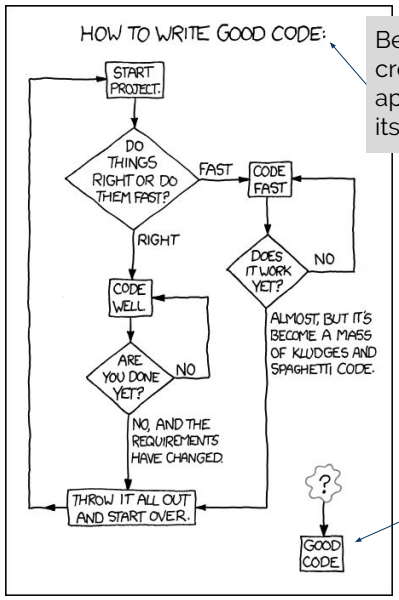
Software Development Overview

Slide Series #1

Software Problem



Better: How to create a high quality application fulfilling its intent?

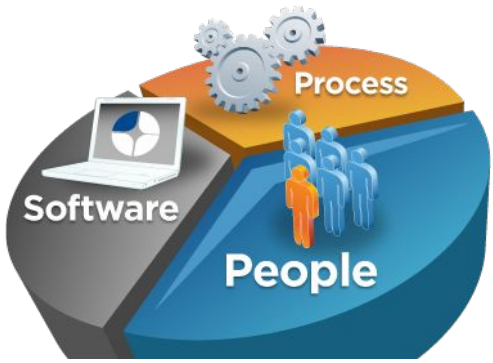
Software development (of the kind in course)

- Is often very complex because
  - many stakeholder,
  - often large applications,
  - concepts or specifications not well defined,
- Is a young engineering discipline, somewhat of an art ...
- Is in between very informal (dynamic/chaotic)
- Is short of mathematical tools (formulas)
- Is normally a group task
- Is highly dependent on communication

How to construct a high quality application fulfilling its intent?

- Unsolved problem
- But many impressive applications constructed

Software Process



To try control the construction of the application (and thus deliver a high quality application fulfilling its intent) we use a software development process

- An ordered (finite) number of "well defined" phases (or steps) to develop the application
- The process should at least guarantee a better result than no process

