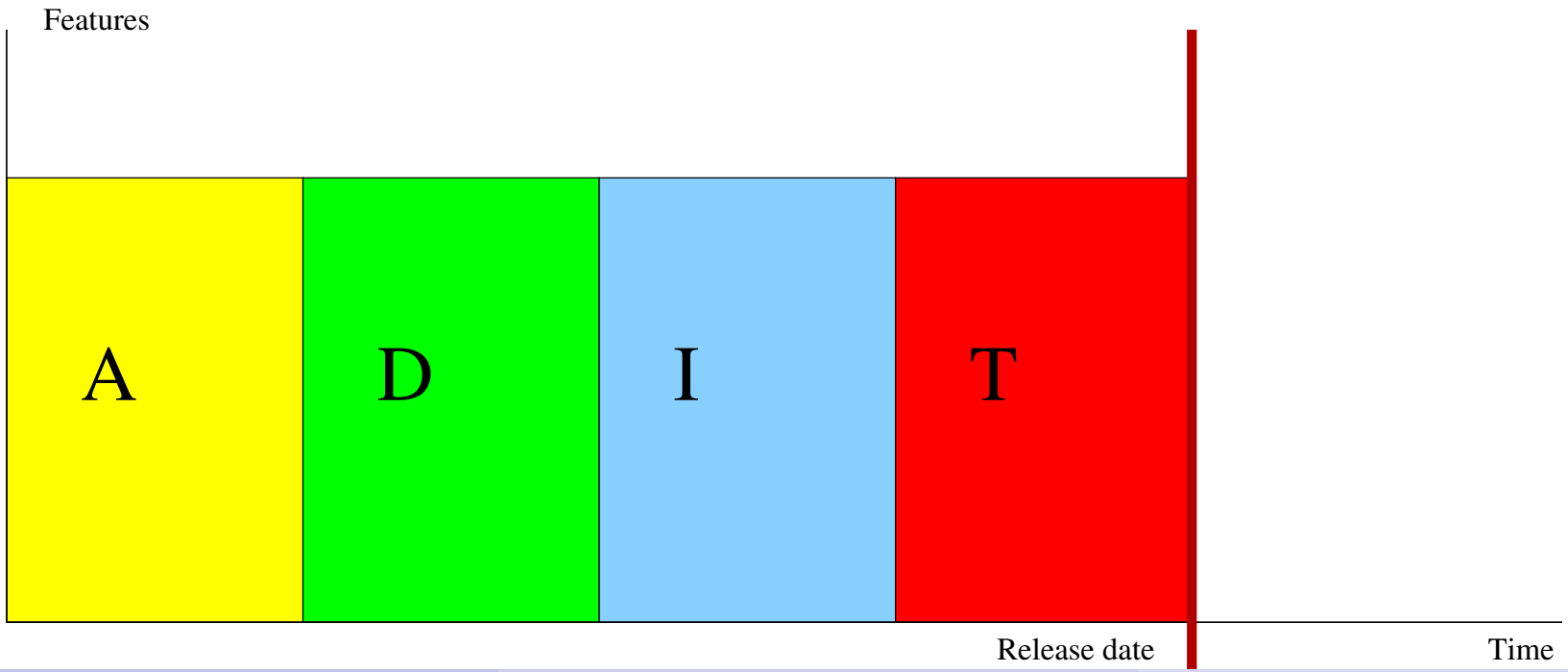
→ Use these waste categories and look for them in your processes

