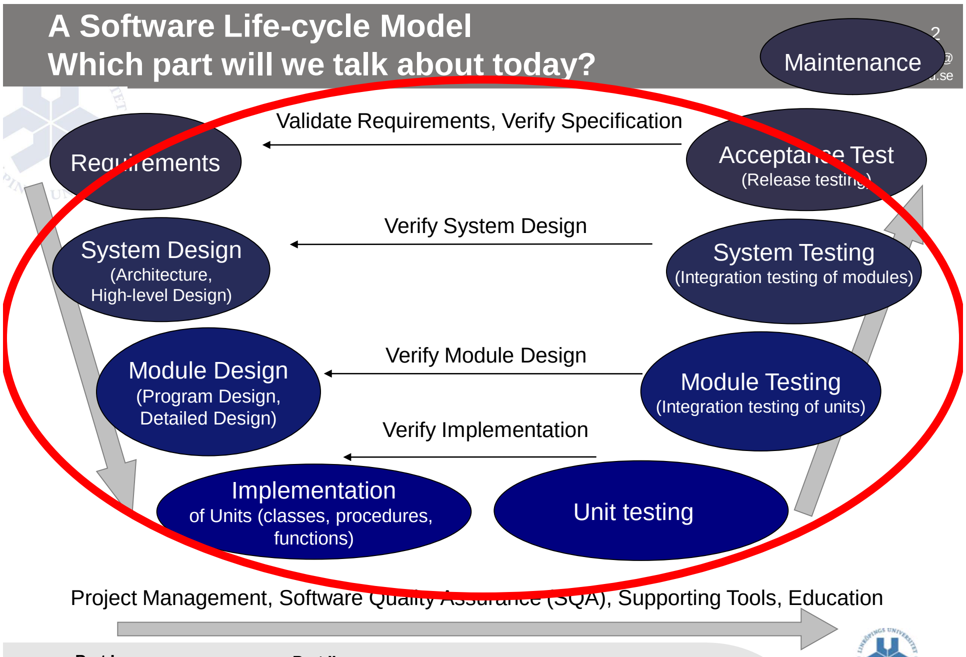
Software Life Cycles and Processes

Lecture 2

Software Engineering
CUGS course
Spring 2011

KristianSandahl, slides by David Broman
Department of Computer and Information Science
Linköping University, Sweden
Kristian.Sandahl@ida.liu.se

A Software Life-cycle Model
Which part will we talk about today?

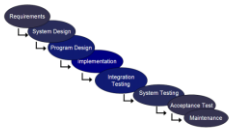


Part I
Life Cycles and
Process Models

Part II
Methodologies and Processes

Agenda - What will you learn today?

