Software Development Overview

Joachim von Hacht

Software Development Is ...

- ... often **very complex!**
- ... a young engineering discipline
  - Somewhat of an art
- ...in between very informal (dynamic/chaotic)
- ...short of mathematical tools (formulas)
- ...normally a group task
- ...highly dependent on communication

Software Development Process

- To handle the complexity of software development a **software development process** is used
  - Opposite: Ad hoc, "Happy hacking"
- A software development process is a framework that is used to structure, plan, and control the process of developing a program (system).
- ...but no guarantees!
  - There's no **"Silver Bullet"!**

Process Philosophies

- Big design up front (BDUF), heavy process
  - Everything is specified before starting to implement (the traditional engineering approach)
  - Pros: Efficient in general ... but software seems different?!
  - Cons: Hard to handle changes (specification obsolete before we begin to implement)
- **Agile** development, lightweight process
  - Start with a rather preliminary specification.
    Implement **iteratively** (in small steps) and learn
  - Pros: Quick adaption to changes/problems
  - Cons: Insufficient design and documentation (missing general aspects of the problem)
- … and many others
  - See Wikipedia for a list
  - Latest hype: Scrum

Philosophy In Course

- We use a simplified agile (iterative) process
  - We should have **some** understanding (and some beginning of a solution) before the implementation starts
  - We start to implement a few selected parts ...
  - ... thereby gaining deeper understanding ...
  - ... as a result we update the solution ...
  - ... we refactor the code to reflect the new solution ...
  - ... then we continue with a few more part...
  - ...
  - ... until finished!

Process Phases

- All processes are composed of a number of steps
  - We say **"phases"**
- Phase have input and and output
  - Commonly: Output from phase n, input to n+1

Process Phases in Course

The process has 4 phases.

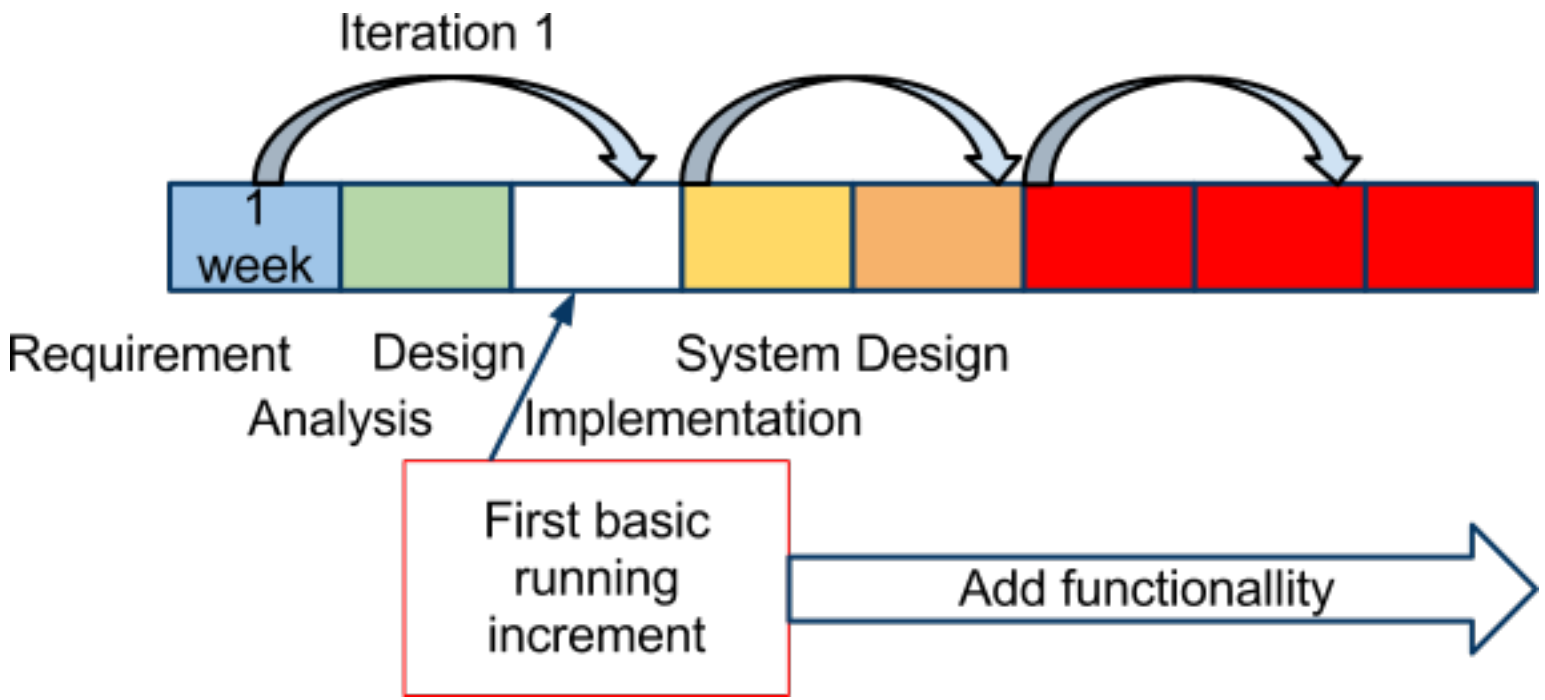
1. Requirement Elicitation
   - What are we going to build?
2. Analysis
   - Build an model of **it**
3. Design
   - Adapt (parts of) **it** so that **it** can be implemented
   - Add supporting systems for **it** (system design)
4. Implementation
   - Implement (part of) **it,** test **it ...**