Traditional Software Engineering (waterfall): Mass Production



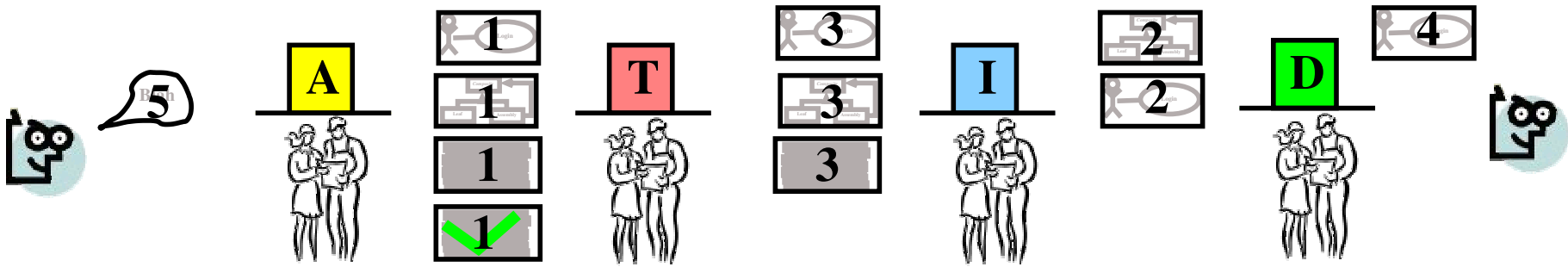
Traditional Software Engineering: Summary

- First features implemented at the same time when all features are implemented
  - = time for each feature * number of features / resources
  - → No feedback
  - → Iterative development improves on that (depends on the number of iterations)
- Delay in one phase of the project delays the whole project



Generating flow using Pull and Kanban

Work Item	A Queue	A WIP	D Queue	D WIP	I Queue	I WIP	T Queue	T WIP	Done
		5		4		2		3	1



Software Engineering: Flow through Pull with Kanban

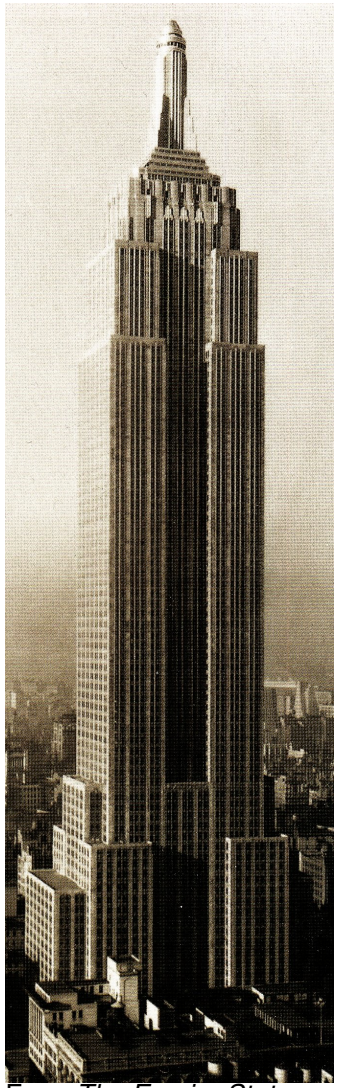
- Traditional Software Engineering: Summary
  - First features ready after all features are ready
    - = time for each feature * number of features / resources
- Kanban Development
  - First feature ready after 4 days
    - = time per feature
    - almost immediate feedback
  - All features implemented after 8 days
    - ≈ time per feature * features / resources
  - More robust against delay
    - Focus on the marketable features with highest priority