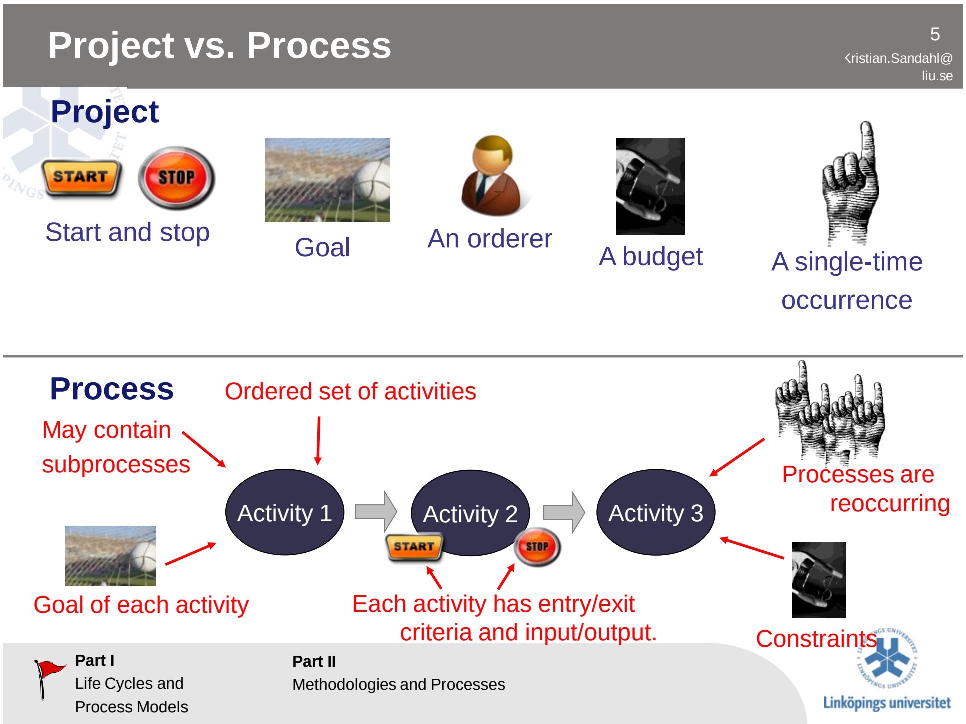
Part I
Life Cycles and Process Models



Part II
Methodologies and Processes

Part I

Life Cycles and Process Models



Project vs. Process

Project

- Start and stop
- Goal
- An orderer
- A budget
- A single-time occurrence

Process

Ordered set of activities

May contain subprocesses

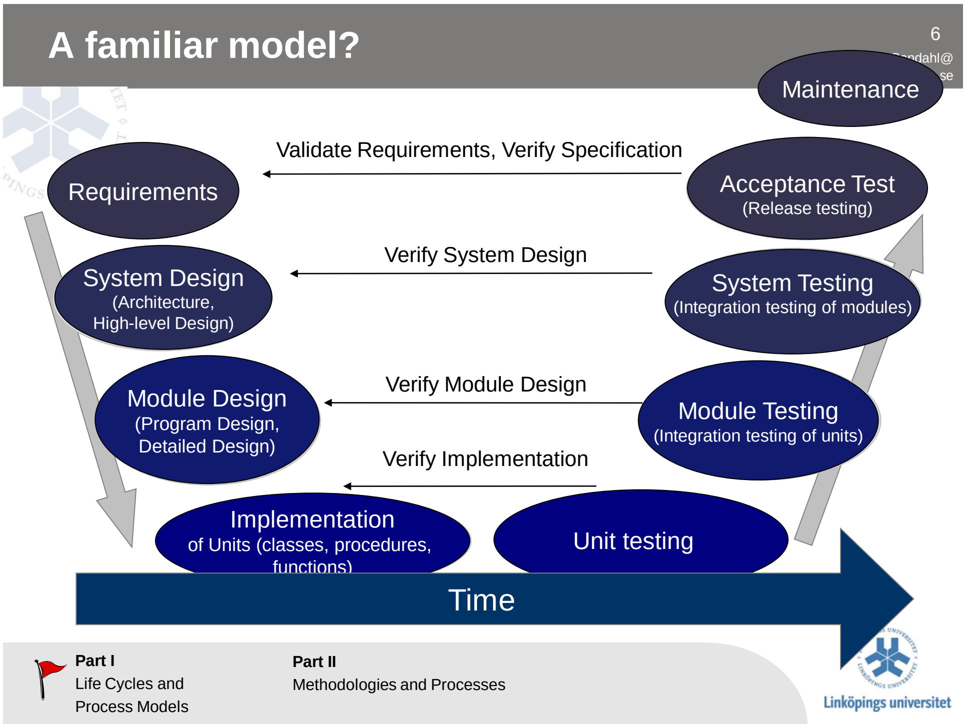
Activity 1 → Activity 2 → Activity 3

Goal of each activity

Each activity has entry/exit criteria and input/output.

Processes are reoccurring

Constraints



A familiar model?

Requirements

System Design
(Architecture, High-level Design)

Module Design
(Program Design, Detailed Design)

Implementation
of Units (classes, procedures, functions)

Unit testing

Module Testing
(Integration testing of units)

System Testing
(Integration testing of modules)

Acceptance Test
(Release testing)

