Software Development Overview

Slide Series 1

Software Development ...

... of this kind

- 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

... so how should we manage to ..??

Approaches, Styles and Philosophies

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 ar d with the MIT a

No common agreed upon approach, style or philosophy how to develop this kind of software (no silver bullet)!

Software Development Process

The activities: Planning, designing, implementation, testing, documentation, deployment, maintenance,..

Some specific process models (how to organize activities, aka process framework)

- Waterfall model
- Spiral model
- Iterative and incremental development
- Agile development
- Rapid application development
- Code and fix
- ...

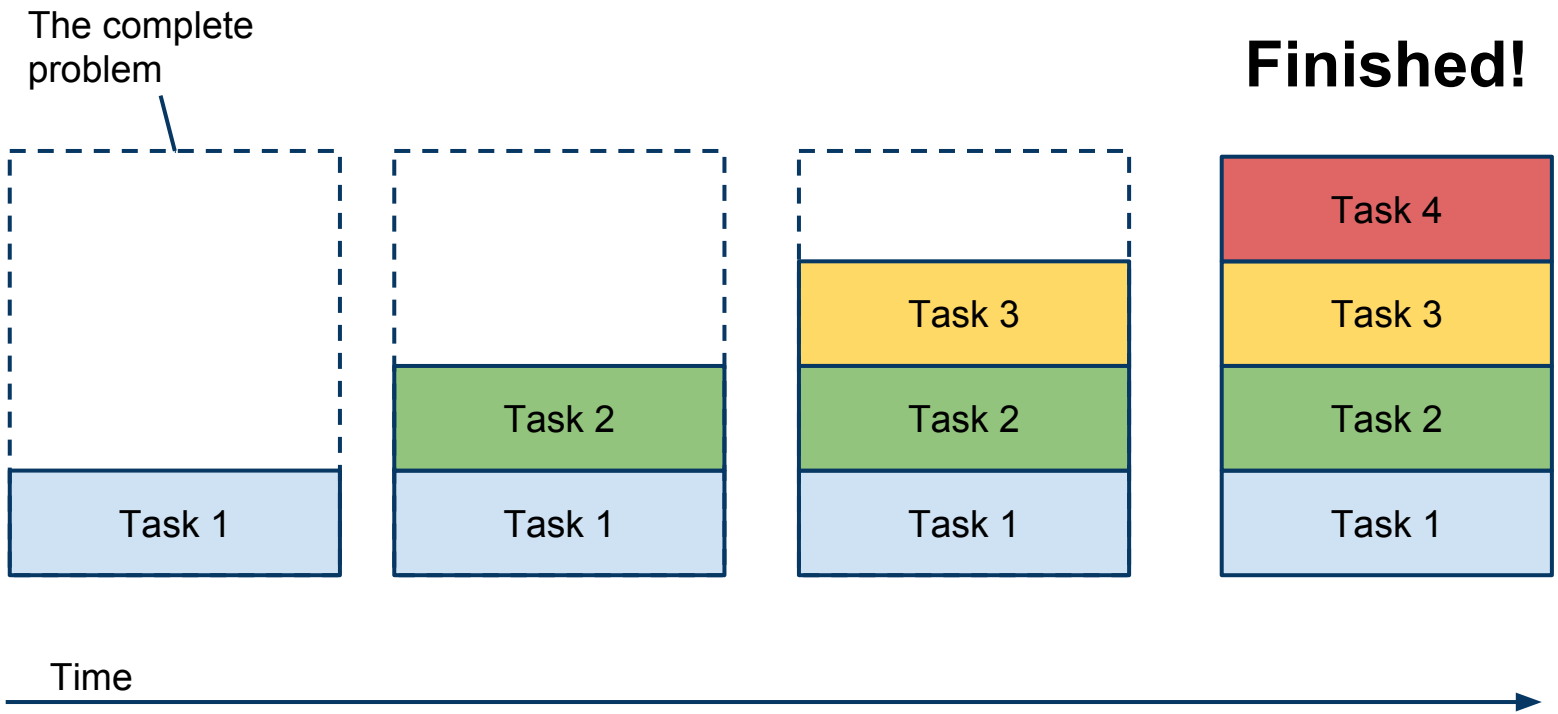
Confusion: Hera a model is a process, later we talk about object oriented models which of course are not processes