Figure from David Anderson `www.agilemanagement.net`

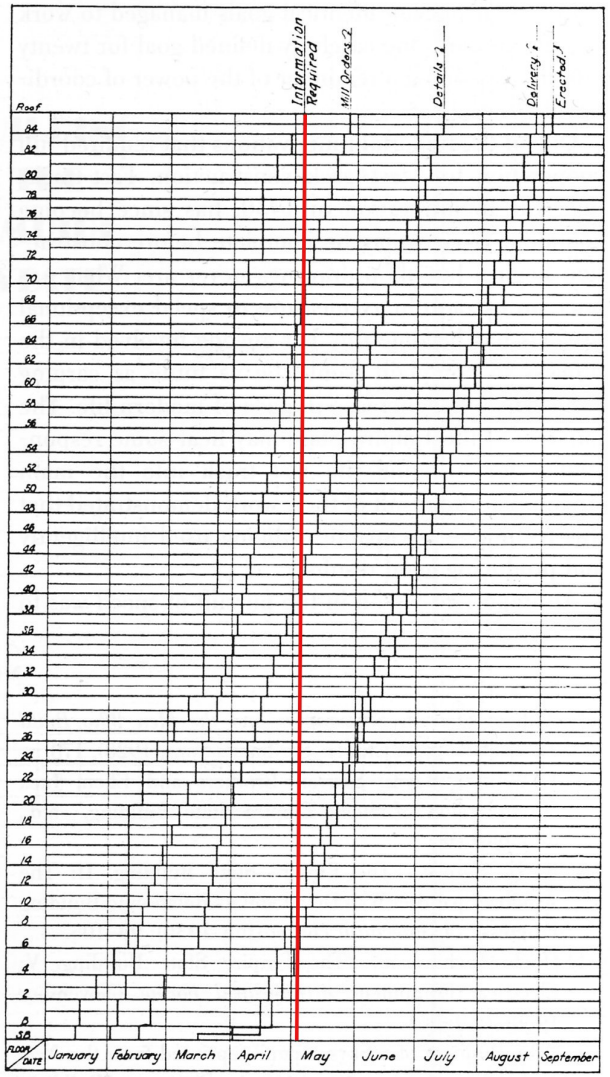
Advantage Lean/Kanban Development

- Process controlling: local rules
- Load balancing: Kanban cards and Work in Progress (WIP) limits
- Early feed back
  - Customer
  - and process
- Assignment of work flexible
  - Traditional teams
  - XP teams
  - Load balancing

Example: Empire State Steel Construction



From *The Empire State Building* by John Tauranac



From Building the Empire State by Willis, 1998

- Kept the budget
- Was finished before deadline
- Built in 21 month (from conception to finished building) (1931)
  - → Basic design in 4 weeks
- Fast-track construction
  - → Begin the construction before the design is complete
  - → create a flow

Planning your project

- Questions to be answered by the planning process
  - How many person hours does a project need
  - How much time does a project need
  - What are the additional resources: e.g. hardware, software, person with certain qualifications (e.g. graphic designer, . . .)
  - When to do what
- Base for the planning process
  - Overview over the functional requirements: Use cases more or less detailed described
  - Overview over the intended architecture: e.g. Web application, stand-alone application etc.
- In your case: resources are fixed; adjust the functionality of the system

  → When to do what

Techniques for planning your project 1

- Step 1 Determine a set of scenarios (e.g. based on Use Case scenarios) that your system should be able to do
  - Do a brain storming on the requirements (use cases)
    - What are the scenarios? (success, failure, . . .)
    - Is the set of use cases complete?
- Step 2 Do a brain storming on the intended architecture of the system (usually, the customer has some requirements here: e.g. implemented as a Web application . . .
  - Only a rough idea is needed

Techniques for planning your project 2

- Step 3 Estimate the Use Case Scenarios
  - How long, in ideal man hours, do you think you need for implementing the use case scenario?
  - Multiply this with a load factor of 2 to get the real man hours
  - This estimation includes
    - Drawing all the diagrams
    - Documenting the use case scenarios involved
    - Designing (class diagrams, sequence diagrams)
    - Implementing
    - Testing
    - Updating the report
    - ...

Techniques for planning your project 3

- Step 4: Count how many resources you have:
  - E.g. 5 weeks * 8 h = 40 person hours per person times
  - 2—4 persons corresponnds to 80—160 person hours per team
- Step 5 Order the use case scenarios by their value to the customer (In real life this is something the customer needs to do!!!)
  - → Add the time for the scenarios until the time reaches the available time
- The result is an initial plan
  - → The plan needs to be updated as the project proceeds

Techniques for planning your project: Remarks

- The planning should include the writing of the report!
- Plan need not be perfect!
  - Don't spent too much time
  - Experience with the problem and its implementation changes the plan
  - Plan needs to be updated every iteration

What is version control?