Maintenance

Validate Requirements, Verify Specification

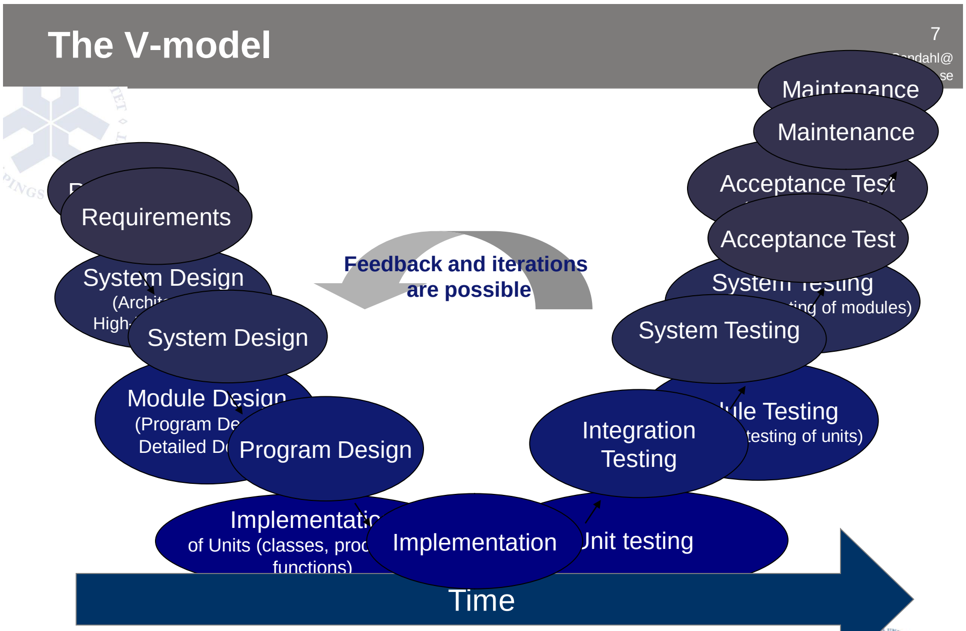
Verify System Design

Verify Module Design

Verify Implementation

Time

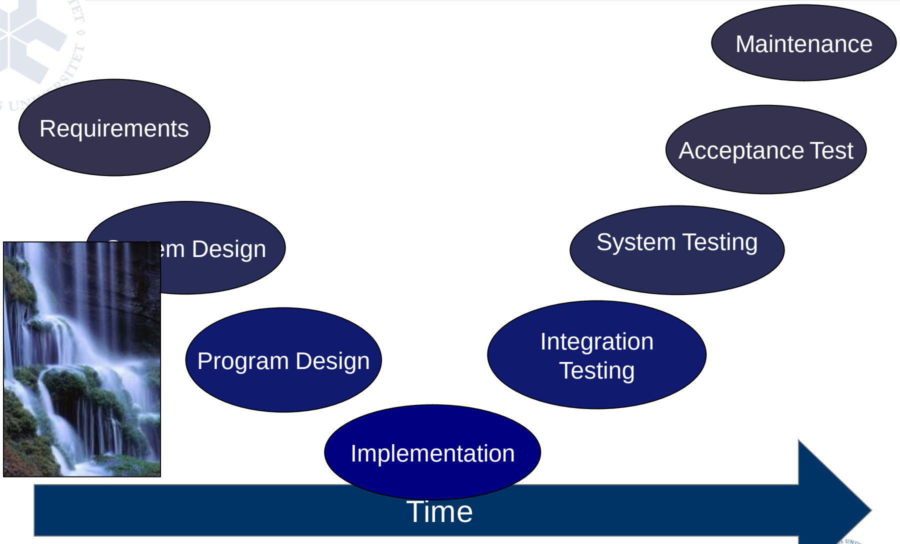
The V-model



Part I
Life Cycles and
Process Models

Part II
Methodologies and Processes

Another model...

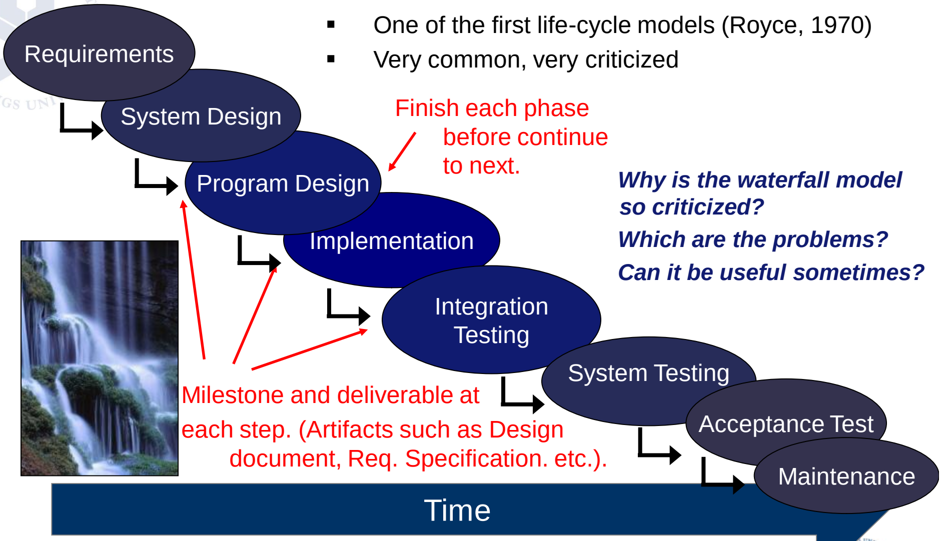


Part I
Life Cycles and
Process Models

Part II
Methodologies and Processes

The Waterfall model

- One of the first life-cycle models (Royce, 1970)
- Very common, very criticized



Finish each phase before continue to next.

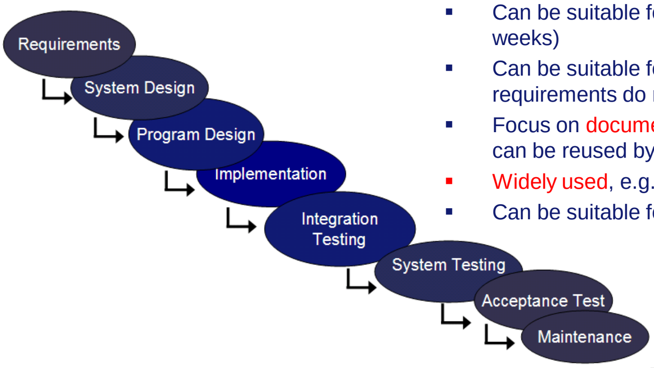
Milestone and deliverable at each step. (Artifacts such as Design document, Req. Specification. etc.).

Why is the waterfall model so criticized?

Which are the problems?

Can it be useful sometimes?

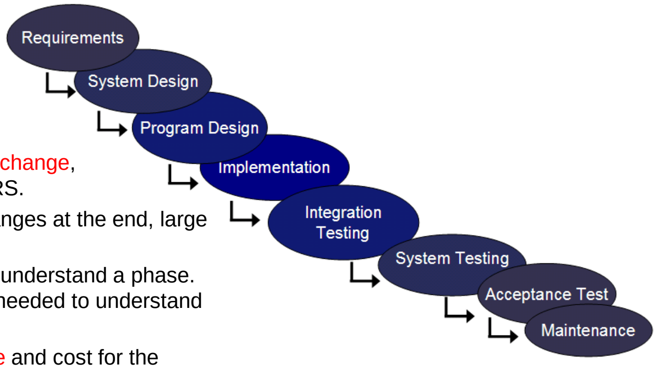
The Waterfall model - some arguments



Pros

- Simple, manageable and easy to understand
- Fits to common project management practices (milestones, deliverables etc.)
- Focus on requirements and design at beginning, save money and time at the end
- Can be suitable for short projects (some weeks)
- Can be suitable for "stable" projects, where requirements do not change
- Focus on documents, saves knowledge which can be reused by other people.
- Widely used, e.g. US Department of Defense
- Can be suitable for fixed-price contracts