Waterfall Model and Similar

Big design up front (BDUF), heavy process

- Planning is the primary control mechanism
- The traditional (and often very successful) engineering approach
- Everything is specified before starting to implement
- Each task is completed before next begins (if not possible to parallelize)
- Pros (positive): Efficient in general ...
- Cons (negative): Hard to handle changes (specification obsolete before we begin to implement)

Waterfall Model and Similar, cont



Agile Models

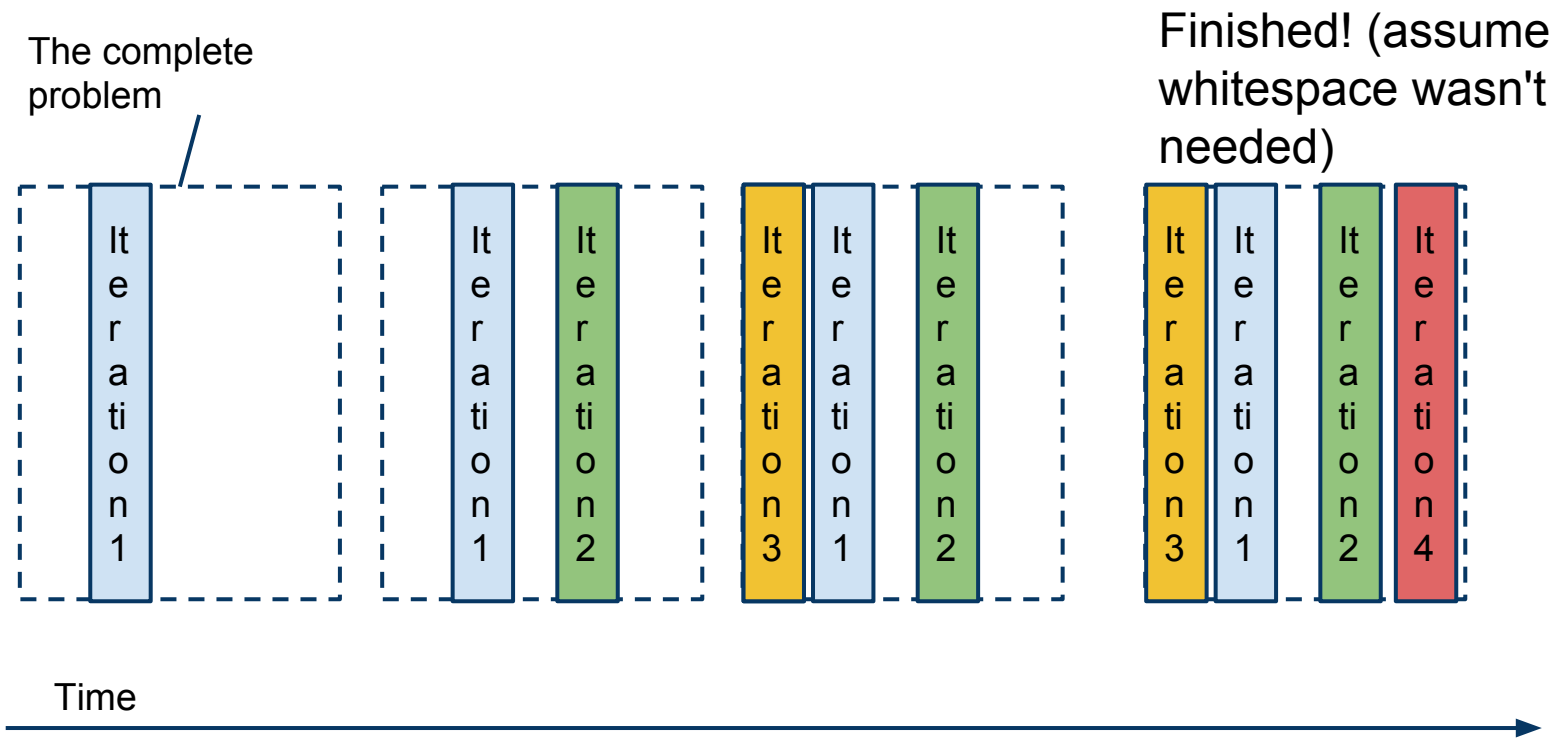
Agile development, lightweight process

- Feedback (customer) is the primary control mechanism
- Start with a rather preliminary specification. Implement functionality **iteratively** (repeatedly in small steps) and learn
- Pros: Quick adaption to changes/problems
- Cons: Insufficient design and documentation (missing general aspects of the problem)

Right now the **Scrum** model seems to be very hot

"Scrum is an iterative and incremental agile software development framework for managing software projects and product or application development"