Version Control

”Revision control (also known as version control, source control or (source) code management (SCM)) is the management of multiple revisions of the same unit of information” Wikipedia

- Stores versions of a file (e.g. a source file)
- Allows to retrieve old versions
- Allows to compare different versions
- Allows to merge different versions (e.g. make one file from two different versions of a file)

→ Is used in projects to

- for concurrent development of software
  - → each programmer works on his version of the file: The results need to be merged

CVS

CVS

Concurrent Versions System

- Originally a set of command line tools
  - → But there exist ”nicer” interfaces: e.g. Eclipse
- A set of files and each file has a tree of ”versions”
  - In principle each file is treated separately from each other
  - → use tagging to indicate that a set of files belong together to, e.g. form a version/release of a software package
  - → branching allows to have parallel versions
- Implemented by storing the differences between the file versions (and not whole files)
- CVS stores its file in a central repository

What are the use cases of version control / CVS?

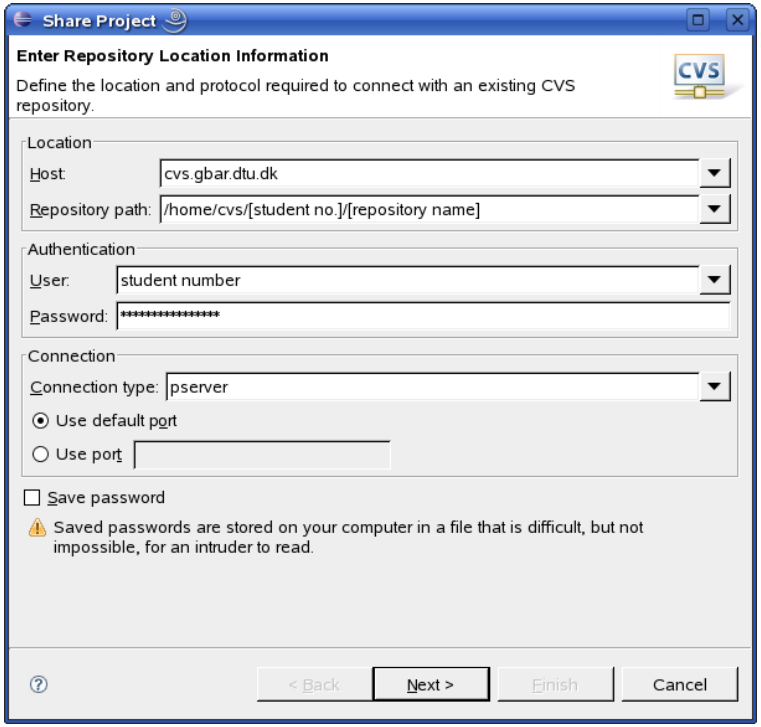
- Creating a CVS repository
- Creating a project within a CVS repository
- Checking out a project from a CVS repository
- Updating a file from a CVS repository
  - Comparing with previous versions
  - Automatically merging changes (note: only files with the ASCII attribute can be merged automatically)
- Committing changes
  - → fails if someone has changed the repository file
  - → requires to to an update, fixing all the conflicts, and then committing again
- Tagging versions
- Branching a version
- Merging a branch

Creating a repository

1. Go to `http://cvs.gbar.dtu.dk`
2. Login using students number and password.
3. Select ”create new repository”
4. Choose a name, eg. 02161
5. Click on the newly generated repository and add the other student numbers from the group with the button ”Add CVS user from DTU”

Creating a project within a CVS repository

- From within Eclipse, select a project in the package explorer and then choose Team→share project and `create a new repository location`
- Fill out the form

