

Development Processes

Introduction to Systems Engineering
I2ISE

Introduction

- What is a (development) process?
- Why do we need a development process?
- The *Bills of Rights*
- Some examples
 - Traditional, iterative, agile and the ASE processes

What is a development process?

- A *process* is the action of taking something through a defined set of steps to transform something into something else
 - Milk → cheese, metal → car, thoughts → products, etc.
- A *development process* is a process defined to support development (of HW, SW, ...)
- A development process may define...
 - How to arrive at a product
 - What input is needed at what times
 - What (secondary) output there should be
 - ...

Why use a development process?

- Using a development process may seem to incur an overhead
 - E.g., you may not actually “produce” anything before “late” in the process
- So...why do we use it?
- Because we are engineers, so we are *concerned*
 - ...that we are producing the right thing
 - ...with the right capabilities
 - ...at the right time
 - ...at the right cost
 - ...

Why use a development process?

- Development processes answers some important questions:
 - What are you going to *produce*?
 - When will you be *done*?
 - What will it *cost*?
 - How will you handle *changes*?
- Answers to these questions are important to the customer
- Are the answers important to you? To your business? Why?

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* that both the Developers' and Customer's Bill of Rights are respected.



- 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 the schedule*.
- You have the right to be informed of *schedule changes*, in time to choose how to reduce the impact or 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

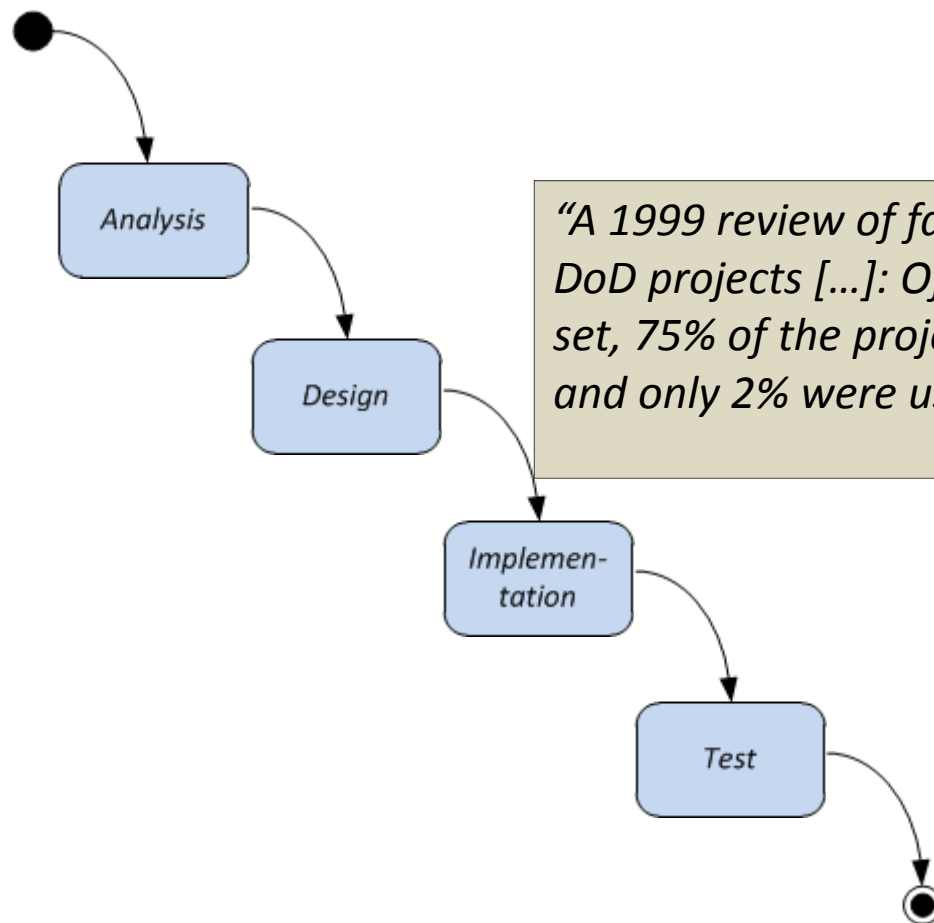
- A claim: The process is successful if *and only if* it preserves the rights and meets the goals

Examples:

Traditional development processes

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

The waterfall process



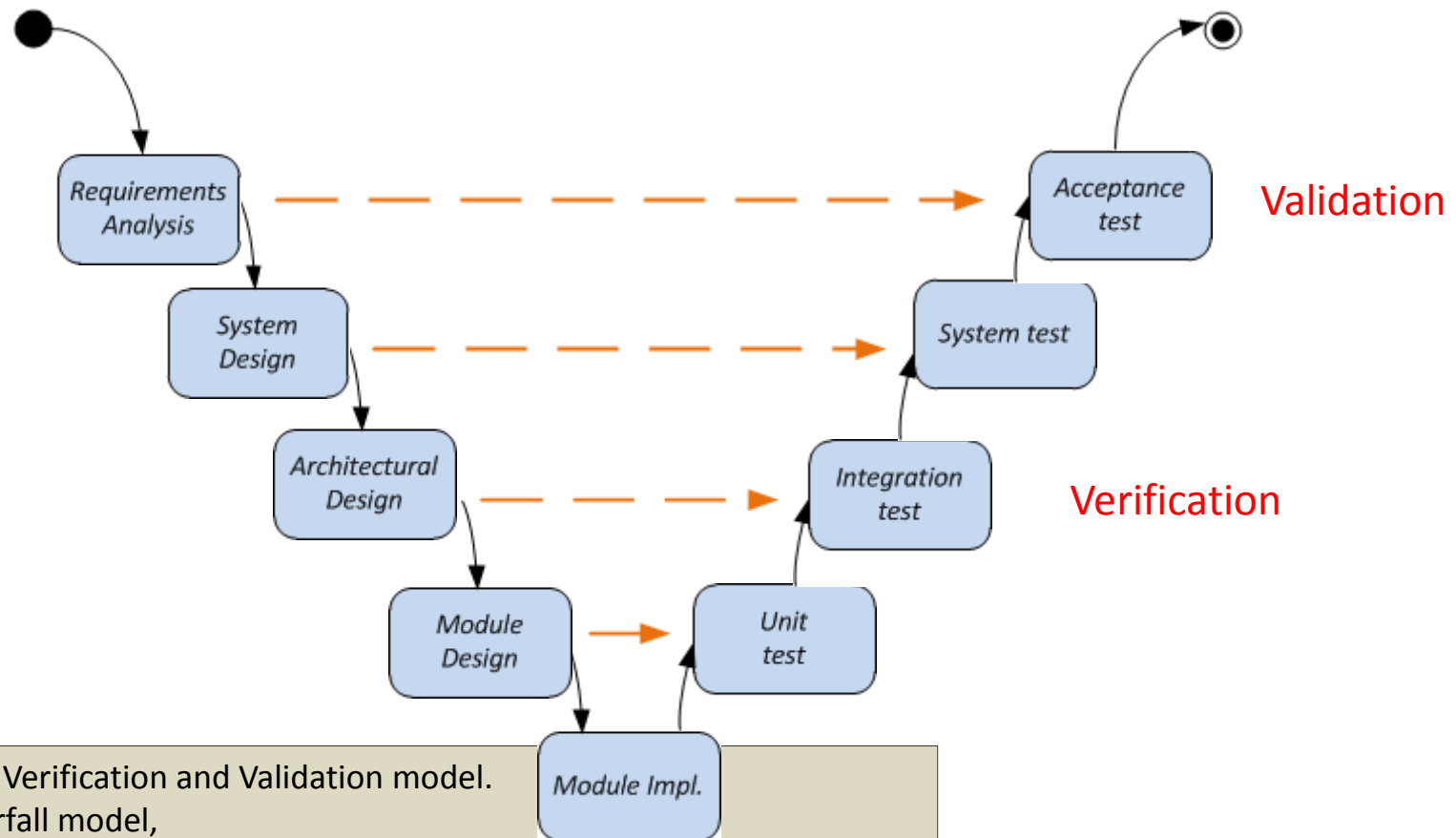
"...a flawed, non-working model"

