Development Processes

Introduction to Systems Engineering
I2ISE

Why use a development process?

- The process will help you answer some very important questions:
  - What will you *produce*?
  - When will you be *done*?
  - What will it *cost*?
  - How will you handle *changes*?

- Answers to these questions are important to you *and* the customer

The *Developer* Bill of Rights

- You have the right to know *what* is needed - clear requirements, clear priorities.
- You have the right to say *how long* each requirement will take you to implement
- You have the right to *revise estimates* given experience.
- You have the right to *accept* your responsibilities instead of having them *assigned* to you.
- You have the right to produce quality work *at all times*.
  - Not just 0900-1700
- You have the right to *peace*, *fun*, and *productive and enjoyable* work.

Ron Jeffries and Kent Beck

The *Customer* Bill of Rights

- You have the right to an *overall plan*, to know what can be accomplished, when, and at what cost.
- You have the right to *see progress in a running system*, proven to work by passing repeatable tests that you specify.
- You have the right to *change your mind*, to *substitute functionality*, and to *change priorities*.
- You have the right to be informed of *schedule changes*, in time to choose how to reduce scope to restore the original date.
- You have the right to *cancel* at any time and be left with a useful working system reflecting investment to date

Kent Beck

The Process and the Rights

- The development process should *guarantee* a way of working that respects both the developer's and customer's Bill of Rights

- You have the right to an *overall plan*, to know what can be accomplished, when, and at what cost.
- You have the right to *see progress in a running system*, proven to work by passing repeatable tests that you specify.
- You have the right to *change your mind*, to *substitute functionality*, and to *change priorities*.
- You have the right to be informed of *schedule changes*, in time to choose how to reduce scope to restore the original date.
- You have the right to *cancel* at any time and be left with a useful working system reflecting investment to date

Kent Beck

- You have the right to know *what* is needed - clear requirements, clear priorities.
- You have the right to say *how long* each requirement will take you to implement
- You have the right to *revise estimates* given experience.
- You have the right to *accept* your responsibilities instead of having them *assigned* to you.
- You have the right to produce quality work *at all times*.
- You have the right to *peace*, *fun*, and *productive and enjoyable* work.

Ron Jeffries and Kent Beck

The goal of the process

- The goal of the process: To deliver a system…
  - …that works
  - …on time
  - …on budget
  - …that is maintainable, extensible and reusable

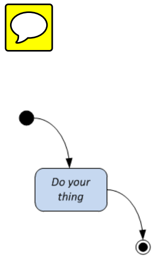
- The process is successful iff it preserves the rights and meets the goals

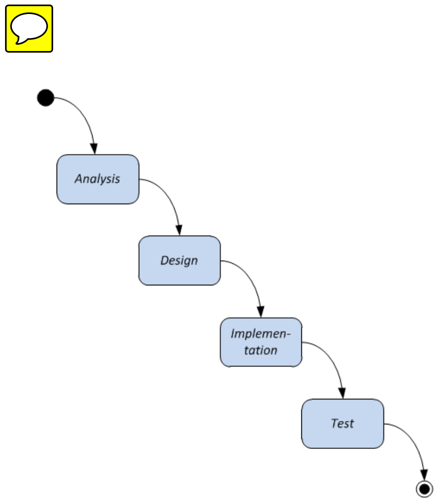
Examples:
Traditional development processes

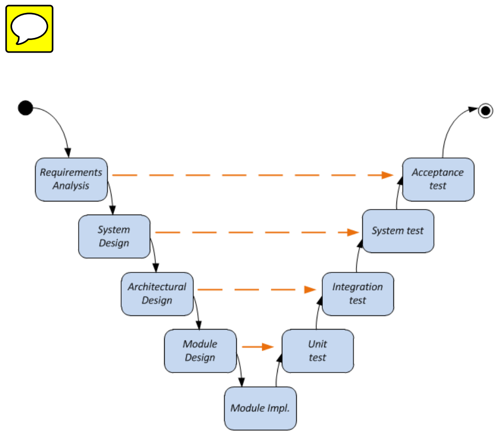
- The null process
- The waterfall process
- The V-model

Discussion

- When do the traditional processes produce a result visible to the customer?







- Is this a problem? Why?

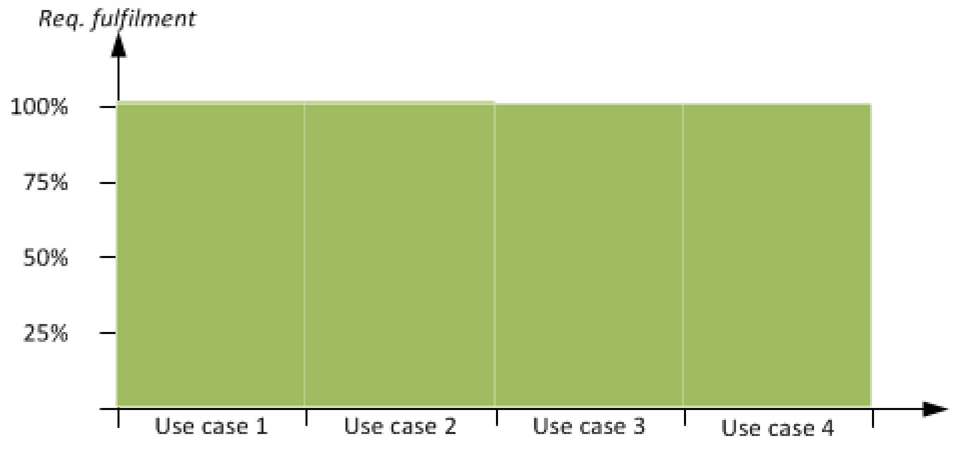
Iterative and incremental development processes

- *Iterative* refers to the repetitive nature of the process
  - An *iteration* is a single repetition of the same sub-process.
  - The sub-process result: Partial *production quality* system