This has very little with computers and programming to do

Iterations

Each iteration runs through the phases

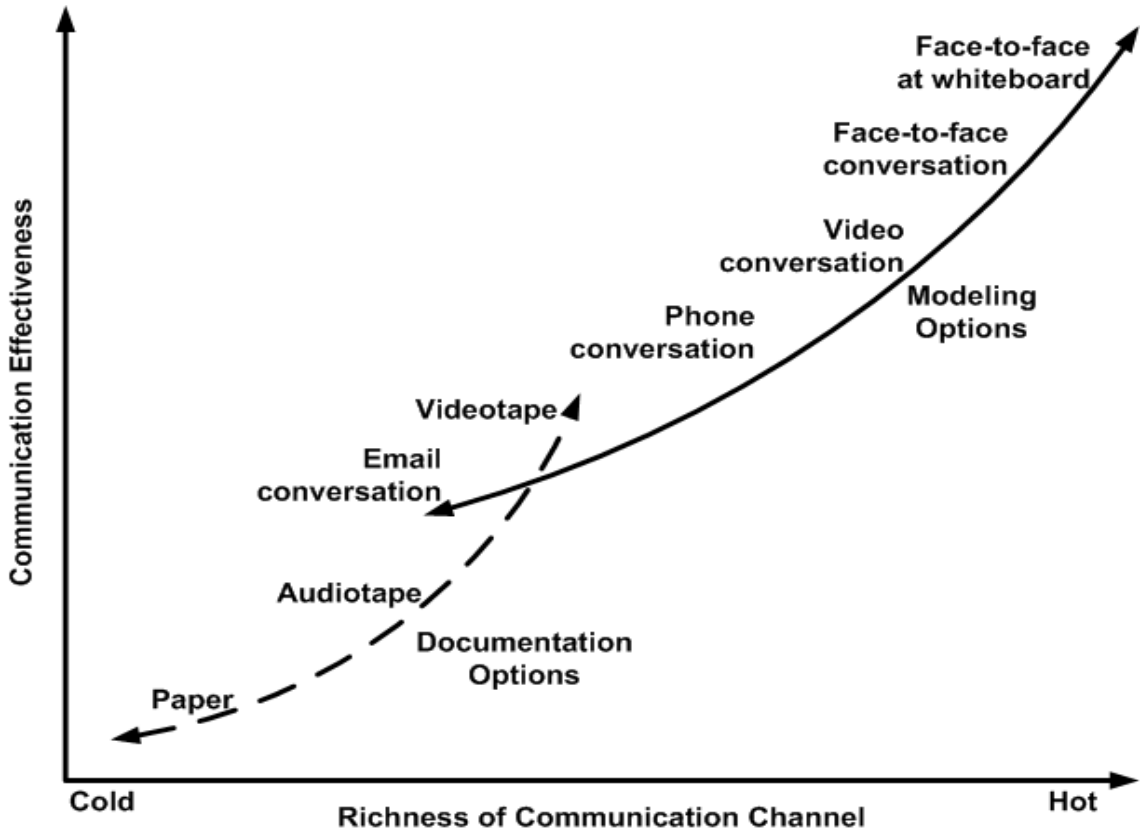
Time Plan



- Must have something to run at week 3!

Software Development and Communication

- Effective communication is a **fundamental** requirement for software development.



Communication in Course

- Find a room with a whiteboard and gather
  - Don't spread the group!
- Use **issue trackers**, can't remember everything...
  - Possible use //TODO in Eclipse
  - Better: Google code (or similar)

Process Documentation

- BDUF processes often implies a lot of documentation
  - Has gained much criticism
  - Code and documentation not synchronized
- Agile methods less of documentation
  - Working code is the ultimate documentation
  - Together with tests, more to come...

Process Documentation in Course

- The group meetings (see Course PM)
  - There should be an agenda and and an **documented** outcome!!
    - It's part of the working process
    - Being a chairman is a qualified task (rotate)
    - Be efficient!...socialize after the meeting...
    - Follow up ... !
- For us to be able to trace the project you have to handle in the the meeting agendas (in e-form, no papers)
  - We'll also use the version control history to be able to trace individual contributions

Software Documentation

- Undocumented software is at best a pain... at worse ... (we say no more in this course, you'll understand sooner or later...)
  - Documenting is hard...

- Optimal: Self documenting code
  - Should be possible to read "as a book"
  - A book has chapters, paragraphs, sentences

Software Documentation in Course

- We require two documents (besides the code)
  - The "Requirement and Analysis Document" (RAD)
  - The "System Design Document" (SDD)
- The documents will give us an introduction, overview and high level explanation of your application (the big picture)...
  - ...if sufficiently qualitative!
- Good (**not long**) documentation will make more justice to your project
  - Don't over-do it, **short, precise** descriptions appreciated
  - It's no magic, just try to explain the structure/behaviour of it!