-Winston R. Royce, 1970

"A 1999 review of failure rates in a sample of earlier DoD projects [...]: Of a total \$37 billion for the sample set, 75% of the projects failed or were never used, and only 2% were used without extensive modification."

- S. Jarzombek, 1999

The V-model



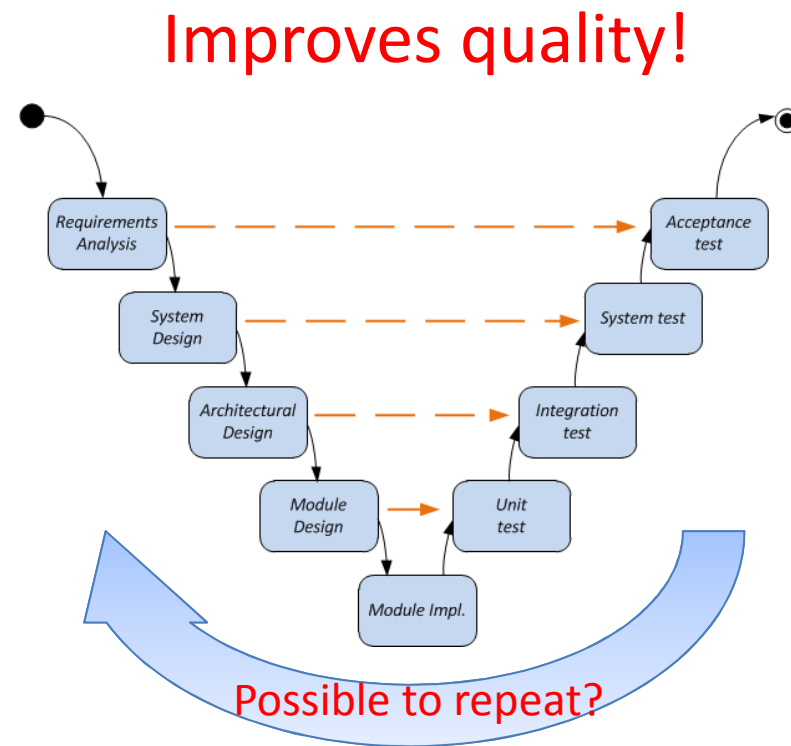
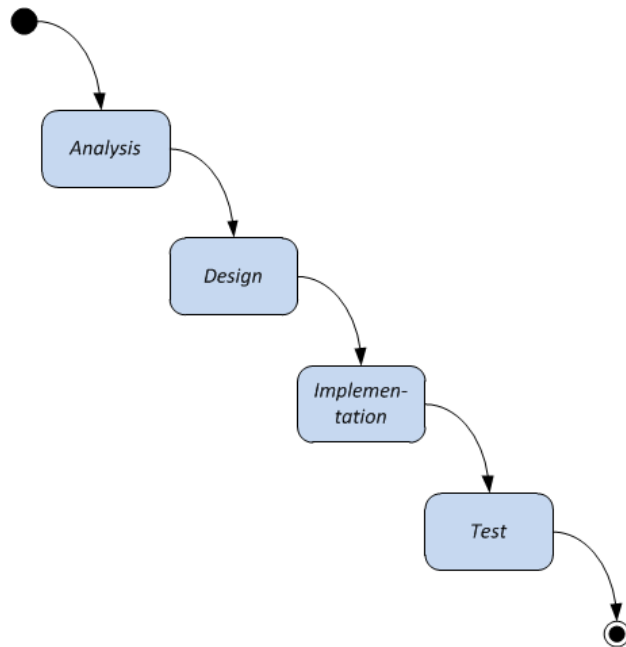
“V- model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development” .. *Try to improve quality*
- ISTQB Certification

When to use the V-model?

- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
- The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.
- High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

Discussion

- What is the difference?