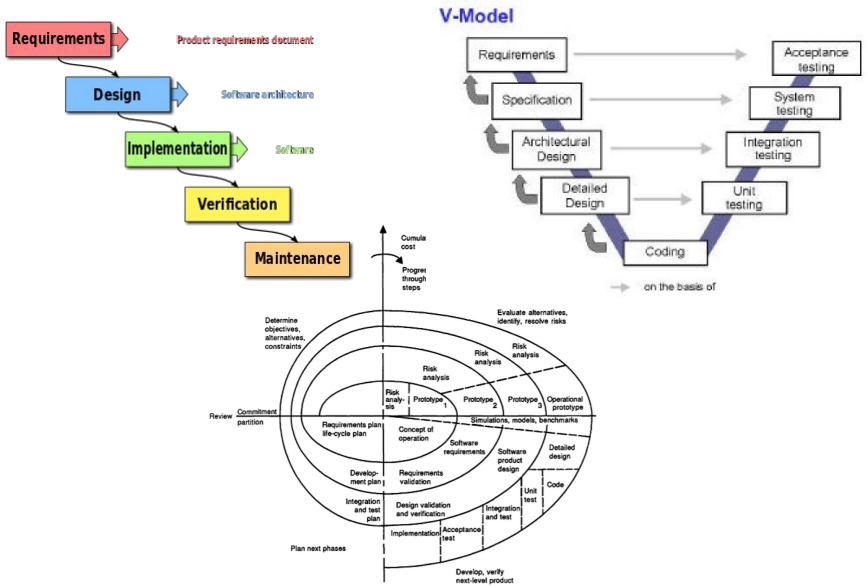
Software Processes

Agent-oriented programming, Agile software development, Agile Unified Process (AUP)
Aspect-oriented Programming (AOP), Behavior-driven development (BDD), Big Design Up Front (BDUF), Black box engineering Brooks's law, Cathedral and the Bazaar (see also Release early, release often (RERO)), Chief programmer team, CMMI Code and fix, Code reuse), Cone of Uncertainty, Constructionist design methodology (CDM), Continuous integration Control tables, Convention over configuration, Conway's Law, Cowboy coding, Crystal Clear, Design competition Design-driven development (D3), Design Driven Testing (DDT), Domain-Driven Design (DDD)
Don't Make Me Think (book by Steve Krug about human computer interaction and web usability)
Source Of Truth (SSOT), Dynamic Systems Development Method (DSDM), Easier to Ask Forgiveness than Permission (EAFP)Evolutionary prototyping, Extreme Programming (XP). Feature Driven Development (FDD). Free software license
Good Enough For Now (GEFN), Iterative and incremental development, Joint application design, aka JAD or "Joint Application Development", Kaizen, Kanban, Lean software development, Lean-To-Adaptive Prototyping in Parallel (L2APP) [1] Literate Programming, Microsoft Solutions Framework (MSF), Model-driven architecture (MDA), MoSCoW Method Open source, Open Unified Process, Pair programming, Project triangle, Protocol (object-oriented programming) Quick-and-dirty, Rapid application development (RAD), Rational Unified Process (RUP), Release early, release often (RERO) - see also The Cathedral and the Bazaar, Responsibility-driven design (RDD)
the Right thing, or the MIT approach, as contrasted with the New Jersey style, Worse is better., Scrum, Service-oriented modeling, Spiral model, Stepwise Refinement, Structured Systems Analysis and Design Method (SSADM) SUMMIT Ascendant (now IBM Rational SUMMIT Ascendant), Team Software Process (TSP), Test-driven development (TDD) Two Tracks Unified Process (2TUP), Ubuntu philosophy, Unified Process (UP)
Unix philosophy, User-centered design (UCD), V-Model, Hybrid V-Model [2], Waterfall model
Wheel and spoke model, When it's ready [3], Win-Win Model, Worse is better (New Jersey style, as contrasted with the MIT approach)

No common agreed upon approach, style or philosophy how to develop this kind of software

- No silver bullet!

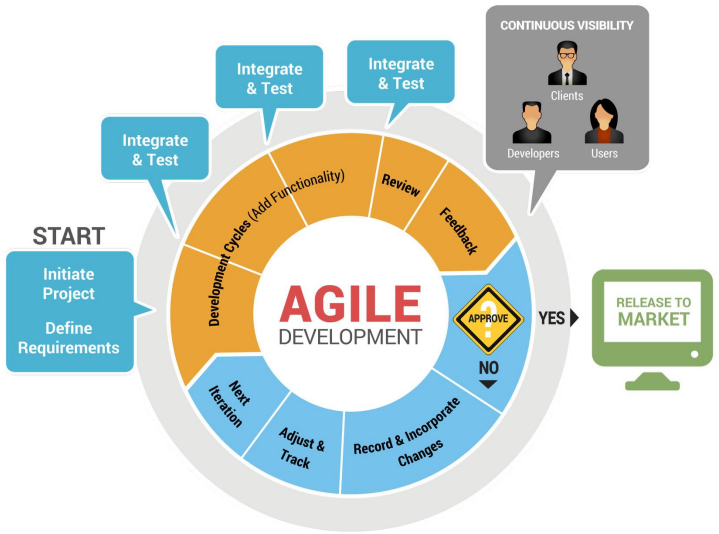
Process Models



How to organize the activities

- WE WILL NOT GO INTO THIS (general knowledge here, more in other courses …)
- Confusion: Hera a model is a process, later we talk about object oriented models which of course are not processes