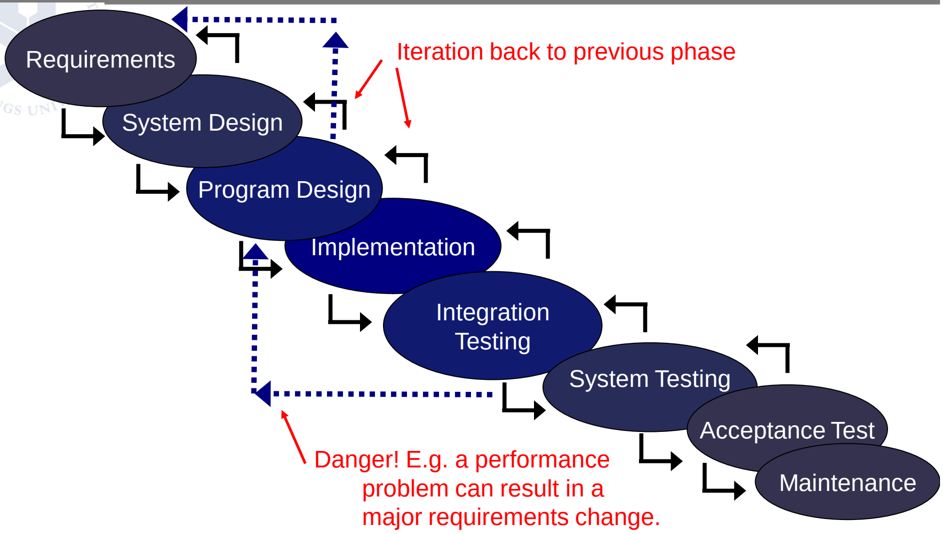
The Waterfall model - some arguments



Cons

- Software requirements change, hard to sign-off on a SRS.
- Early commitment. Changes at the end, large impact.
- Feedback is needed to understand a phase. E.g. implementation is needed to understand some design.
- Difficult to estimate time and cost for the phases.
- Handling risks are not part of the model. Pushes the risks forward.
- Software "is not" developed in such a way. It evolves when problems are more understood. Little room for problem solving.

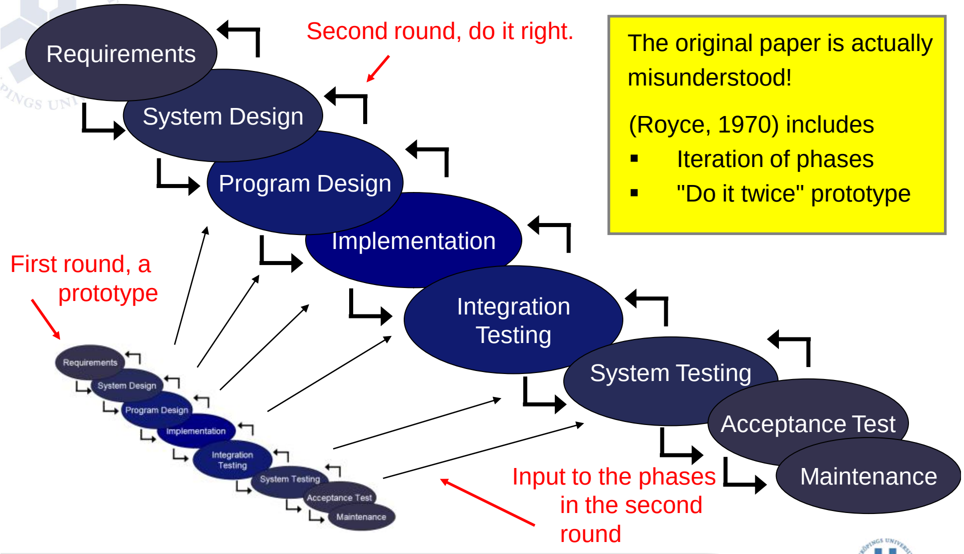
Can we improve the model?



Iteration back to previous phase

Danger! E.g. a performance problem can result in a major requirements change. Very expensive rollback...

Do it twice?



Second round, do it right.

First round, a prototype

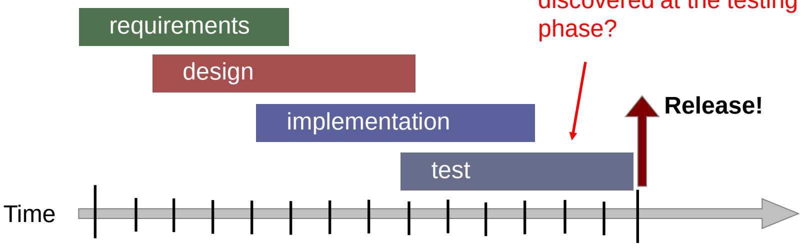
Input to the phases in the second round

The original paper is actually misunderstood!