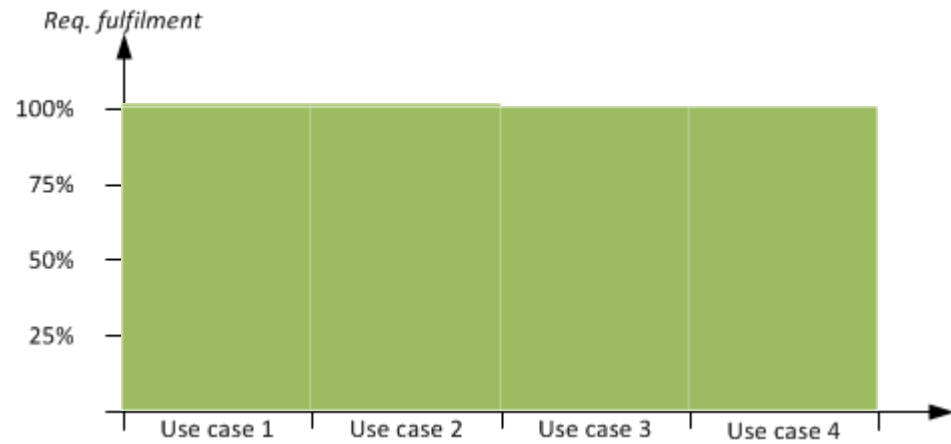


Iterative and incremental development processes

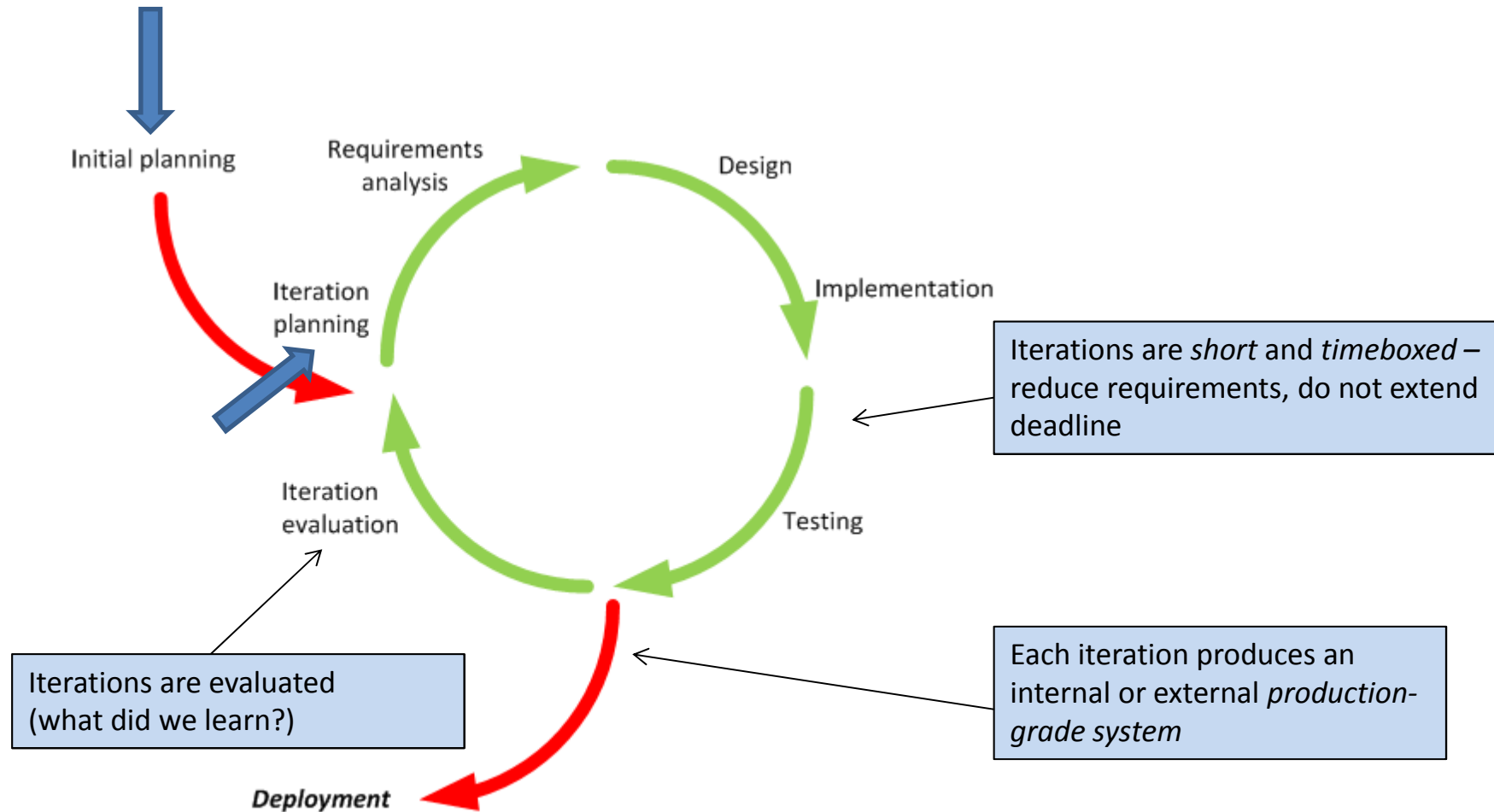
- *Iterative* refers to the repetitive nature of the process
 - An *iteration* is a single repetition of the same sub-process.
 - The sub-process result is a partial working system of *production-quality*
- *Incremental* refers to the *continued expansion* of system capabilities.