Agile Models, cont



Process Model In Course

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 parts ...
- ... thereby gaining deeper understanding ...
- ... as a result we possible update the current solution ...
- ... we refactor the code to clarify/reflect the new solution ...
- ... then we continue with a few more part...
- ...
- ... until finished!

Process Model Phases

Our process models is composed of a number of steps

- We say **phases**

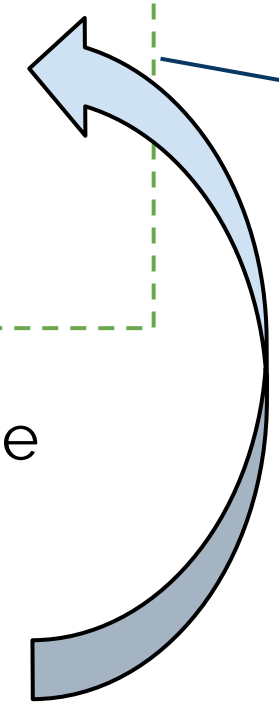
Phases have input and output

- Output from phase n, input to n+1

Phases for our Process Model

We have 4 phases.

1. Requirement Elicitation
   -What are we going to build?
2. Analysis
   Build a model for the problem
3. Design
   Adapt (parts of) model so it can be implemented
4. Implementation
   Implement (parts of) model



This has little with computers and programming to do

Iterations

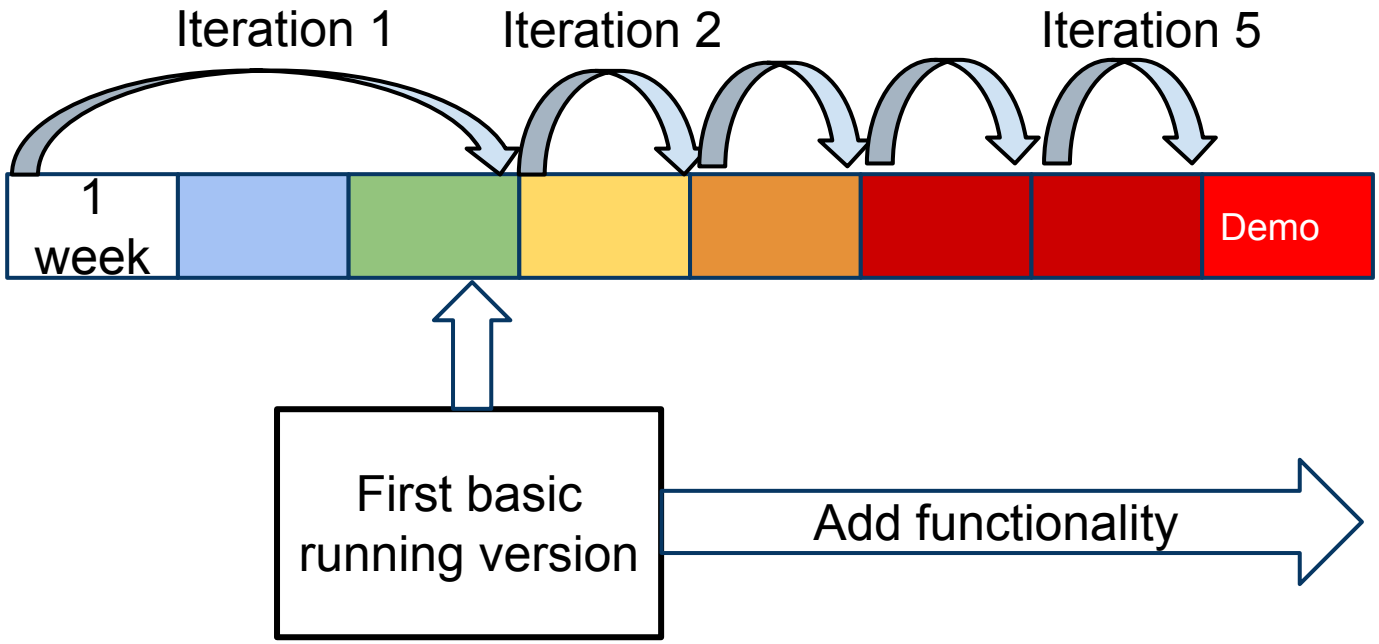
Each iteration runs through the phases (focus shifts. Early iterations focus 1 & 2 later more on 3 & 4)