(Royce, 1970) includes

- Iteration of phases
- "Do it twice" prototype

Is overlapping phases a solution?

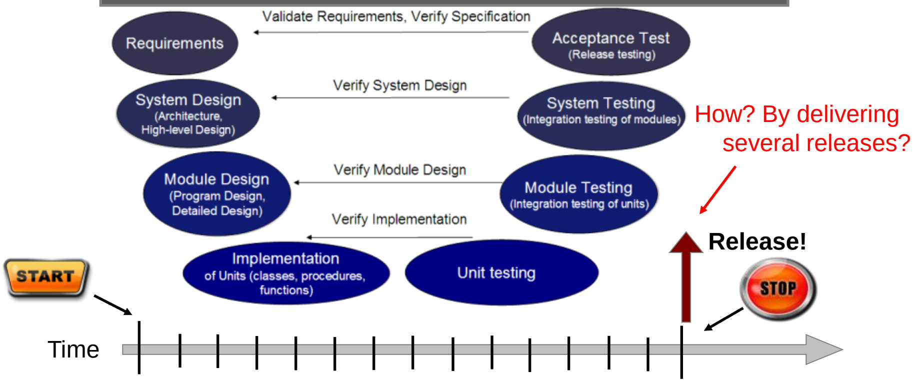


When do we "sign-off",
e.g. when do we have all requirements?

What if a major design flaw is discovered at the testing phase?

What should be built?

> *”The hardest single part of building a software system is deciding precisely what to build”*
> **(Frederick P. Brooks)**

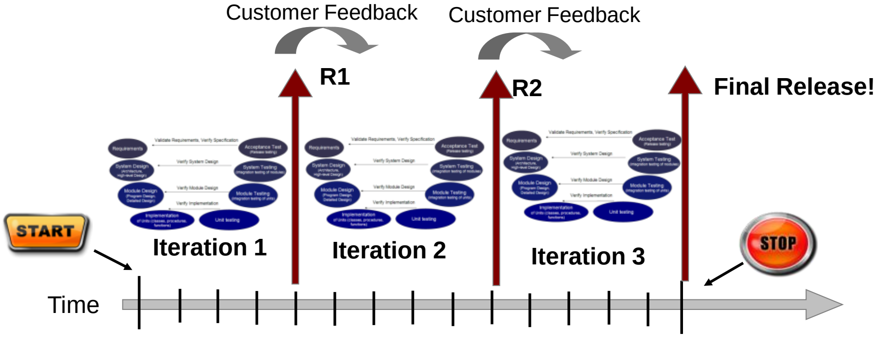


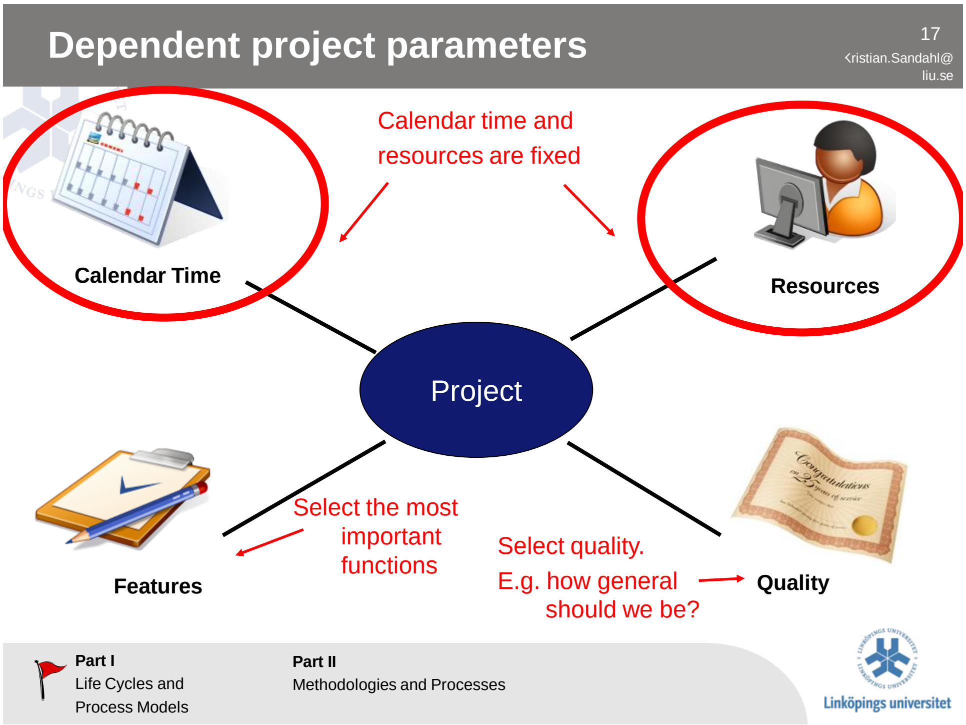
How? By delivering several releases?