Iterative vs. incremental

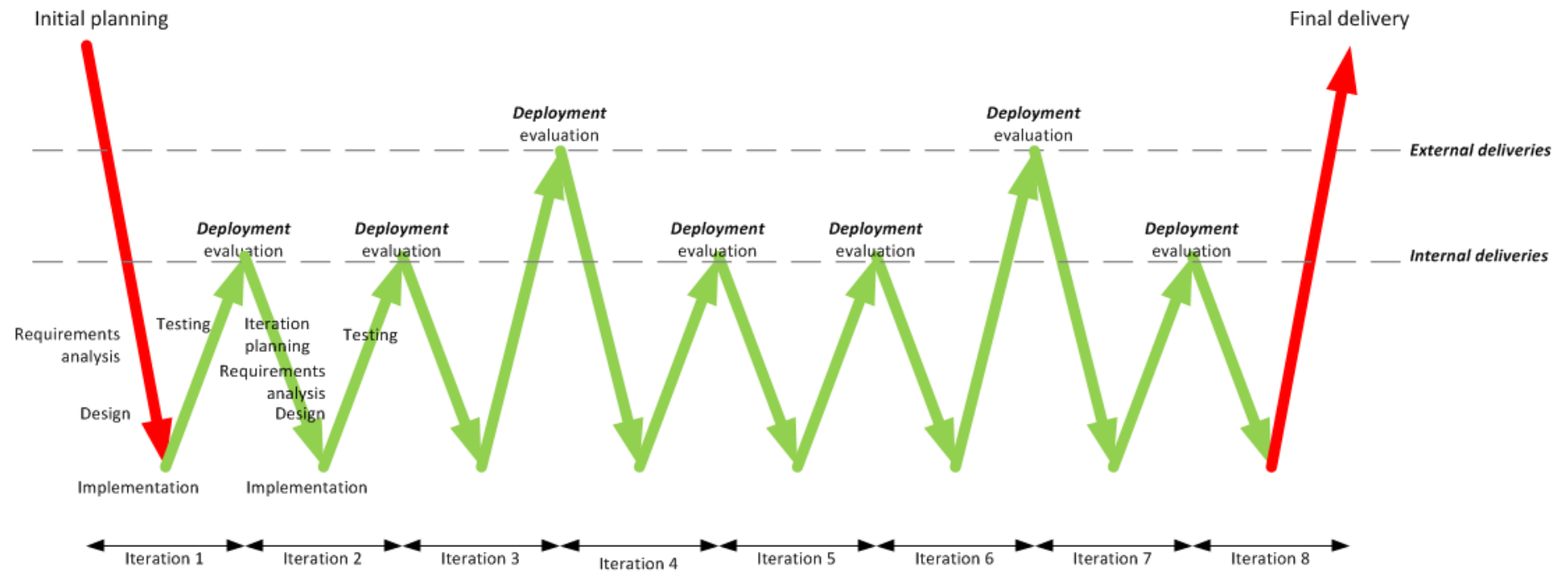
- Iterative *and* incremental



Iterations

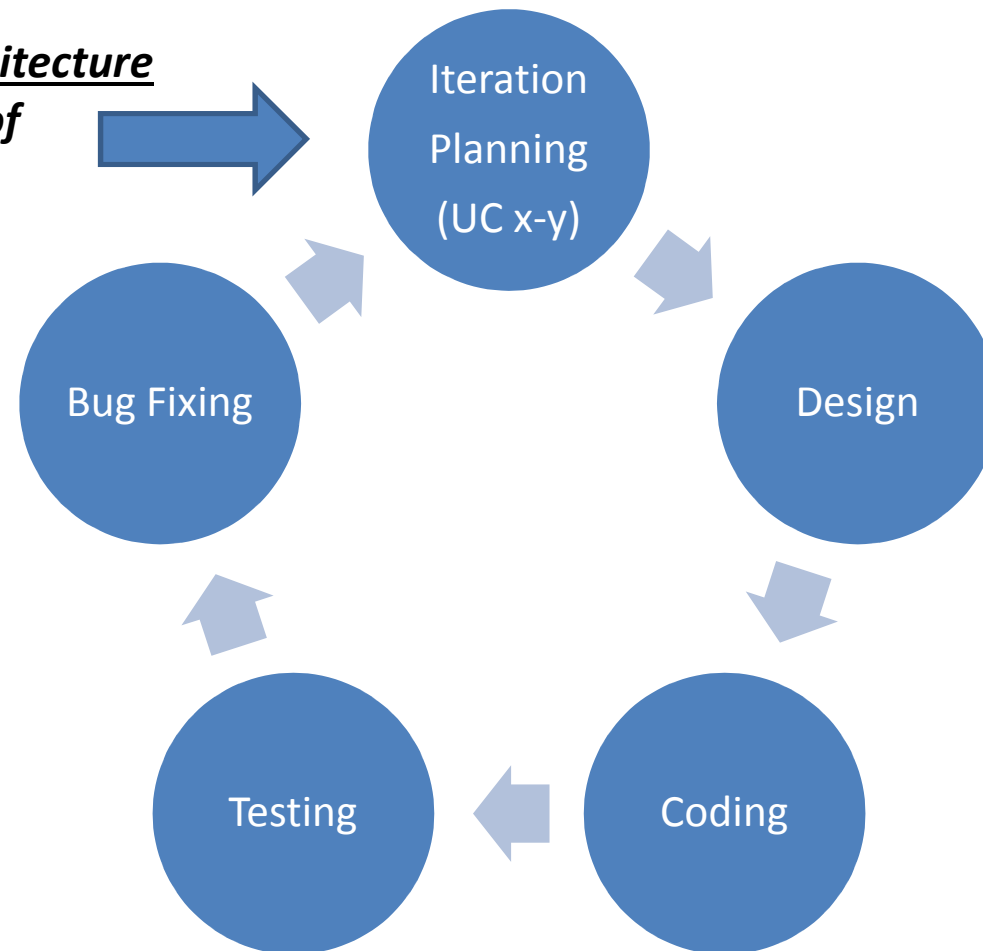


Iterations – another view



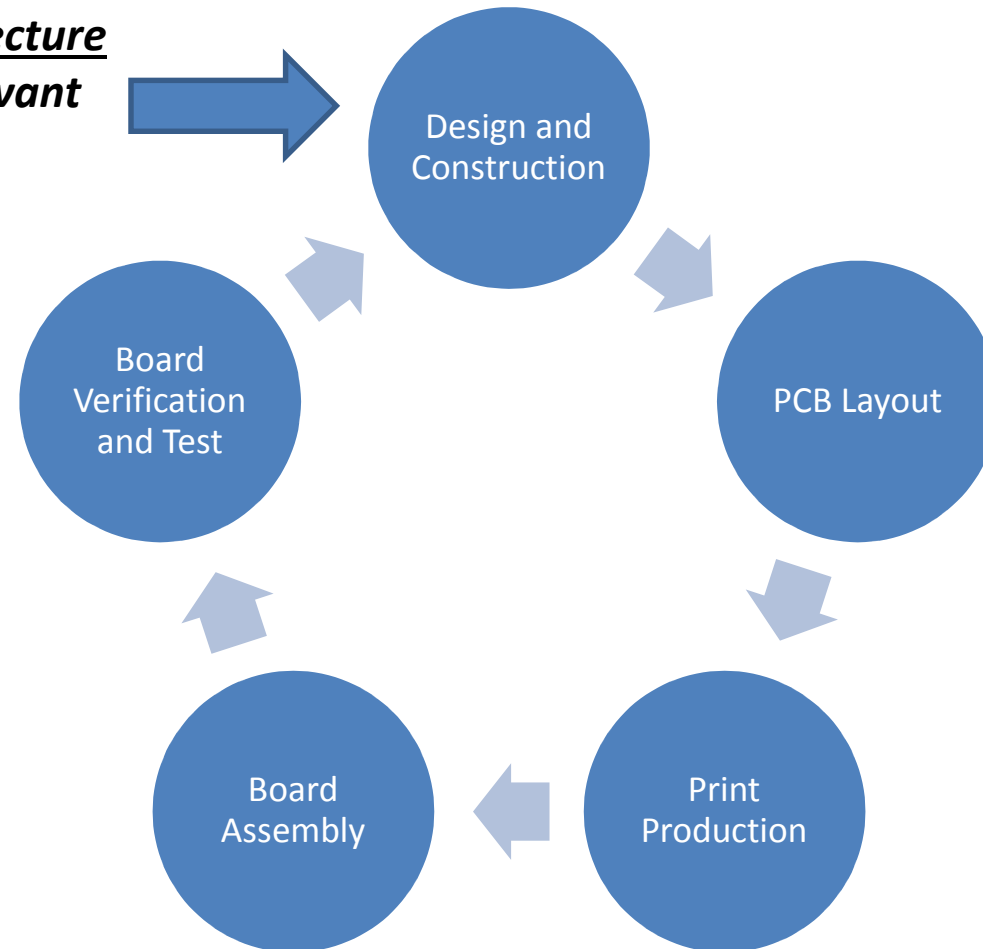
Typical SW Iterations

Specification/Architecture
***Selected number of
Use Cases (UC)***



Typical HW Board Spins (Iterations)