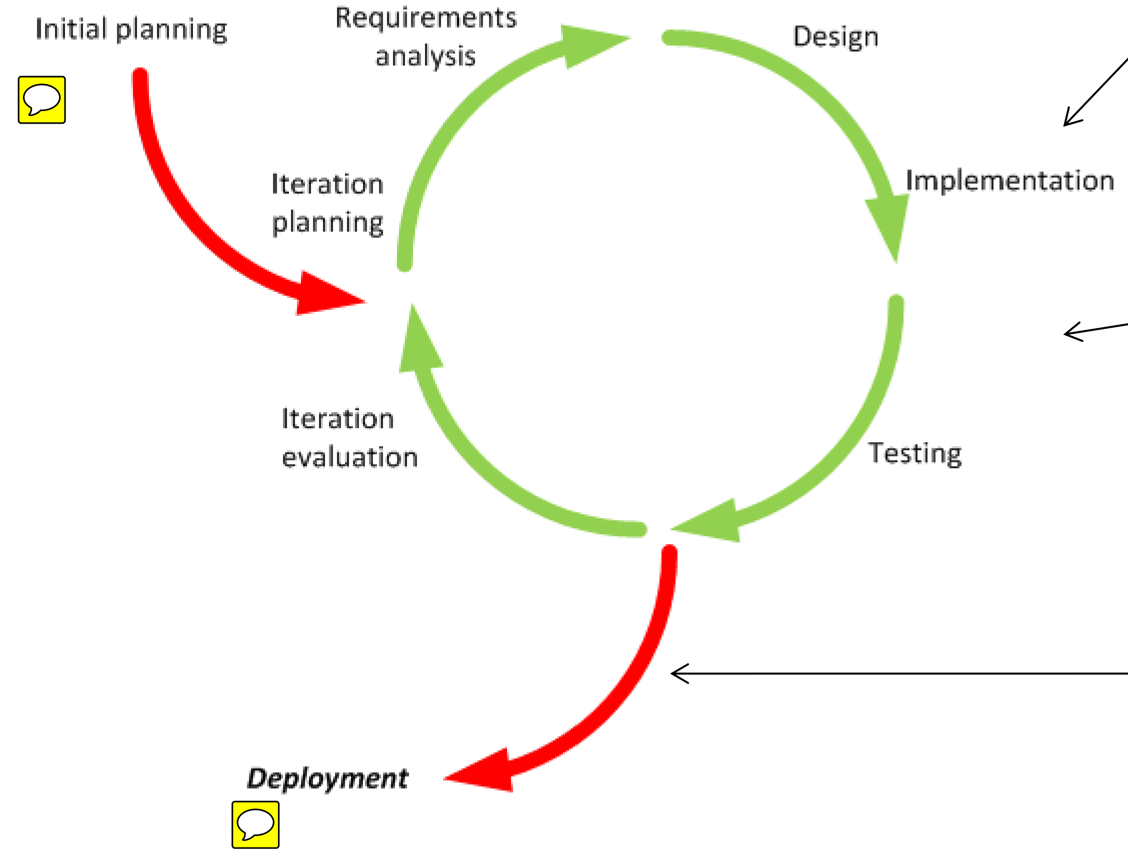
- *Incremental* refers to the *continued expansion* of system capabilities

Iterative vs. *incremental*

- Iterative *and* incremental



Iterations



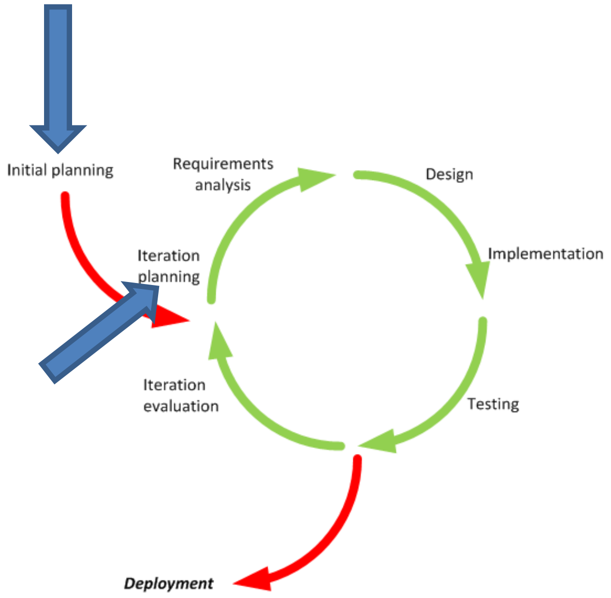
Iterations are short (2-6 weeks)