Iterative Development

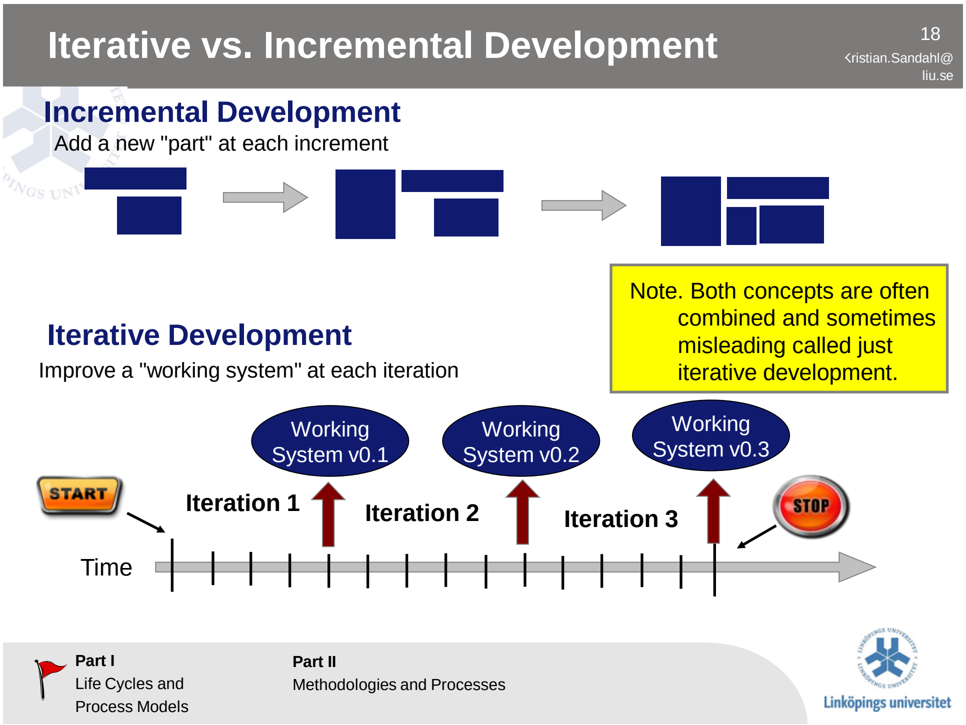
When **should the releases take place?**
Time-boxing - The time period is fixed for each iteration.

What **should be included in the release?**
Prioritized functionality - Do the most important parts first.





Dependent project parameters



Iterative vs. Incremental Development

Incremental Development

Add a new "part" at each increment

Note. Both concepts are often combined and sometimes misleading called just iterative development.

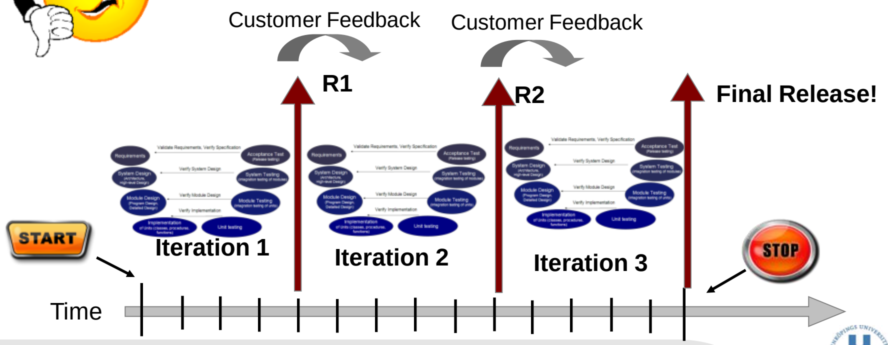
Iterative Development

Improve a "working system" at each iteration

Iterative Development - Cons

Is iterative development the silver bullet?

- Problem with current business contracts, especially fixed-price contracts.
- With short iterations it can be hard to map customer requirements to iterations.



Iterative Development - Pros

Pros

- Misunderstandings and inconsistency are made clear early (e.g. between requirement, design, and implementation)
- Encourage to use feedback -> elicit the *real* requirements
- Forced to focus on the most critical issues
- Continuous testing offers project assessment
- Workload is spread out over time (especially test)
- The team can get "lesson learned" and continuously improve the process
- Stakeholders gets concrete evidence of progress

Part II

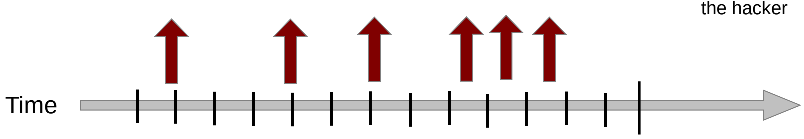
Methodologies and Processes

We are using an iterative process!

Define a plan with 1..N iterations. We do not have to care about plans...

Now, let's hack!