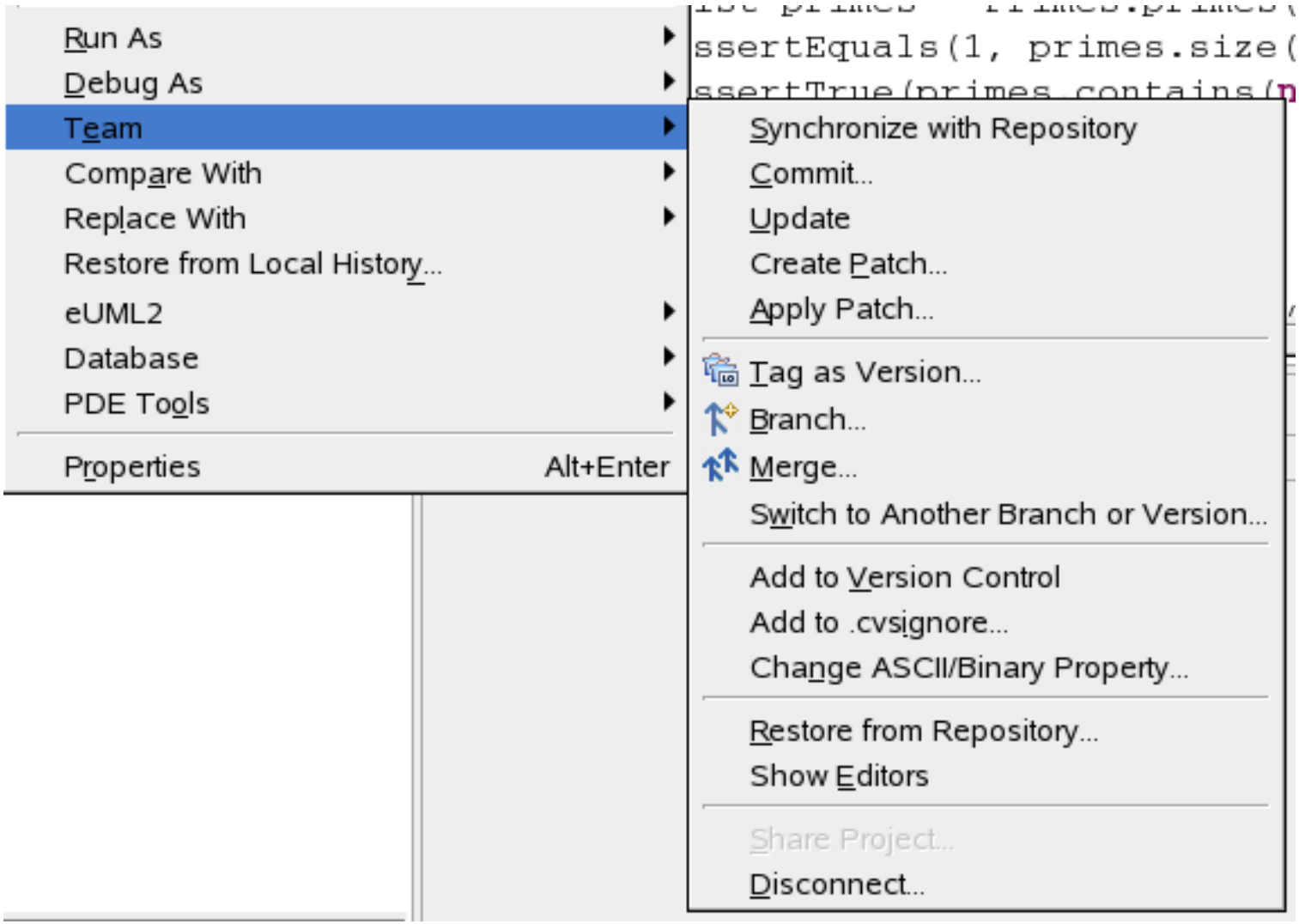


- Click next, mark ”Use project name as module name”, click next and finish

Checking out a project from a CVS repository

- Open the ”CVS Repository Exploring” perspective (Window→open perspective→other
- If not present, create a new repository location selecting new→repository location in the right button menu
- Open the repository location and then HEAD to get to the projects for that location (use Branches and Versions to get to project branches and project versions)
- Right click and then check out the project. You can use as project name a new name or the name of the project in the CVS repository