Iterations are *timeboxed* – reduce requirements, do not extend deadline

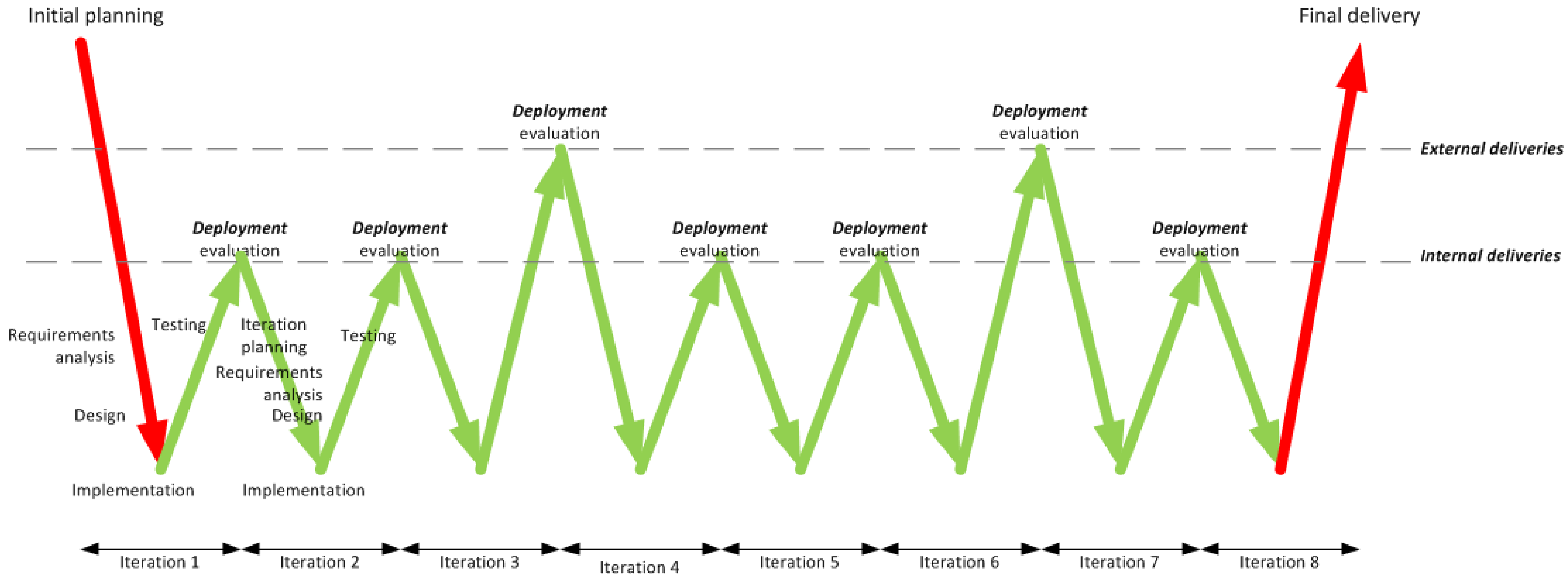
Each iteration produces an internal or external *production-grade system*

Iterations - planning

- Planning is done in two stages
- *Initial* planning
  - Allocate requirements to iterations 💬
  - Specify deliverables
- *Iteration* planning
  - Plan implementation of allocated requirements and previous iteration overruns
  - Refine deliveries
- Planning is very often done in units of *use cases*
  - Most *ciritcal* use cases are done first 💬



Iterative process – another view



Iterative and incremental development processes

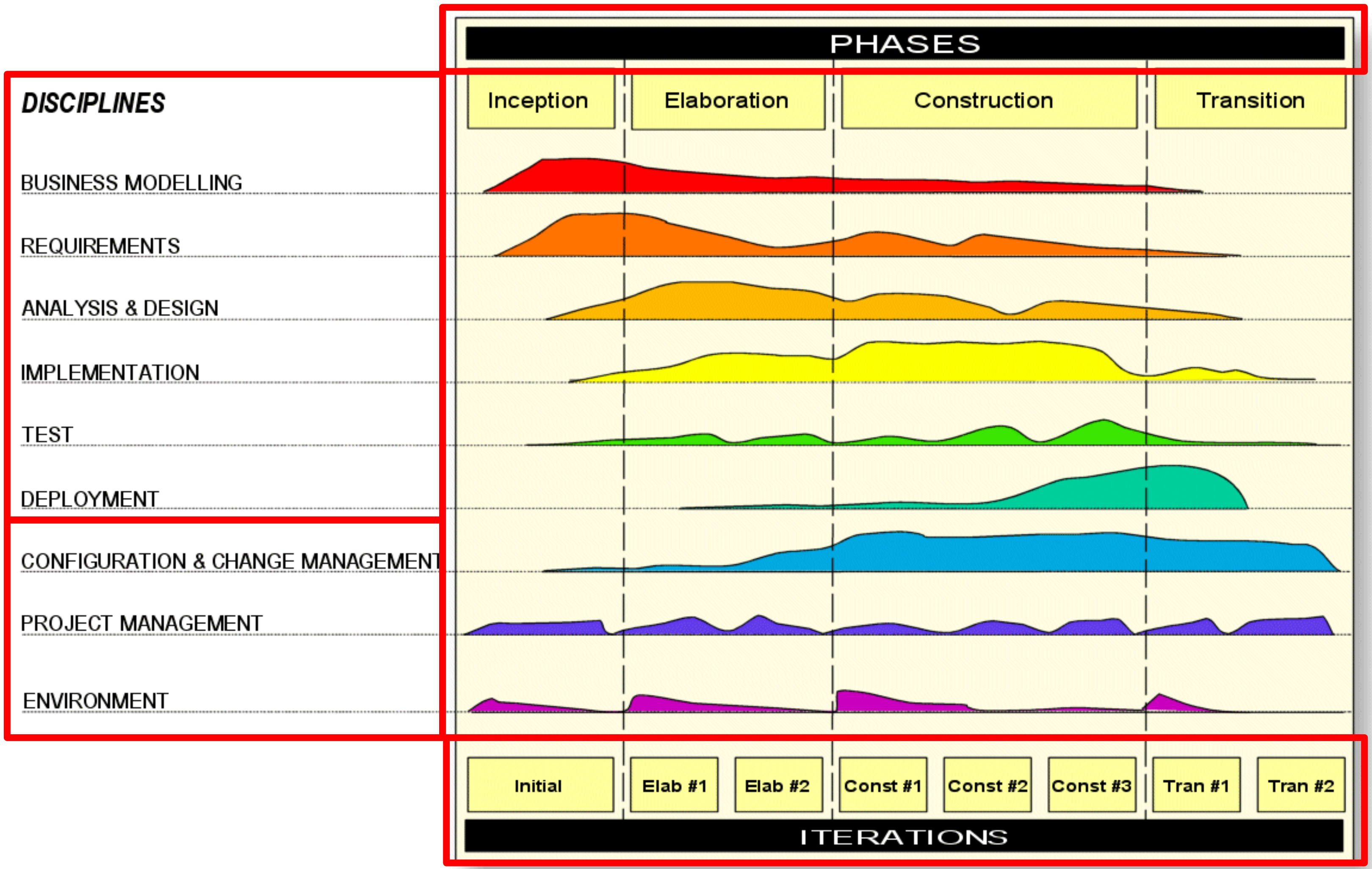
- Examples iterative and incremental processes:
  - Rational Unified Process (RUP)
  - Rapid Object-oriented Process for Embedded Systems (ROPES)
  - XP
  - Other agile processes