To Build a Model

Because we use the object oriented paradigm we build an object model

- Later our OO language will support the implementation of an executable version of the model (by providing classes, etc.)

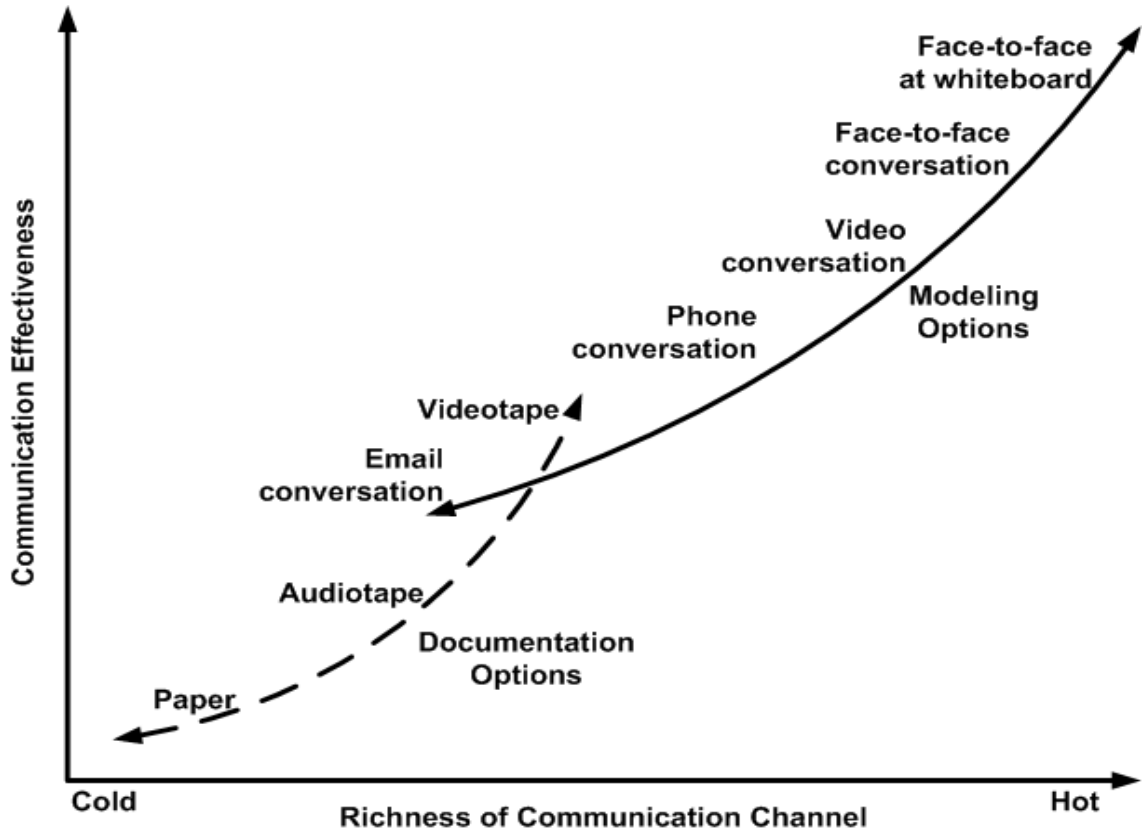
Iteration Planning



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 NetBeans
- Better: Google code (or similar...)