Harry
the hacker

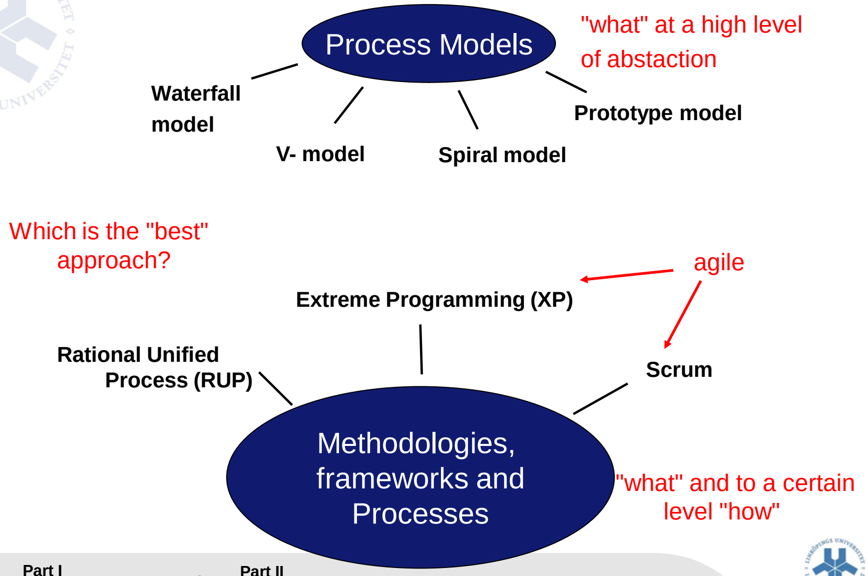


Is this a good iterative process?

Of course not. We need some structure!

Methodologies and defined Processes

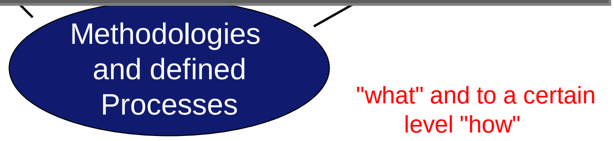
Processes, Models, Methodologies...



Processes, Models, Methodologies...

Question:
What is the difference between a methodologist and a terrorist?

Answer:
You can negotiate with a terrorist.



Goals with a software development process

- Guidance about order and content of team activities.
- Specify when and which artifact that should be produced.
- Direct individual developers' tasks and the team as a whole
- Give criteria for monitoring and measuring activities and generated products.

Agile Approaches - Agile Alliance

Lightweight approaches to satisfy the customers with "early and continuous delivery of valuable software"

Manifesto for Agile Software development

Favor

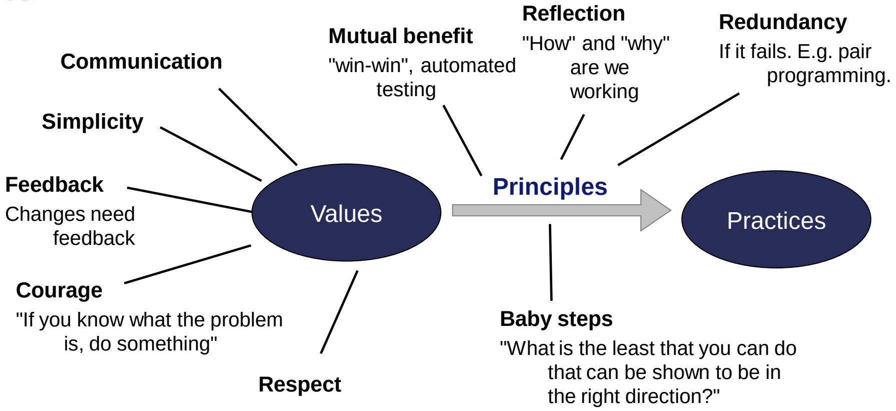
- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

(http://agilemanifesto.org, 2001)

Extreme Programming (XP)

Extreme Programming - Values and Principles

A lightweight methodology for vague or rapidly changing requirements



Communication

Simplicity

Feedback
Changes need feedback

Courage
"If you know what the problem is, do something"

Respect

Values

Mutual benefit
"win-win", automated testing

Reflection
"How" and "why" are we working

Redundancy
If it fails. E.g. pair programming.

Principles

Baby steps
"What is the least that you can do that can be shown to be in the right direction?"

Practices

Extreme Programming - Some Practices

Pair Programming

- Focus on task
- Clarify ideas
- Rotate frequently

Stories

- "requirements", but not mandatory
- Name + short story
- On index cards (paper)

Refactoring

- Behavior preserving transformation
- Tool support, e.g. Eclipse

Continuous Integration

- Integrate and test often
- Automated build system
- Automated regression tests (e.g. JUnit)

Test-First Programming

- Create tests before code
- Focus on interface and "what is needed"
- Gets tests for free

Scrum

Scrum Overview

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

Sprint, Task board, Burn-down chart, Done, Velocity

The Sprint (1)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- An iteration
- Time-boxed
- 30 days or less
- No time between sprints

- 40 hours week
- Open and visible

Sprint, Task board, Burn-down chart, Done, Velocity

The Sprint (1)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- An iteration
- Time-boxed
- 30 days or less
- No time between sprints

- 40 hours week
- Open and visible

Sprint, Task board, Burn-down chart, Done, Velocity

The Team

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Cross functional
- No titles
- Self-organized
- 7 (5) plus minus two
- Develops, tests, documents etc. in intervals - sprints

Sprint, Task board, Burn-down chart, Done, Velocity

Product Owner

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- One and only one person
- Prioritize and manage the product backlog
- Manage ROI
- The customer ”interface”

The product owner may **not**
- act as a project manager
- tell when and what something should be done

Sprint, Task board, Burn-down chart, Done, Velocity

Scrum Master

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Make sure the scrum team adheres Scrum values, practices and rules
- Run meetings
- Protects the team from disturbance
- Collects and removes obstacles (Impediment list)

The scrum master may **not**
- Mange the scrum team - the scrum team is self-organized

Scrum master cannot be product owner

Sprint, Task board, Burn-down chart, Done, Velocity

Pigs and Chickens

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Scrum team members are ”pigs”
- Everyone else is a ”chicken”
- Chickens cannot tell ”pigs” how to do their work

”A chicken and a pig are together when the chicken says ”Let’s start a restaurant!”. The pig thinks it over and says ”What would we call this restaurant?” The chicken says ”Ham n’ Eggs!” The pig says ”No thanks, I’d be committed, but you’d only be involved!”