Example: Rational Unified Process

- Rational Unified Process (RUP)
  - Developed by *Rational Software (*now IBM*)*
  - Developed from the *Unified Process*
    Jacobson, Booch, Rumbaugh

- Actually a process *framework* from which processes can be *instantiated*

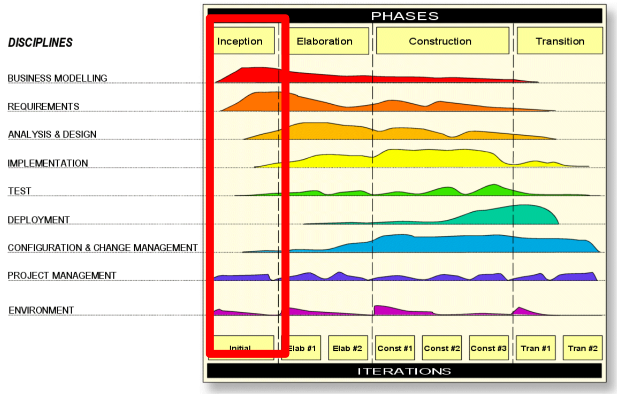
RUP: The works



RUP: Inception phase

- Life-cycle objectives of the project are stated, so that the needs of every stakeholder are considered.
- Scope and boundary conditions, acceptance criteria and some requirements are established.

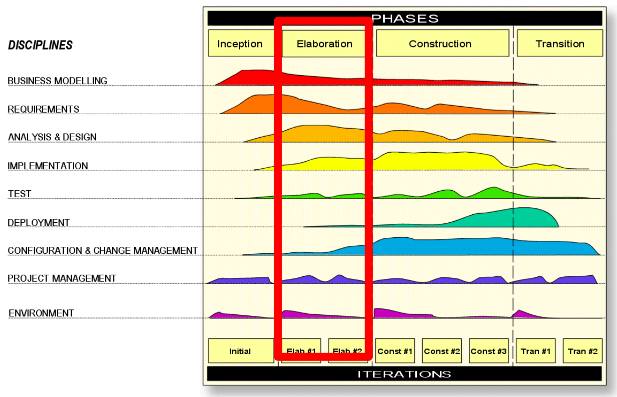
- Activities:
  - Problem description
  - Product limitations
  - Requirements definition (use cases)
  - Acceptance test plan
  - Risk analysis
  - High-level architectural considerations



RUP: Elaboration phase

- Determine risks, stability of vision of what the product is to become
- Determine stability of architecture and expenditure of resources

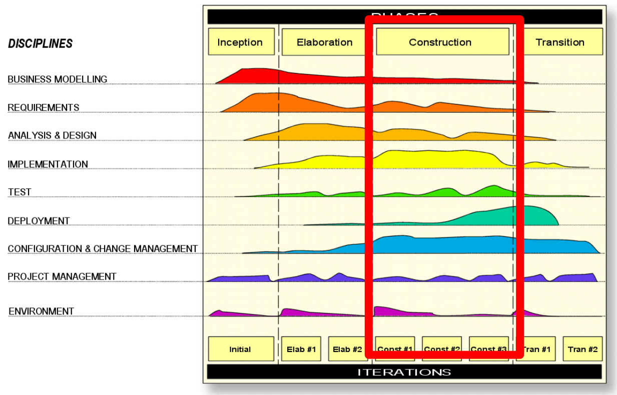
- Activities:
  - Requirements elaboration, prioritization and allocation to Construction iterations
  - Risk mitigation
  - Domain analysis and design
  - HW/SW architectural considerations
  - Interface specifications



RUP: Construction phase

- Manufacture produce
- Manage risk, resources, etc. to optimize cost, schedule and quality
- Detailed iteration planning and tracking

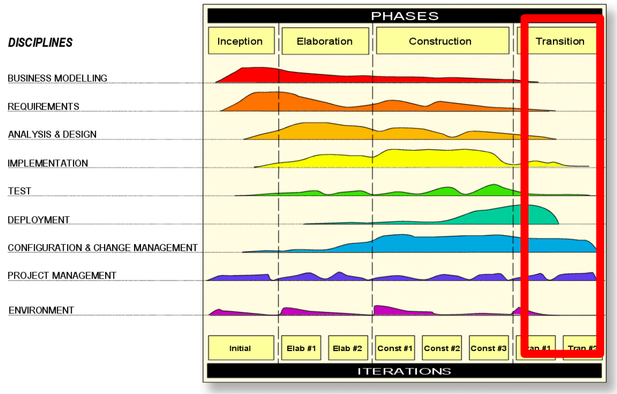
- Activities:
  - Construction, unit/integration/system tests
  - Per-iteration working system prototype
  - Continuous focus on risk mitigation, planning etc.



RUP: Transition phase

- Marketing, packaging, installing, configuring
- Supporting user community, making corrections, updates, etc.