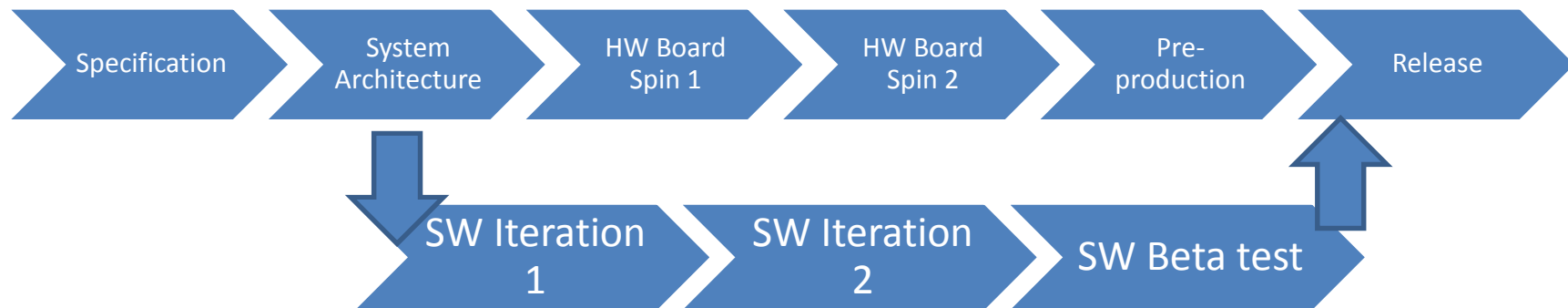
Specification/Architecture
Fulfilling all HW relevant requirements (UC)



Embedded (HW/SW) Development Overall Project Plan

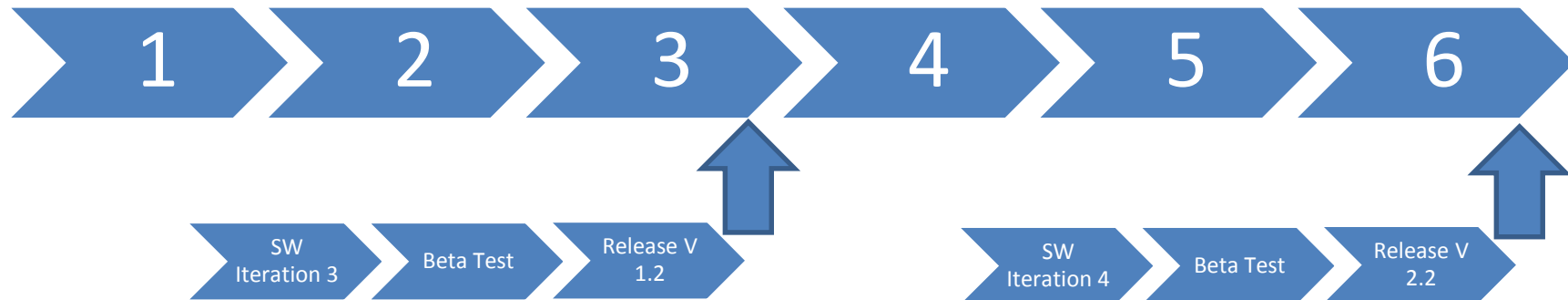
Project start

Project end



Embedded (SW) Development Product Life-time

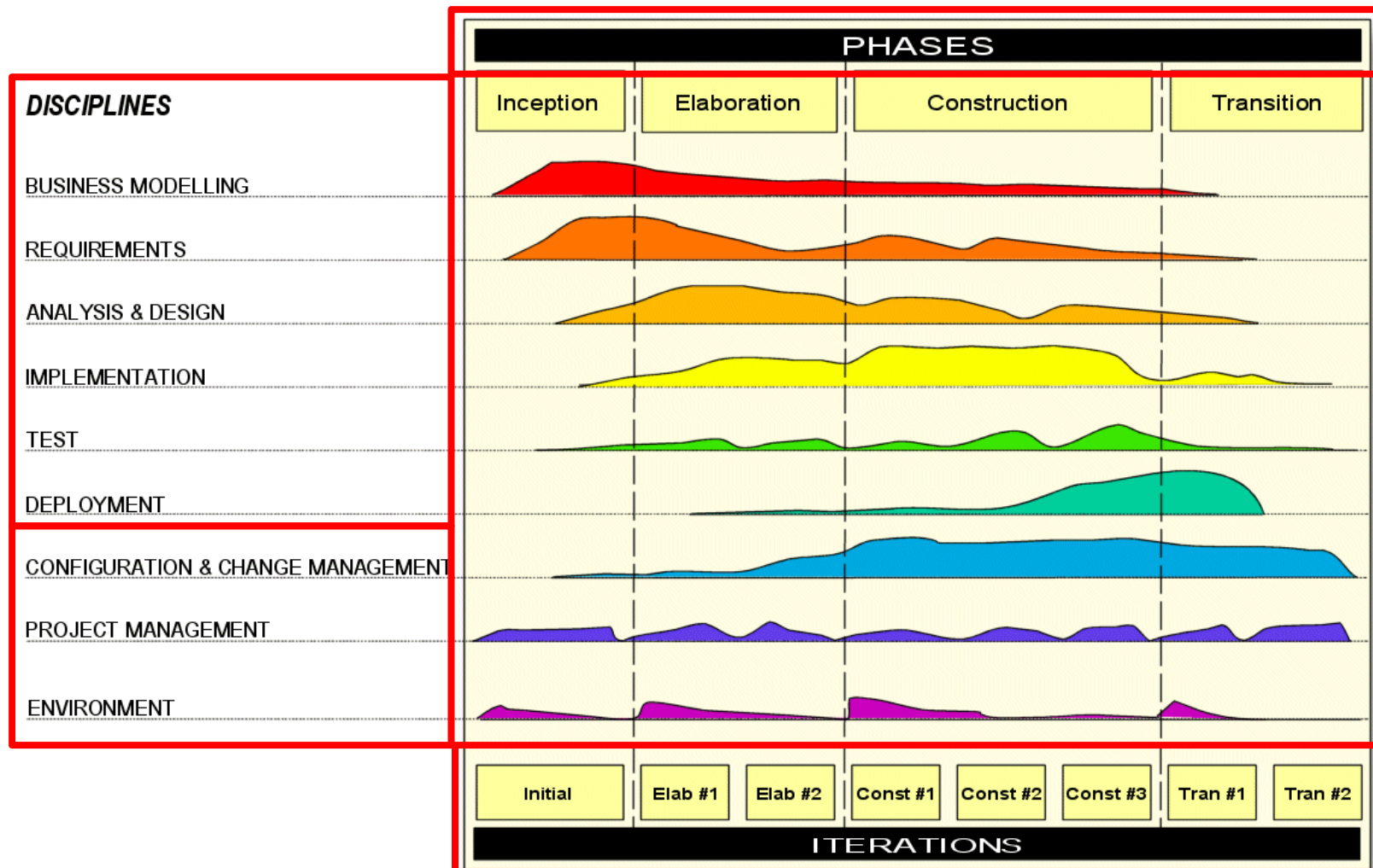
Product life (months)