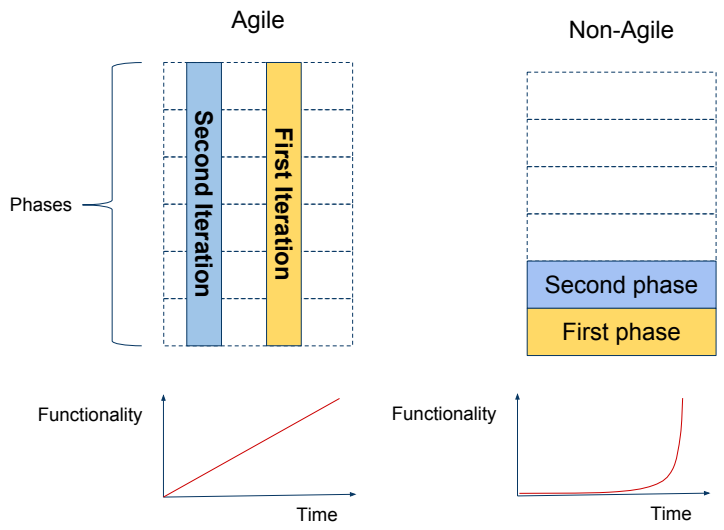
Agile Development



Agile development

- Feedback (customer) is the primary control mechanism
- Start with a rather preliminary specification.
- Implement functionality iteratively and incrementally (repeatedly in small steps) and learn
- Pros: Quick adaption to changes/problems
- Cons: Insufficient design and documentation (missing general aspects of the problem)
- Picture is a bit overkill for this course, this is NOT a process course.

Iterations

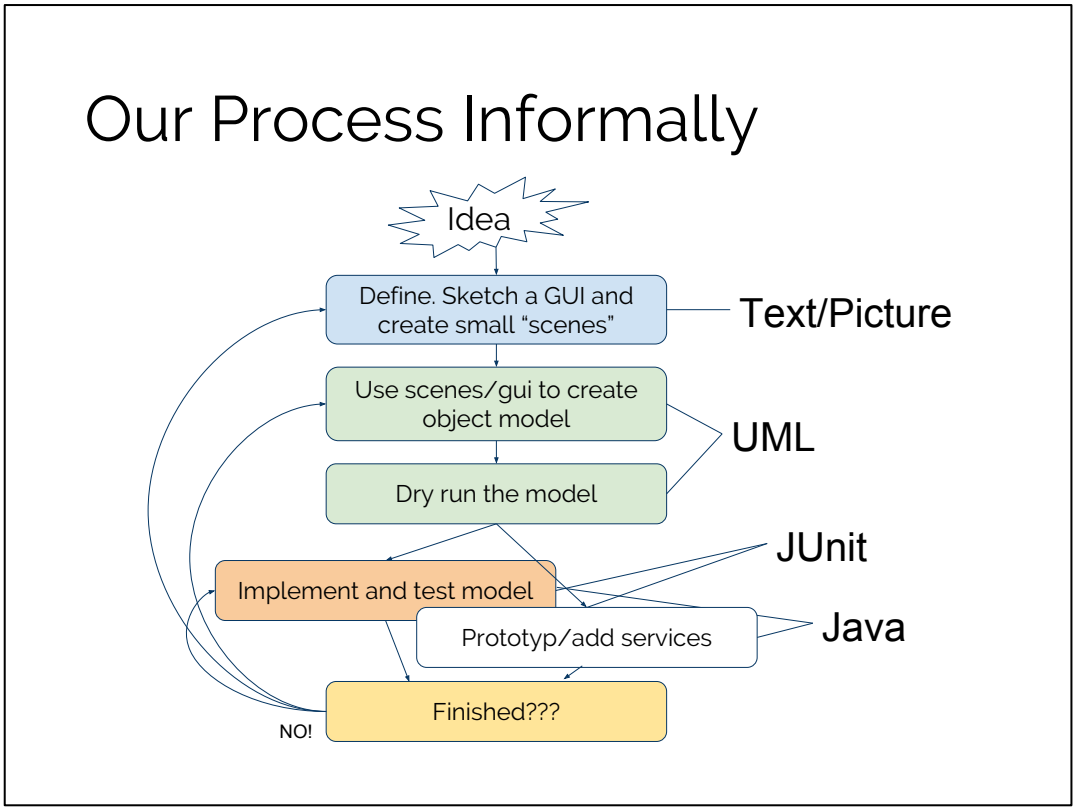


Agile

- We'll build an executable small portion of the application covering all development phases
- We will have something to run after first (each) iteration
- Each iteration will increment functionality