Package Explorer Team Menu Project

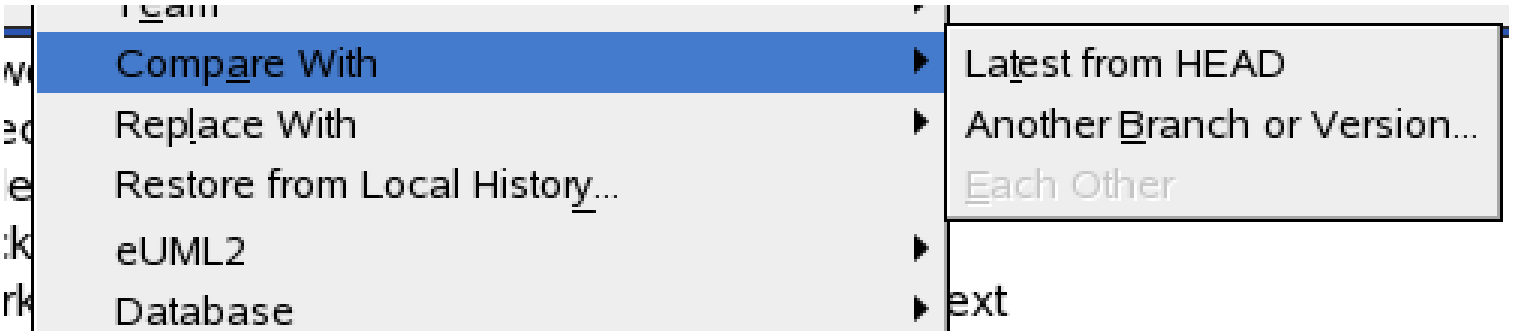


Update a project from a CVS repository

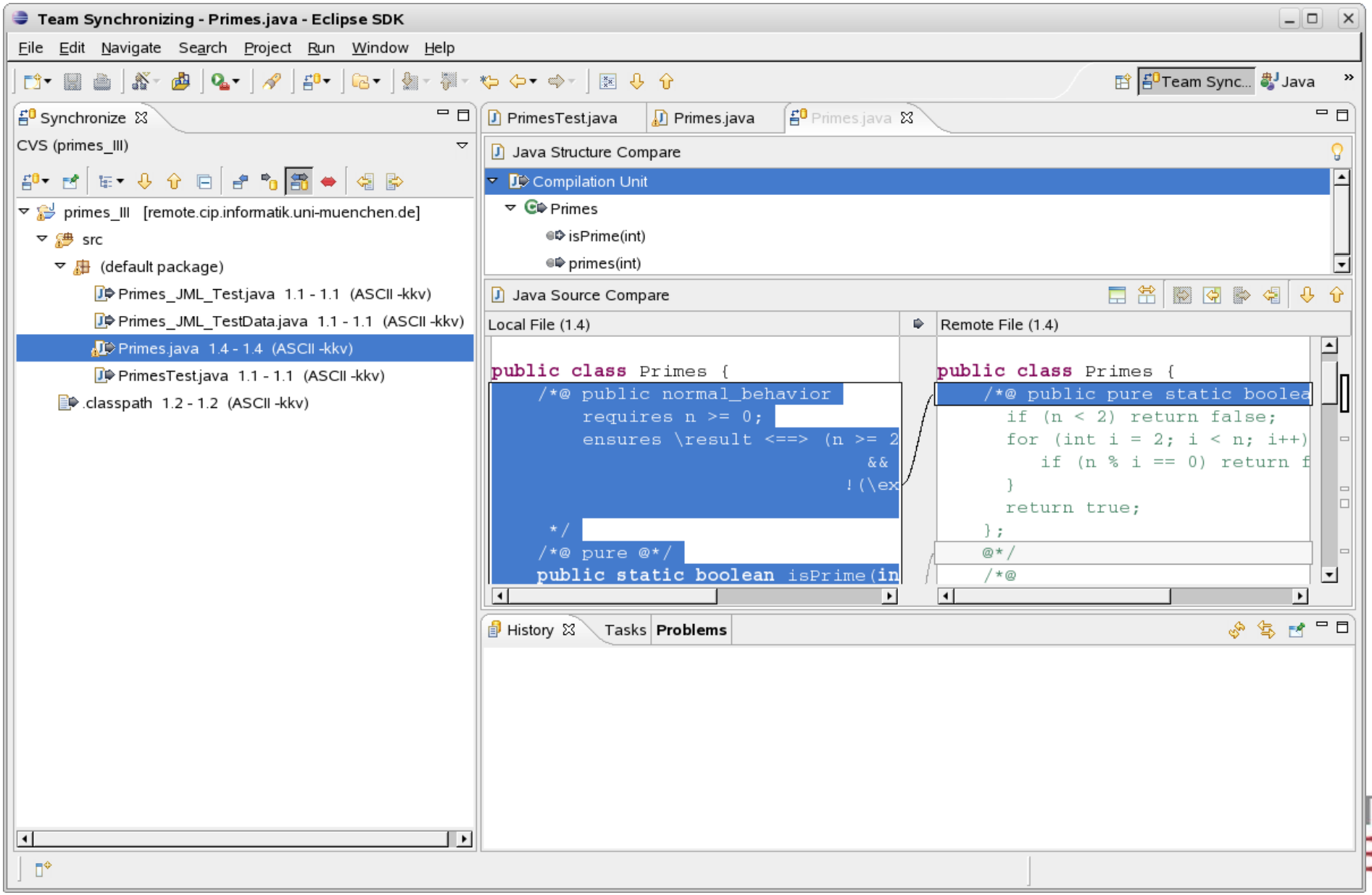
Copies all the changes which are in the repository to the current version of the local files