Example: Rational Unified Process

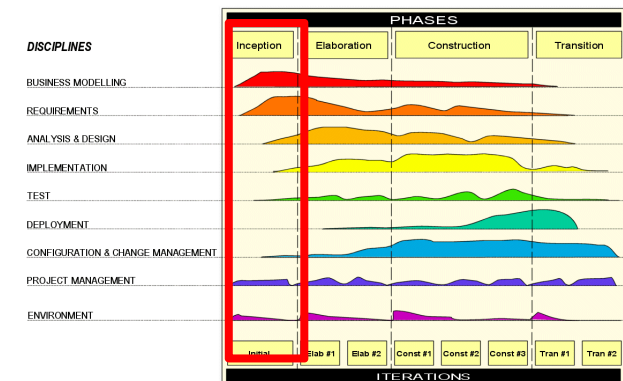
- Rational Unified Process (RUP)
 - Developed by *Rational Software* (now IBM)
 - Developed from the *Unified Process*
Jacobson, Booch, Rumbaugh
- Actually a process *framework* from which processes can be *instantiated*

RUP: The works



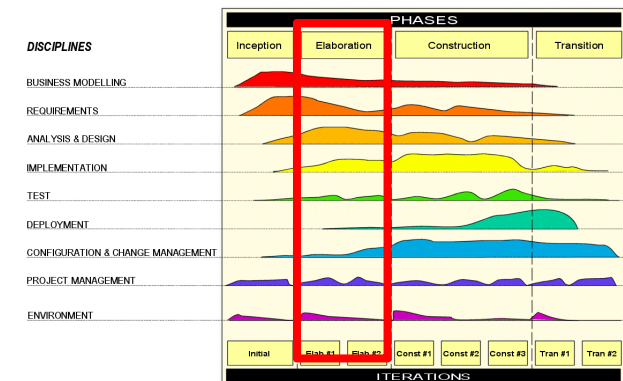
RUP: Inception phase

- 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



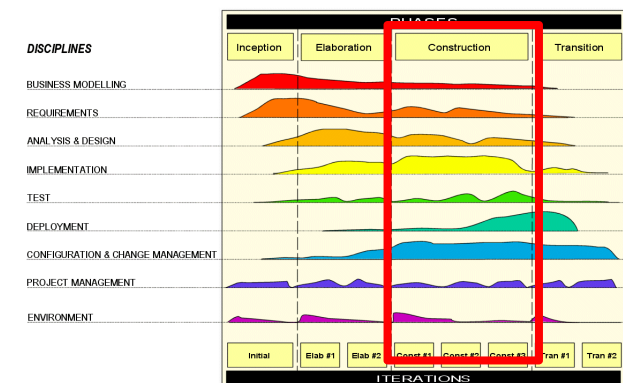
RUP: Elaboration phase

- 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



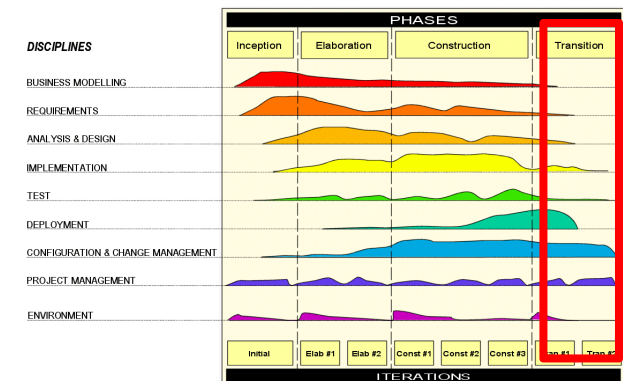
RUP: Construction phase

- 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.



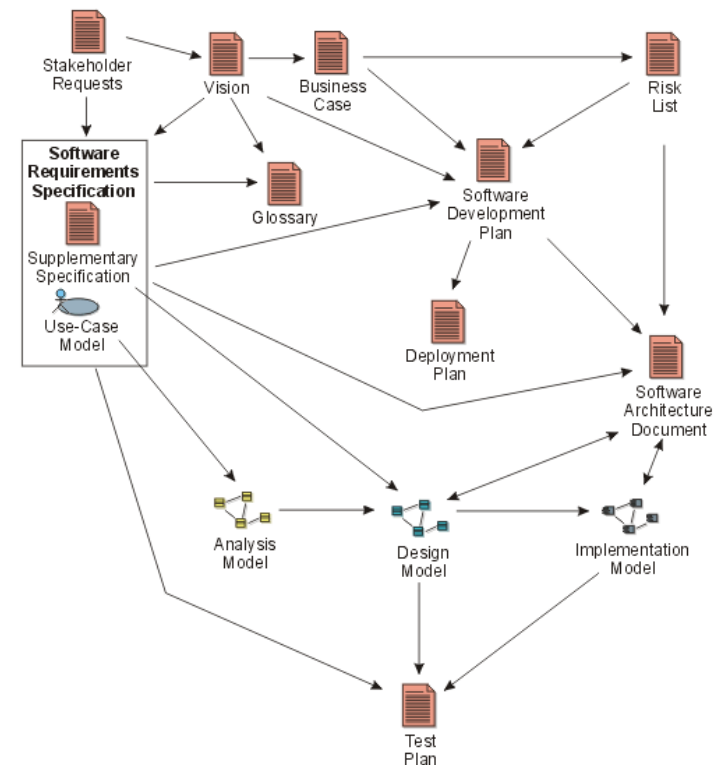
RUP: Transition phase

- 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?



What's the problem?