Non-agile

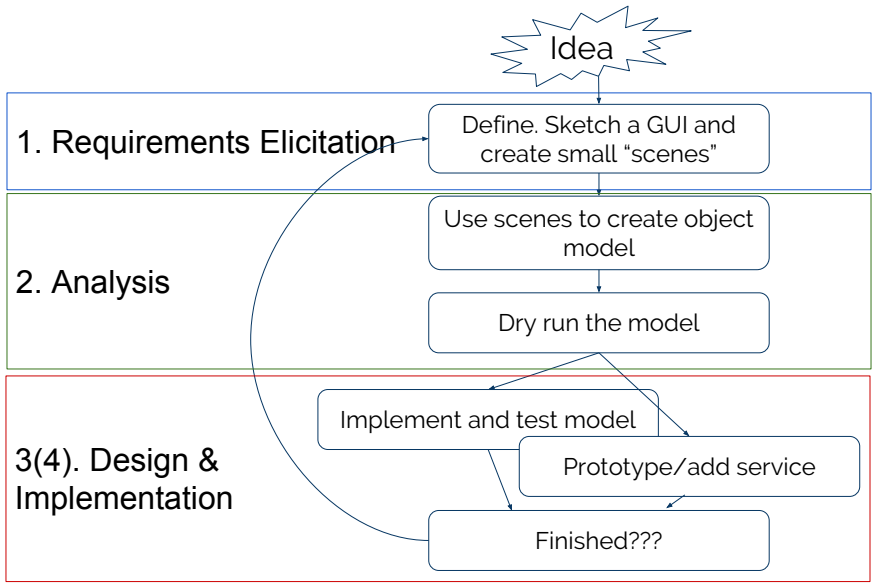
- Each development phase is completed before next is started.
- Nothing to run until (close to) finished.



We use a simple (no name) agile process model

- We need to have some common vision what to build
- We should have some understanding (and some beginning of a solution) before the implementation starts
- We start to implement a few selected functions ...
- ... thereby gaining deeper understanding ...
- ... as a result we possibly update the current solution ...
- ... we refactor the code to clarify/reflect the new solution ...
- ... then we continue with a few more functions (and other stuff needed, like services to the model)...
- ... until finished!

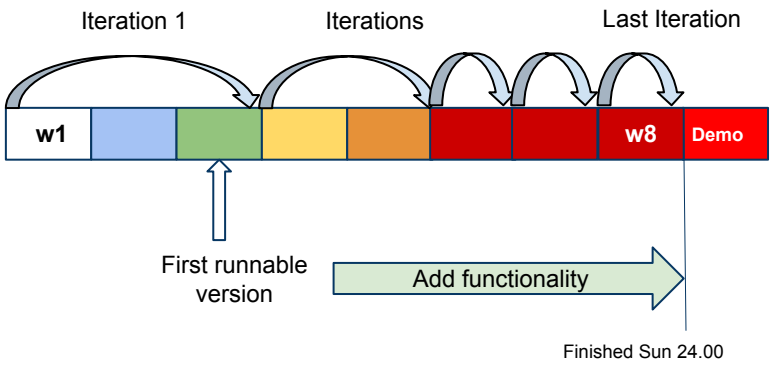
Our Process More Formally



The formal names for the phases/steps for <u>one</u> iteration

- Requirement Elicitation
- Analysis
- Design and Implementation
- … more to come