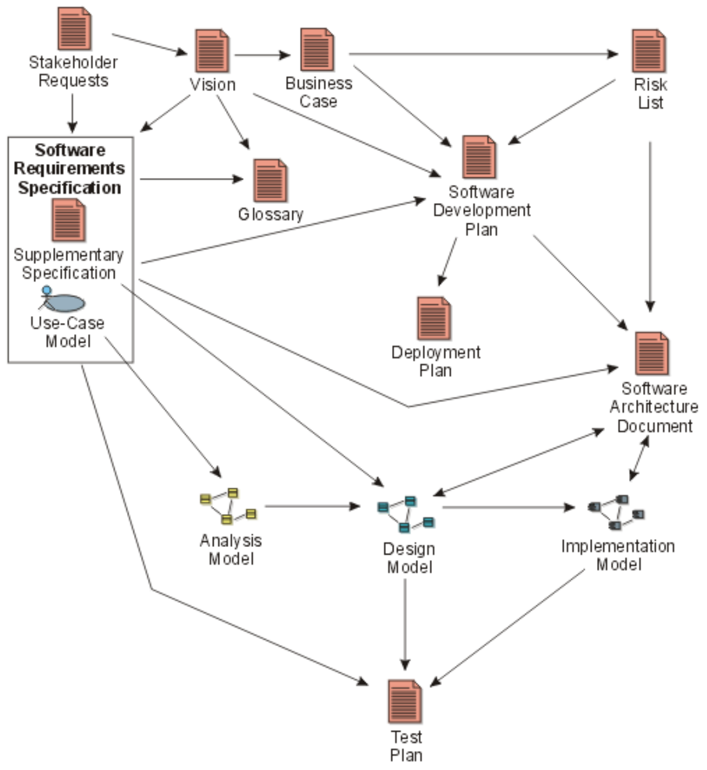
- Activities:
  - Acceptance test (alpha/beta test if planned)
  - Corrections, configuration control
  - User education
  - Production tests and documentation
  - Marketing
  - Market implementation



RUP: Artifacts

- RUP defines a lot of *artifacts* associated to the disciplines
  - *Documents*
  - *Models* (or *model elements*) with associated *reports*

- Is RUP a ”light” or ”heavy” process?



Agile methods

- For small-to-mid-scale development projects, *agile* processes are more applicable
  - Faster production
  - More adaptive in a changing world

- Faith in *people* (developers) rather than *paper*

Agile methods: The agile manifesto

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile methods: Principles

- Developers communicate face-to-face, not through documentation
- Team and customer work together beginning → end
- Embrace change 💬
- Produce software rather than paper
- Early and continuous deliveries of software

Example: eXtreme Programming (XP)

- Developed by Beck, Cunningham and others
  - First used in 1996

- Some characteristics:
  - Focus on *customer satisfaction*
  - Permanent on-site *customer presence*
  - Short development cycles
  - Incremental planning
  - Continuous feedback
  - Evolutionary design

XP core activities

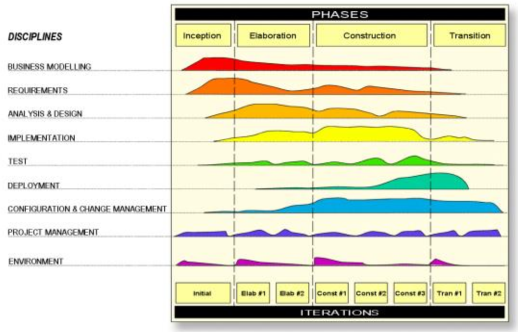
• **Coding**	The only true product is software
• **Testing**	If a little testing finds a few errors, a lot of testing finds a lot of errors
• **Listening**	Listen to the customer and give him feedback
• **Designing**	Good design avoids a lot of complications – and errors

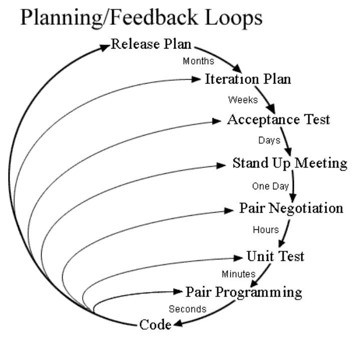
XP values

• **Simplicity**	Do what's asked – no more!
• **Communication**:	Everyone is part of the team and will communicate – daily, face-to-face
• **Feedback**:	Demonstrate early and often, listen to feedback, make changes
• **Respect**:	Developers respect each other, developers and customer respect each other
• **Courage:**	Tell the truth about progress and estimates

Discussion

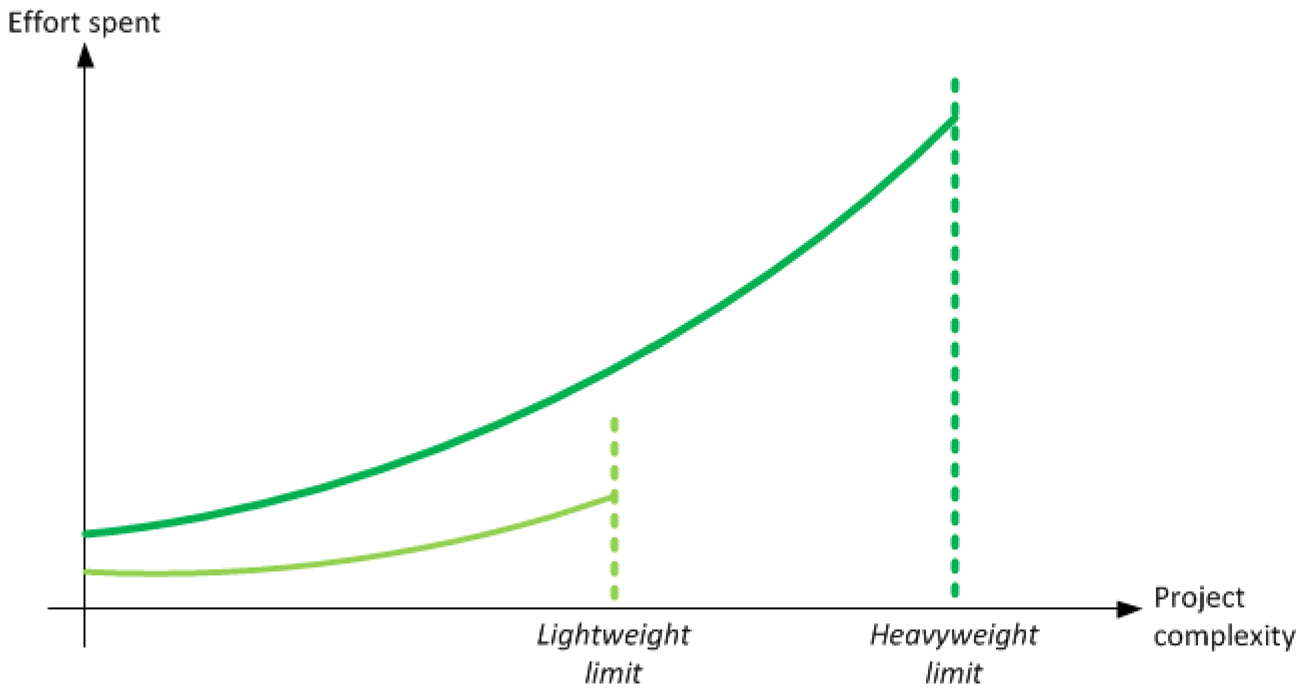
- Imagine you are the *developer* in a team. What would make you feel more comfortable – RUP or XP? Why?
- Now imagine you are the *customer*. What would make you feel more comfortable – RUP or XP? Why?
- Do you think it is *easier* to work in an XP project than in a RUP project?