Wordlist for definitions

- Have experienced group members using same notion for different concepts! Confusion ... time lost...
- If any ambiguity write down in wordlist.

Process Models: Documentation

BDUF processes

- Often specifies a lot of documentation
- Has gained criticism for putting too much attention to documenting (where's the (running) code?)
- Code and documentation not synchronized

Agile methods

- Less documentation centric
- Document "big picture" (overview)
- Working code is the ultimate documentation (also documentation in code, see Java library classes)
- Code style: Self documenting code
- Tests used as documentation, more to come...

Documentation in Course

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

Anyway need some documentation

- **Meeting agendas***
- Two documents describing the application
  - **The Requirements and Analysis Document (RAD)***
  - **The System Design Document (SDD)***

All documents in Git

*) Also used as basis for report in LPS310

Groups Meetings (self organized)

Purpose

To keep the project focused and on track. Controlling the process. There should be an agenda, what to discuss, and 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

Audience

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...

The RAD

Purpose

Describes the system in terms of models, functional and nonfunctional requirements and serves as a contractual basis between the **customer and the developer**. The RAD must be written in the language of the customer's domain of business/expertise. Under no circumstances should any "computerese" terminology creep into this document.

Audience

The customer, the users, the project management, the system analysts (i.e., the developers who participate in the requirements), and the system designers (i.e., the developers who participate in the system design).

The first part of the document, including use cases and nonfunctional requirements, is written during requirements elicitation[phase 1]. The formalization of the specification in terms of object models is written during analysis [phase 2], more to come...

It's ok to update the RAD during the process, don't try to write a final version for iteration 1

The SDD

Purpose

The system design is recorded in the System Design Document (SDD). This document completely describes the system at the architecture [high] level, including subsystems and their services, hardware mapping, data management, access control, global software control structure, and boundary conditions [start/stop]. A foundational guide for further implementation details all the way to an executable solution.

Audience

The audience for the SDD includes the software architect and lead members (liaisons) from each subsystem development team (i.e. programmers).

The SDD is a "live" document that should be incrementally expanded and refined during/after iterations.

Some considerations for RAD and SDD

General: 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
- Top down (overview->details) probably best approach

Use templates (on course page)

Documentation Quality In Course

This is a First Year Course

- Can't expect fully 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)

Supporting the Process

During the process we use;

- **"Domain driven design"**
- **"Test driven development"**

Both the above are ways to work inside the process

Domain Driven Design

DDD means

- Using the language of the domain (problem area)
- Placing primary focus on the core domain and domain logic,
- Solutions to the problem is in a domain model (can't solve problem just by using technique, "happy hacking"...)
- Design application on model of the domain.

For short: "Focus on the model" (more later)

Test Driven Development

Testing is our primary technique to ensure quality (i.e. no bugs)

- As soon as we have a, to some extent stable , executable model, we adhere to test driven development
- We write tests in parallel to the application code, testing all non-trivial code, ensuring the functionality of the code
- Test are collected into test suites covering the complete model (and possible other parts)

Reminder: **There will be a workshop on testing using JUnit**

Practical Organizing of Code and Other Artifacts

Version handling for everything; code, agendas, documents, ... **Git** , mandatory!

- There will be a Git-workshop in this course.
- Only need very basic usage
- NO passwords, please!