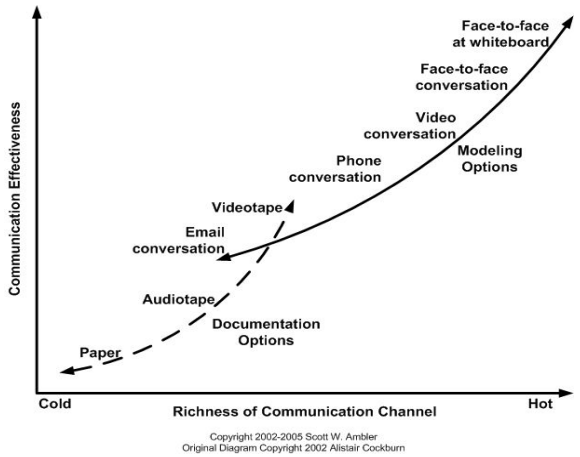
Iteration Planning



Iterations and weeks

- Must have something to run late week 3 (probably some tests, more later …)!

Process Communication



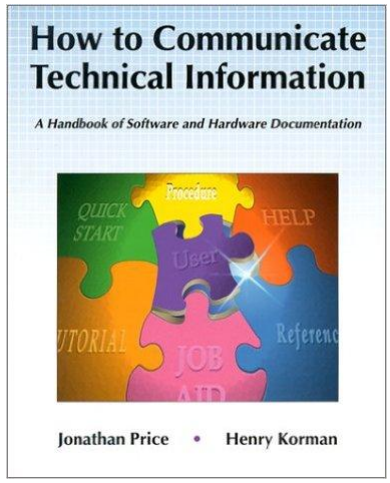
Effective communication is a fundamental requirement for software development.

Find a room with a whiteboard and gather
- Don't spread the group!

Use issue trackers, can't remember everything…
- Most IDE have “TODO” lists (use // TODO in NetBeans, IntelliJ)
- Web based issue trackers

Wordlist for definitions
- Have experienced group members using same notion for different concepts! Confusion … time lost…
- If any ambiguity write down in wordlist (and/or as class comment).

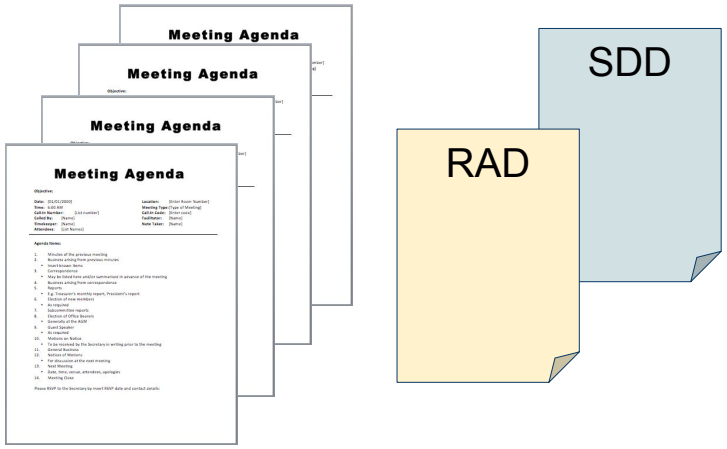
Software Documentation



Software documentation is a very debated topic

- Very hard to achieve good documentation
- A topic in it's own right (more in other courses)
- Many views, no real common opinion
- Problems: How to, what to, … how to keep in sync with code
- Agile documentation
- Criticism (and alternative)
- .. anyway we do some basic documentation

Documentation in Course



This is a programming/design course. If in any doubt always prioritize the code!

- Overall goal for documentation: To trace process, to understand application

Documentation consists of:

- Meeting agendas
- Two different documents describing the application
  - The Requirements and Analysis Document (RAD)
  - The System Design Document (SDD)
  - … more to come.