Lightweight vs. Heavyweight methods

- So…are heavyweight methods *ever* justified?

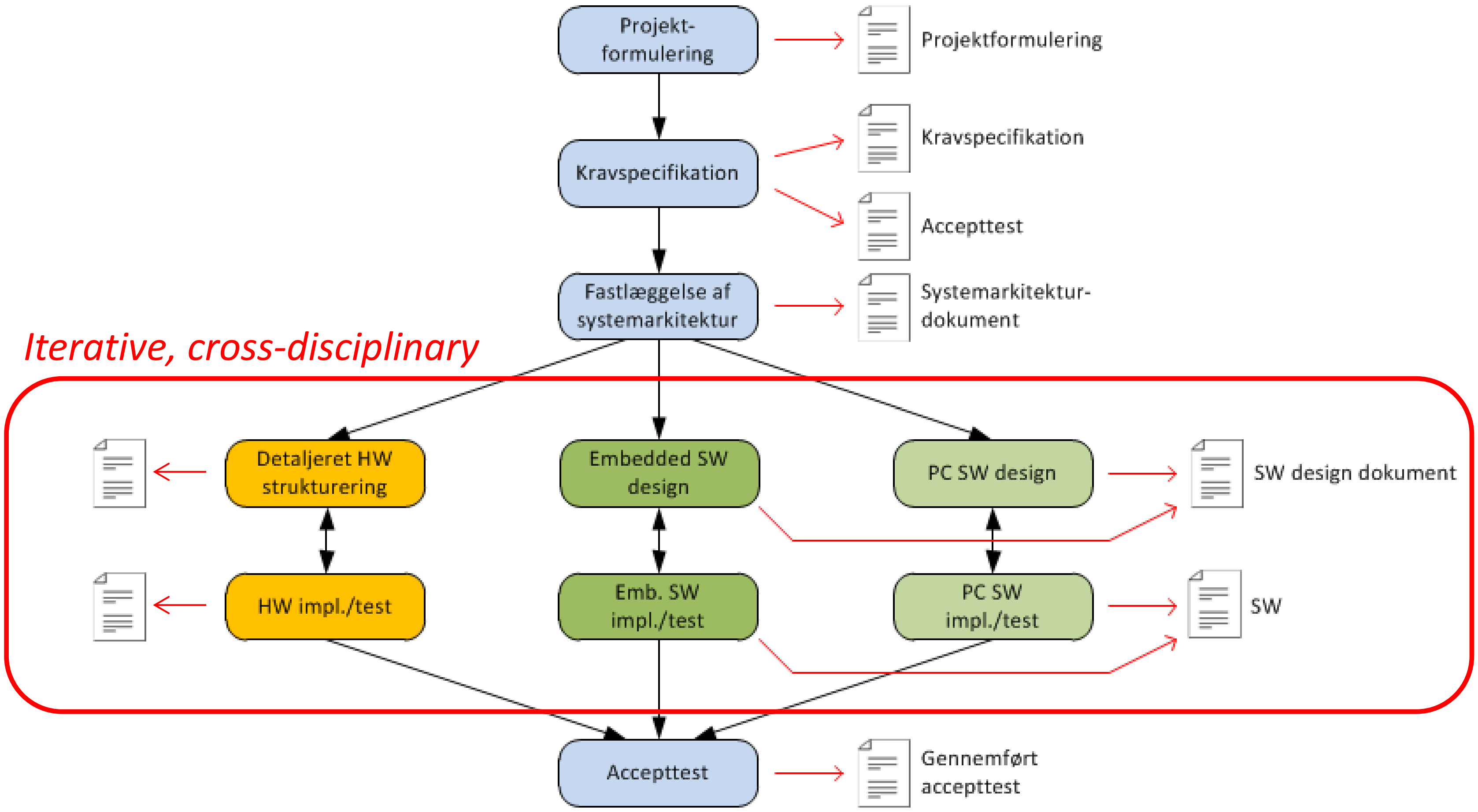


Last but not least: The ASE Process

- This is the process you are going to use in your semester project

- A *use case*-driven, ”middleweight” *semi-iterative* development process

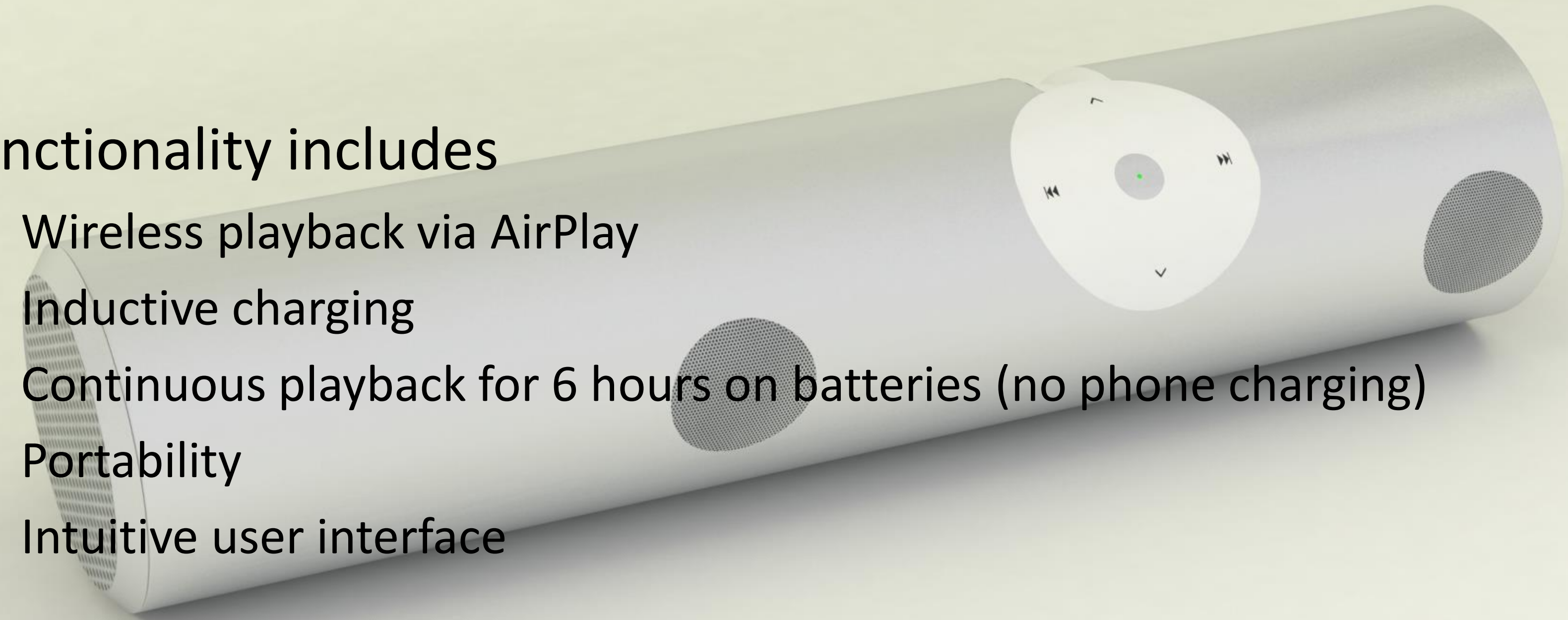