Reminder: **There will be a workshop on Git**

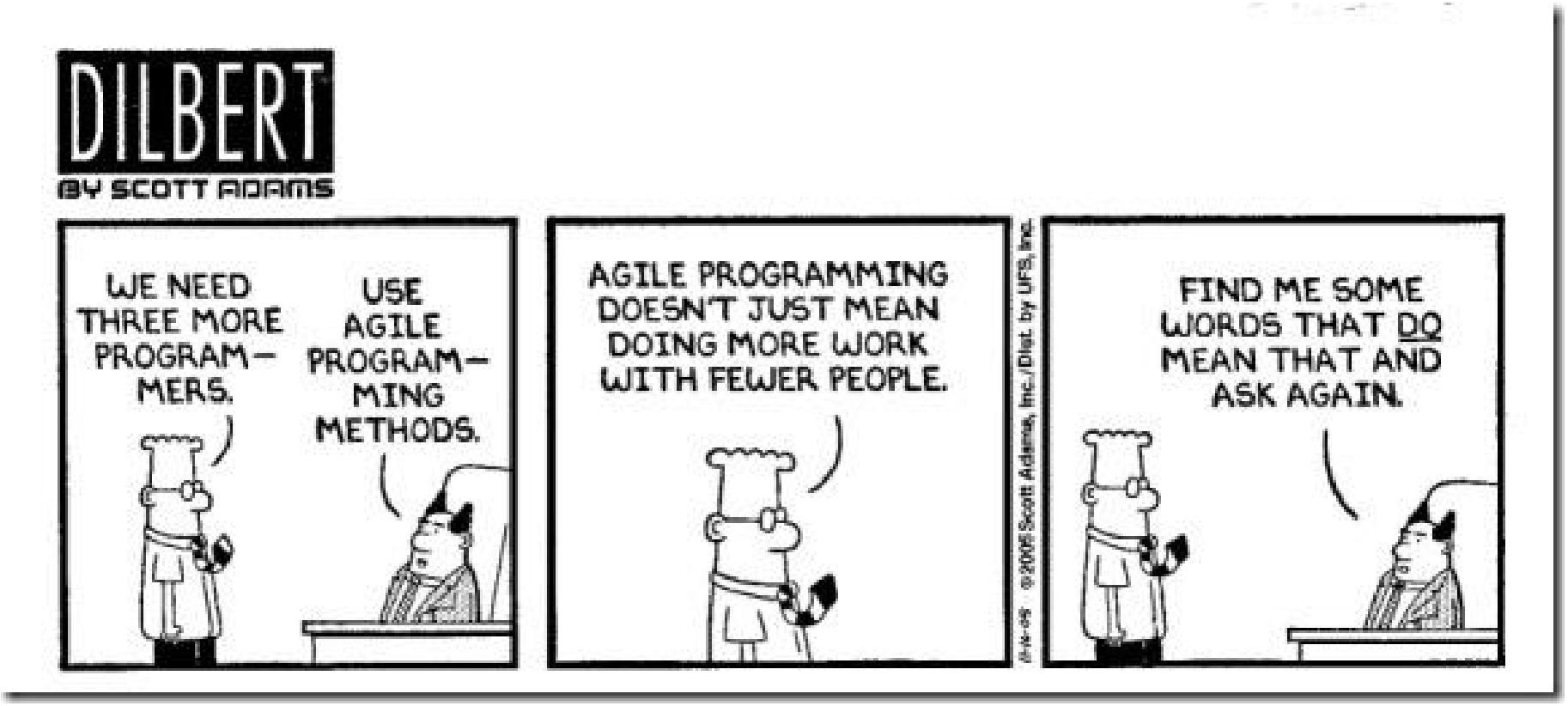
So, Is this supposed to work...?

Yes*!

"We started to feel nervous just doing modelling, no (graphics) programming, ... but when we added the graphic's it just worked!"
// Quote from student

*) Remember no "silver bullet", must have skills...

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.
©2008 Scott Adams, Inc./Dist. by UFS, Inc.

Summary

- We use a simple agile process model
- The model 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
- Communication is extremely important
- We do some basic documentation
  - Of the process: Meeting agendas
  - Of the software: RAD and SDD
- We use DDD and TDD

Next: Requirement elicitation