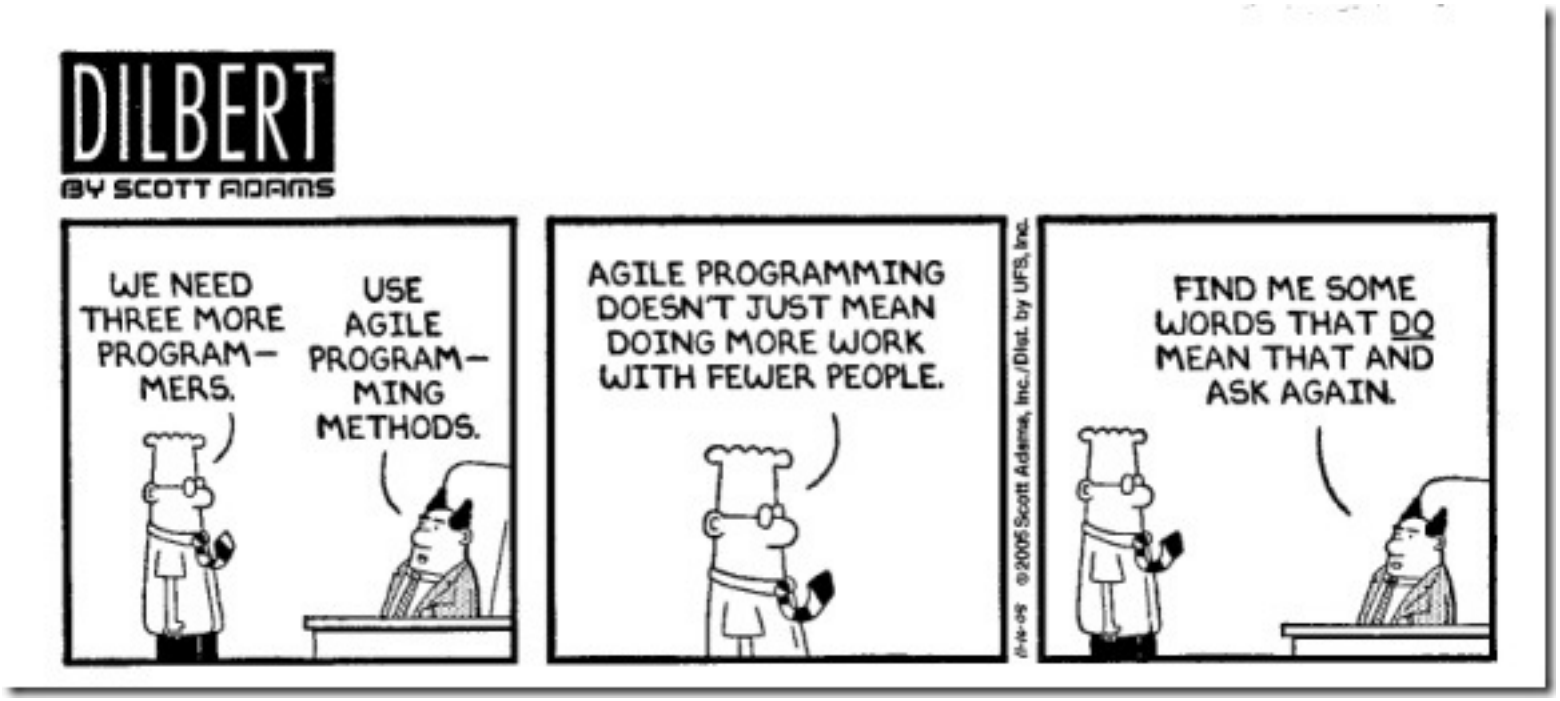
Supporting the Process

- During the process (or at different stages) we use;
  - "Domain driven design" (DDD), that means **"Focus on the model"** (more later)
  - Kind of "test driven development", (TDD) i.e. **all non-trivial code should be tested**
    - Unit testing for objects (classes)
    - Integration testing for collaborating objects (classes)
    - **There will be a workshop** (how to use the JUnit testing framework)
    - Tests are improve the projects quality (i.e. grading)

Practical Organizing of Software Development

- Version handling (for everything), Git , **mandatory!**
  - **There will be a workshop.** How to use the Git version control system

Hmmm


DILBERT
BY SCOTT ADAMS
WE NEED THREE MORE PROGRAM-MERS.
USE AGILE PROGRAM-MING METHODS.
AGILE PROGRAMMING DOESN'T JUST MEAN DOING MORE WORK WITH FEWER PEOPLE.
FIND ME SOME WORDS THAT DO MEAN THAT AND ASK AGAIN.
©2005 Scott Adams, Inc./Dist. by UFS, Inc.

Summary

- We use a simplified agile process
- The process has 4 phases
  - Output from one phase is input to next
  - During the process we repeatedly iterate the phases thereby for each iteration extending the application (adding functionality)
- Communication is extremly important
- We do some simple documentation
  - Of the process: Meeting agendas
  - Of the software: RAD and SDD

Next: Requirement elicitation