The ASE Process



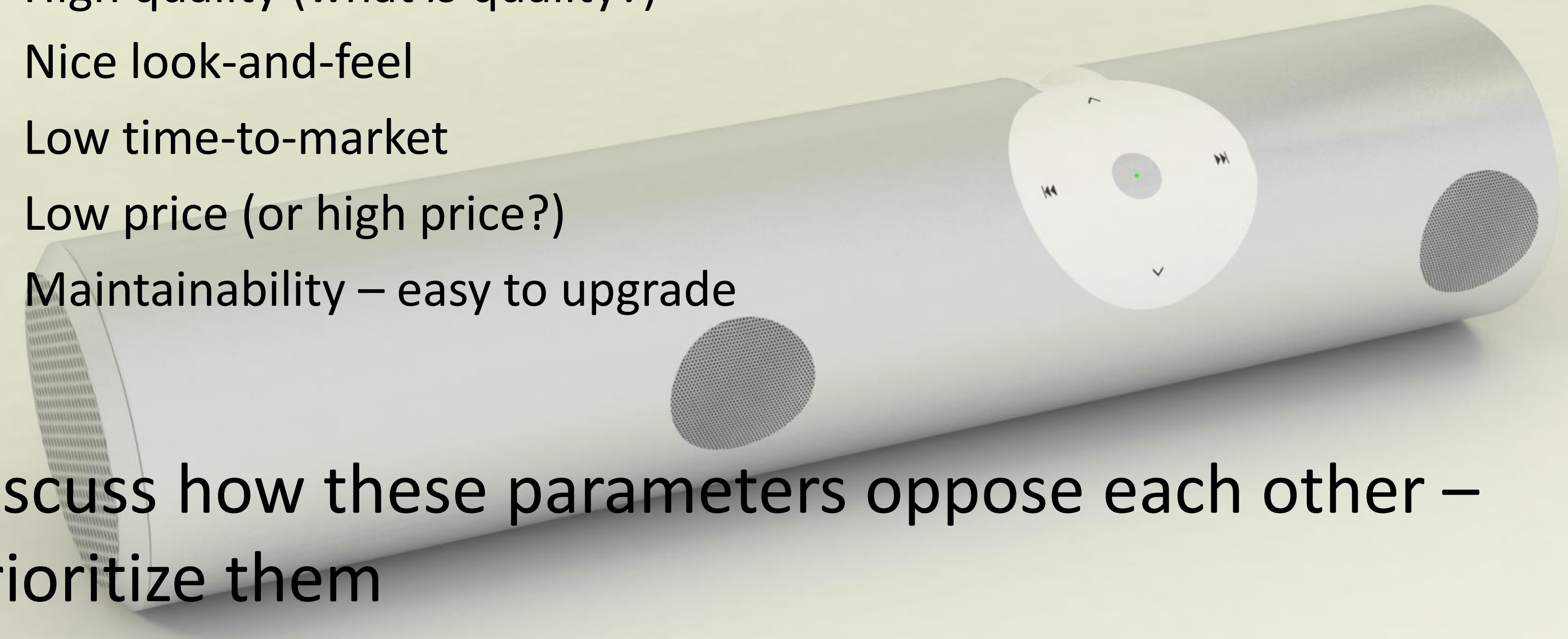
Case study

- Your team has been tasked to develop a new innovative dock for an Apple iPhone

- Functionality includes
  - Wireless playback via AirPlay
  - Inductive charging
  - Continuous playback for 6 hours on batteries (no phone charging)
  - Portability
  - Intuitive user interface