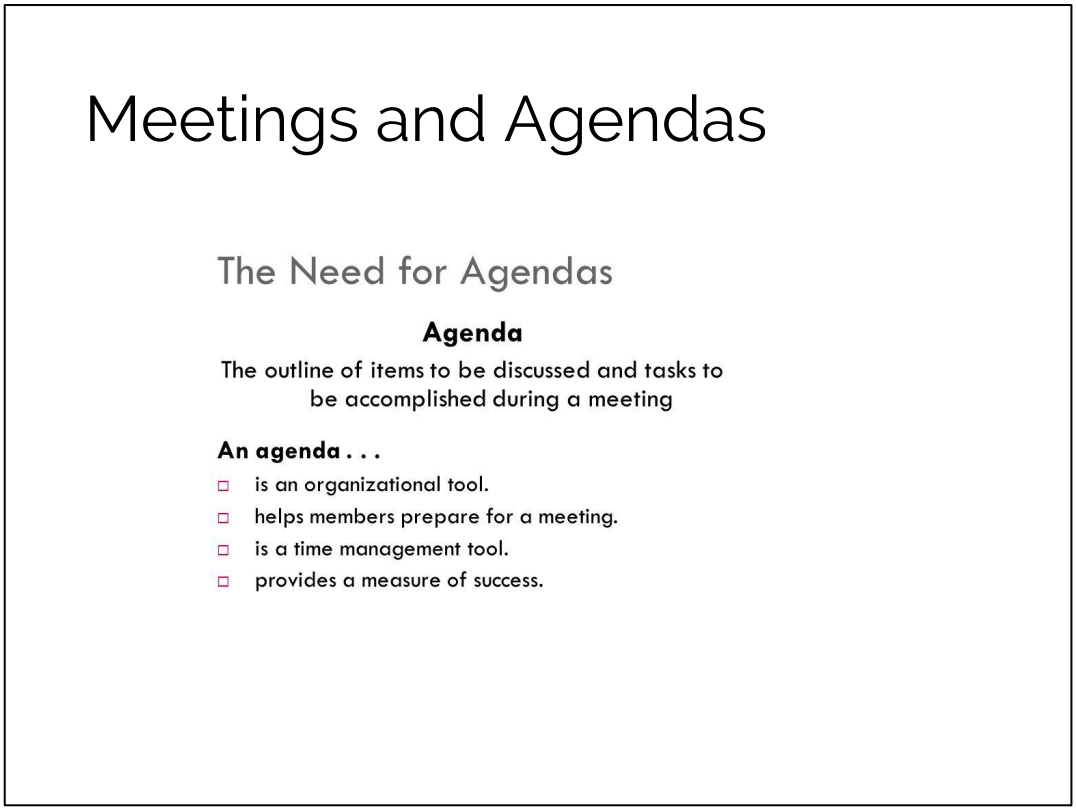
For all documentation we prefer

- Short and focused
- If sections not applicable just put an "NA"
- Try to be Pedagogical
  - Next person to be involved should have an easy road to understanding
- Use templates (on course page)

This is a First Year Course

- Can't expect professional documentation
- We'll not be able to produce "real" RAD and SDD.
- Sometimes we have to use our imagination to fill in, do so...
- … but
  - We will use the documentation during grading to get an understanding of your application
  - Easier to get a fair grade if we understand (time for grading is limited)

All documentation should be in Git repo!



To keep the project focused and on track and controlling the process there should be meetings organized by you

- There should be min. 2 organized meeting/week (besides the mandatory supervised meeting)

The meeting should have an agenda (mandatory),

- What to discuss,
- A documented outcome!
  - What decisions are made?
  - Who is responsible for what?
  - Outcome documented in agenda (i.e. same document)

Being a chairman is a qualified task (rotate);

- Keep time, keep discussion focused and on track, …
- Be efficient!…socialize after the meeting…
- … and of course we all arrive in time and end in time

Target audience for documented agendas are

- First and foremost the group
- To a lesser degree assistants and later examiner
- It's not ok to update the agendas/outcomes, they are immutable
  - Fix another meeting if previous failed…

Is this Suppose to Work?!

I claim this way to work will be better than no process

- But as noted … there are no silver bullets …
- Must use your skills and good judgement!

Common student comment after course: Now we know how we should have worked …

- So this is a learning process

Summary

- We use a simple agile process
- The process has 3 (4) phases
- Communication is important
- We do as little documentation as possible
  - Meeting agendas
  - RAD and SDD

Next: Requirement Elicitation

Cartoon: Not the way we do it!