Disciplined execution
kills innovation



Innovation requires
"no discipline"



*Disciplined
execution*

Plans, deadlines,
documents

*Continuous
innovation*

Open environment,
no "management"

Agile methods

- Acknowledging that the traditional processes are fundamentally flawed and that many iterative processes are heavy, *agile* processes emerged in the 1990's.
- Defined in the *agile manifesto* in 2001 by 17 signatories.
- The agile "bottom line": Faith in *people* rather than *paper*
- ***Mostly used for pure software development***

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 – how do you climb a mountain?

“Like a climber planning a route over glaciers and up a mountain face, the destination is clear, but getting a safe route up the mountain and back down again takes a mixture of careful planning and an adaptive approach to events. The climbers may find impassable chasms, dangerous overhangs or unpredictable changes in weather. The plan will probably change”

- P.G. Armour, "The Laws of the Software Process"

Agile methods:

Some of the 12 agile principles

- Satisfy the customer through early and continuous delivery of valuable software
- Welcome changing requirements, even late in development
- Business people and developers must work together daily throughout the project
- Working software is the primary measure of progress
- Simplicity – the art of maximizing the amount of work not done – is essential

Example: eXtreme Programming (XP) (Software)

- Developed by Beck, Cunningham and others
 - First coined in 1996
- Some characteristics:
 - Focus on *customer satisfaction*
 - Permanent on-site *customer presence*
 - Short development cycles
 - Incremental planning
 - Continuous feedback
 - Evolutionary design
 - Pair programming

XP values

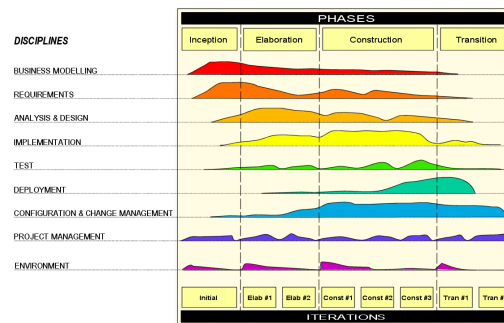
- XP is based on four *values* that governs all work:
 - **Simplicity** Do what's asked – no more!
 - **Communication:** Everyone is part of the team and will communicate daily, face-to-face
 - **Feedback:** Demonstrate early and often, listen to feedback, make changes
 - **Courage:** Have the courage to change what needs to be changed, and to respond to changing requirements

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

Discussion

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



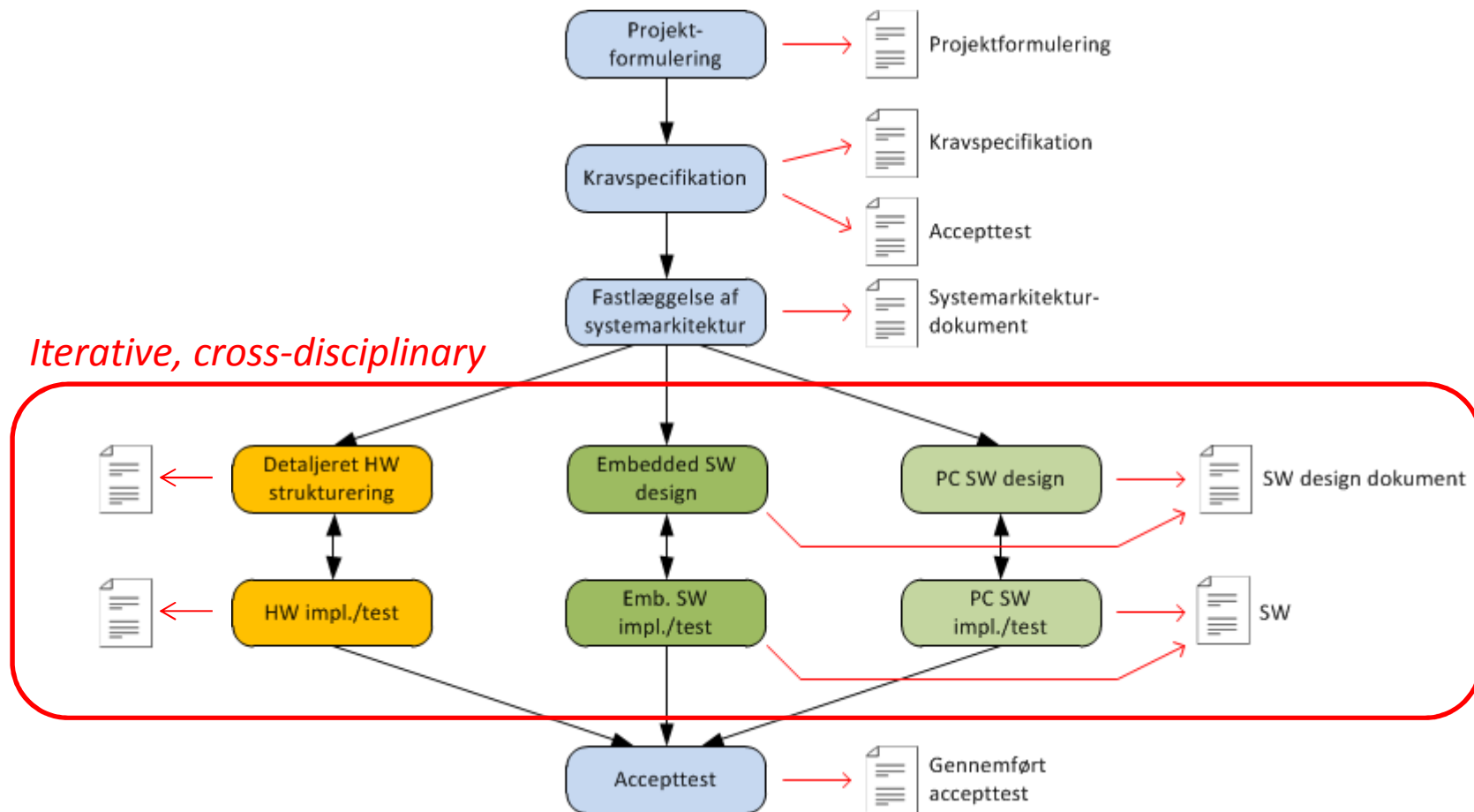
The state of things

- The days of the standard process are over...
- The most commonly seen process today is tailored to meet the business needs
- Usually the process will be highly iterative with selected agile elements (team, iterations, customer involvement, etc.)
- Usually, it will be managed by Scrum (which we'll learn about later)