- If the local files have not been modified after the last update / check out, the local files are overwritten
- If the local files are modified, then they are merged
  - → Merging happens only for files marked with the ASCII property; Other files will be overwritten and the local files will be copied to a different name
    - Use the team menu to change the ASCII/Binary property
  - → Merging might fail. Then the local file will contain both versions, the repository and the local version
    - → Use the compare with menu to check for conflicts

Package Explorer Compare With Menu



Compare result: Compare with latest from HEAD



Committing changes to a CVS repository

- Use commit from the team menu
- You are required to give a comment
- Commit fails if some else committed changes after your last update
  - → Resolve this by updating, repairing any conflicts, and then committing again
    - A good idea is to do an update before each commit

Steps in Developing a Program using CVS

1. Create Repository
2. Create a project within a repository
3. For all the programming tasks in an iteration
   - 3.1 Update the files / directory you will be working on
   - 3.2 Work on the implementation so that all tests run
   - 3.3 Commit your changes
     - 3.3.1 Update the project
     - 3.3.2 Fix all compile time errors and all broken tests;
       If fixing took longer, repeat from step 3.3.1
     - 3.3.3 Commit your changes
4. Tag you files for major project milestones

Introduction to the project

- What is the problem?
  - Project planning and time recording system
- What is the task?
  - Create a
    - Project plan
    - Requirement specification
    - Programdesign
    - Implementation
    - Tests
- Deliver a
  - report describing the requirement specification, design, and implementation (as a paper copy and PDF uploaded to CampusNet)
  - an Eclipse/NetBeans project containing the source code, the tests, and the running program (uploaded to CampusNet as a ZIP file)

Organisational issues

- Groups with 2, 3, or 4 students
- Report can be written in Danish or English
- Program written in Java and tests use JUnit
- On Monday, May 10 there will be a short (10min) demonstration of the program in the E-databar
  - → At least the tests need to be demonstrated
- Report and Eclipse/NetBeans project is to be delivered and uploaded during the demonstrations on May 10
- Each section, diagram, etc. should name the author who made the section, diagram, etc.

Organisational issues

- **You can talk with other groups (or previous students that have taken the course) on the assignment, but it is not allowed to copy from others parts of the report or the program.**
  - *Any text copy without naming the sources is viewed as cheating*
- There will be a CampusNet group created for each project group
- Latest Friday 26.3 18:00 each project group has to have put the project plan on the CampusNet

Exercises

- Last lecture April 12 (on principles for good design and patterns)
- No lectures after April 12
  - Instead the exercises are moved to 13:00-15:00 so that you can get technical help
- There will be exercises today and April 12 from 15:00-17:00 so that you can ask questions regarding the project
- Consultation hours reqarding any problems with the project description will be Thursday April 8 from 15:00-17:00
- Alternatively you can send an e-mail to `hub@imm.dtu.dk`