Case study

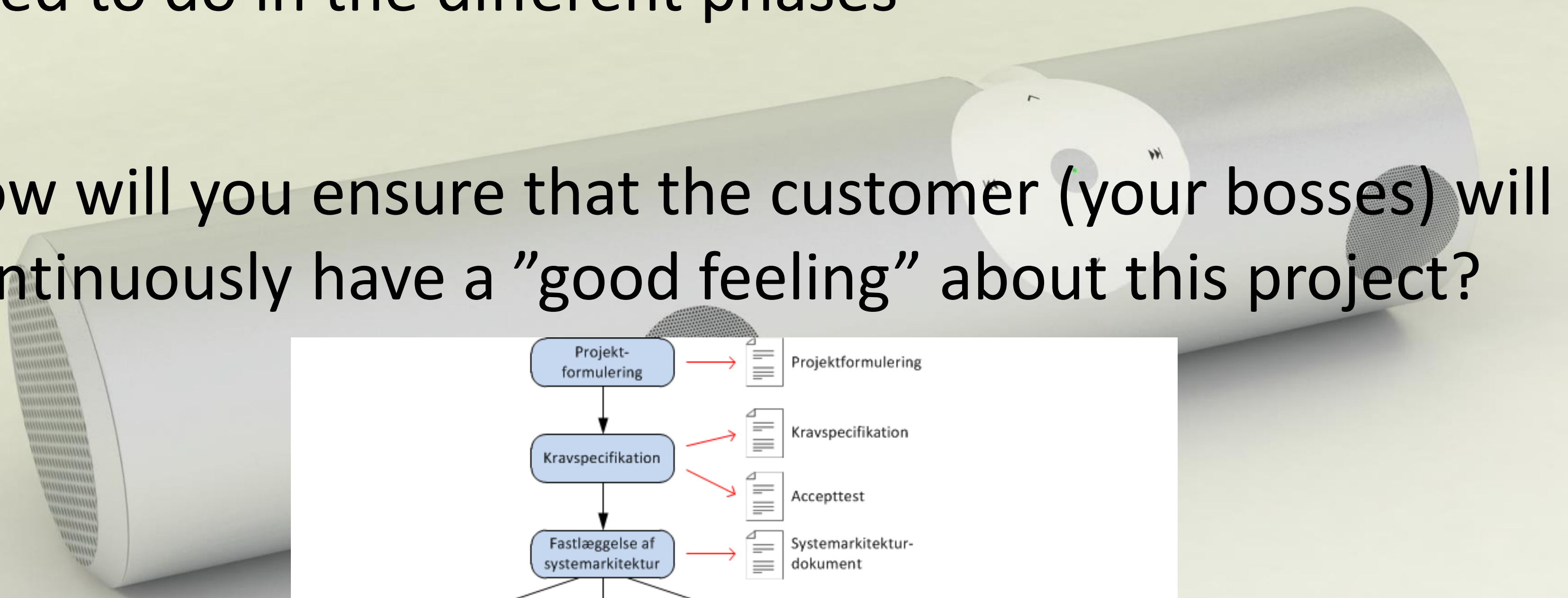


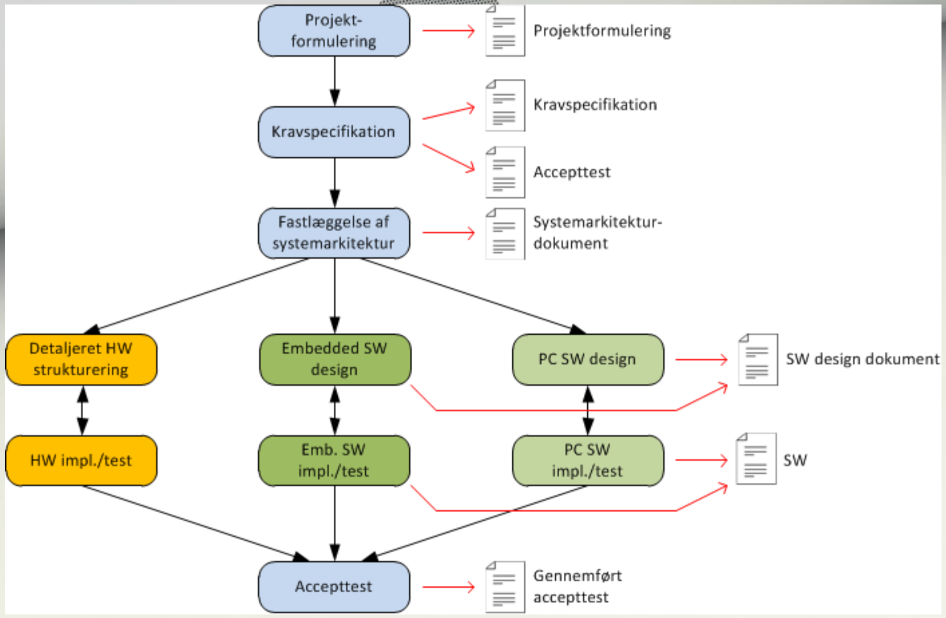
- Discuss the success parameters of the project, e.g.
  - High quality (what *is* quality?)
  - Nice look-and-feel
  - Low time-to-market
  - Low price (or high price?)
  - Maintainability – easy to upgrade
- Discuss how these parameters oppose each other – prioritize them

Case study

- In relation to the ASE Process phases: Discuss what you w
need to do in the different phases

- How will you ensure that the customer (your bosses) will
continuously have a ”good feeling” about this project?




Projekt-formulering
Projektformulering
Kravspecifikation
Kravspecifikation
Accepttest
Fastlæggelse af systemarkitektur
Systemarkitektur-dokument
Detaljeret HW strukturering
Embedded SW design
PC SW design
SW design dokument
HW impl./test
Emb. SW impl./test
PC SW impl./test
SW
Accepttest
Gennemført accepttest