Last but not least: The ASE Process

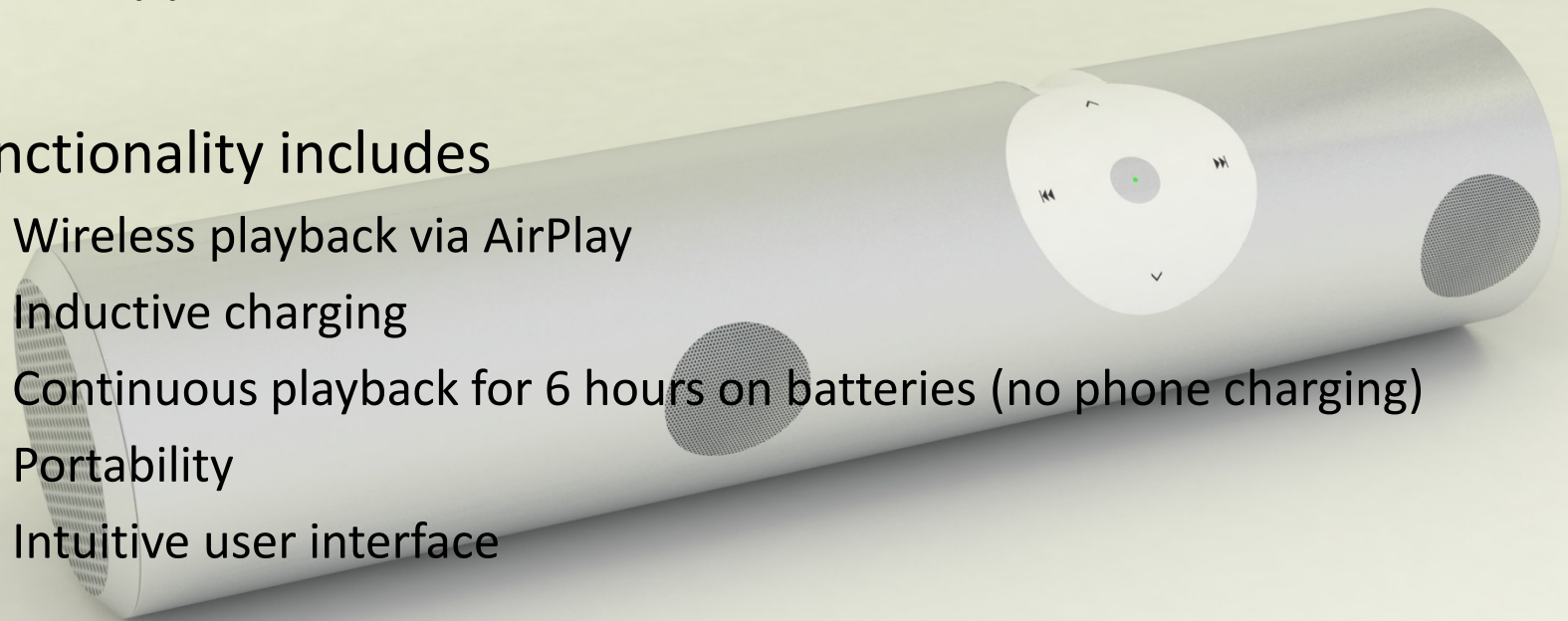
- This is the process you are going to use in your semester project
- A *use case*-driven, "middleweight" *semi-iterative* development process
- Accounts for both *hardware* and *software* development
 - Here the essential architecture needs to be fixed early in the project

The ASE Process



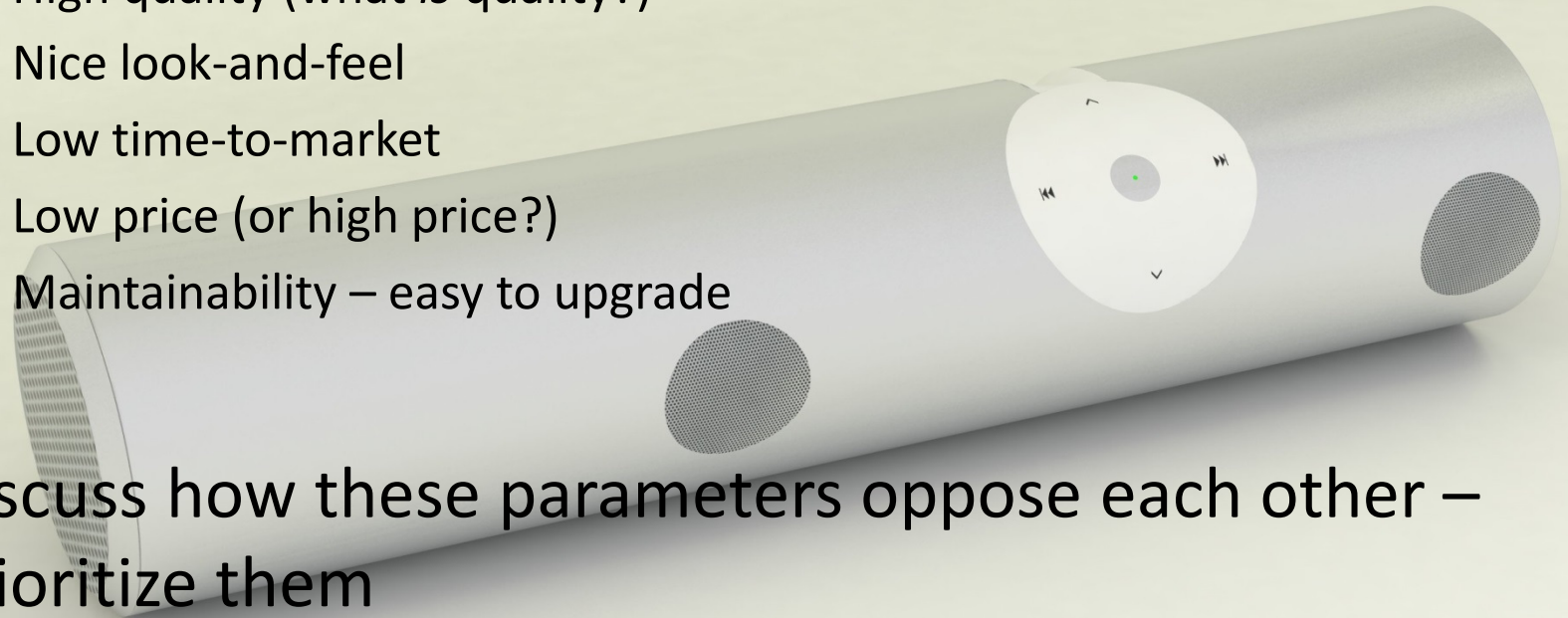
Case study

- 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 will 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?