Sprint, Task board, Burn-down chart, Done, Velocity

Product Backlog (2)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- List of product backlog items (PBI)
  (approx. List of potential requirements)
- Prioritized
- Available
- Never complete
- Features, bug fixes, documentation, tests etc.
- Value (PO) and estimates (Team)

Sprint, Task board, Burn-down chart, Done, Velocity

Release planning meeting (3)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Create the product backlog
- Initial meeting – break down product into deliverables
- Small version, end of each sprint

Sprint, Task board, Burn-down chart, Done, Velocity

Sprint planning meeting

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

Part 1 – ”What” (4)
- Break down top items
- Estimate product backlog
- Select PBIs for a sprint
- Time-boxed 4h

Part 2 - ”How” (5)
- Design
- Identify tasks (less than 1-2 days)
- Estimate tasks
- Output: Sprint backlog

Sprint, Task board, Burn-down chart, Done, Velocity

Sprint Backlog (6)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Consist of tasks
- Only track hours remaining, not hours worked
- Not ordered

Tools
- Task board (PBI, todo, In process, To verify, done)
- Burn-down chart (velocity)

Sprint, Task board, Burn-down chart, Done, Velocity

Done

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- When are we done?

- Possible to ship after each sprint
- Everybody – understand what done means

Tools to support done
- Version handling (SCM)
- Automated build
- Automated tests (Continuous integration)

Sprint, Task board, Burn-down chart, Done, Velocity

Daily Scrum

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Stand-up meeting
- Every morning
- Time-boxed 15min
- 1 minute each person

- What did you do yesterday?
- What will you do today?
- What obstacles are in your way?

Sprint, Task board, Burn-down chart, Done, Velocity

Impediment List

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- List of obstacles
- Scrum Master’s backlog
- Daily update
- Open, visible and honest

Sprint, Task board, Burn-down chart, Done, Velocity

Sprint review (8)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- Time-boxed 4h
- End of sprint
- Informal meeting – what has been done
- Demonstrate – no power points

Sprint, Task board, Burn-down chart, Done, Velocity

Sprint retrospective (8)

Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

- 3h, time-boxed
- Inspect the last sprint, regarding
  - People
  - Relationships
  - Processes
  - Tools
- How to make things better
  – process improvements

Sprint, Task board, Burn-down chart, Done, Velocity

SCRUM

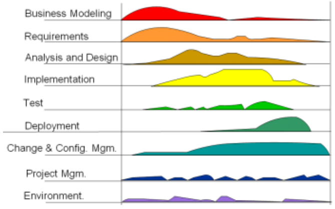
Roles
- Team
- Product Owner
- Scrum master

Lists
- Product backlog
- Sprint backlog
- Impediment list

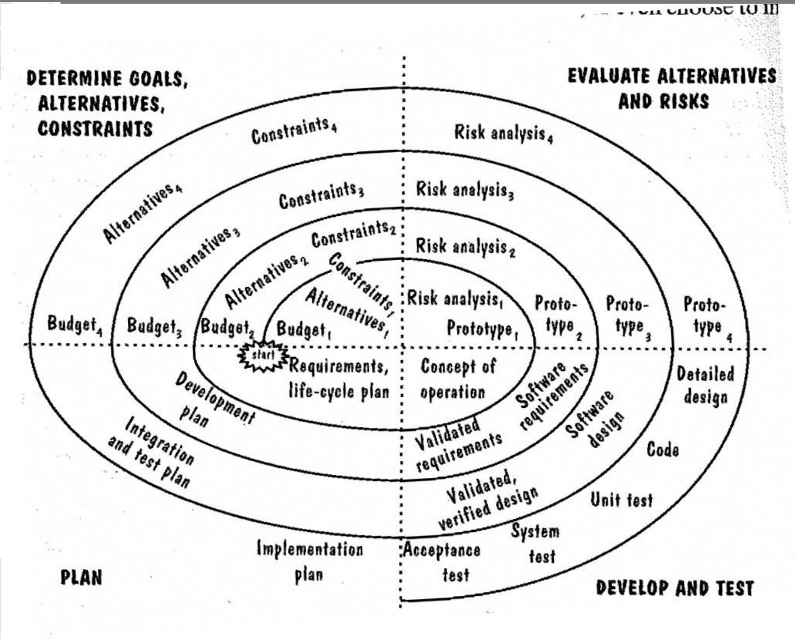
Meetings
- Release planning
- Sprint planning
- Daily Scrum
- Sprint review
- Sprint retrospective

Sprint, Task board, Burn-down chart, Done, Velocity

Rational Unified Process (RUP) and OpenUP

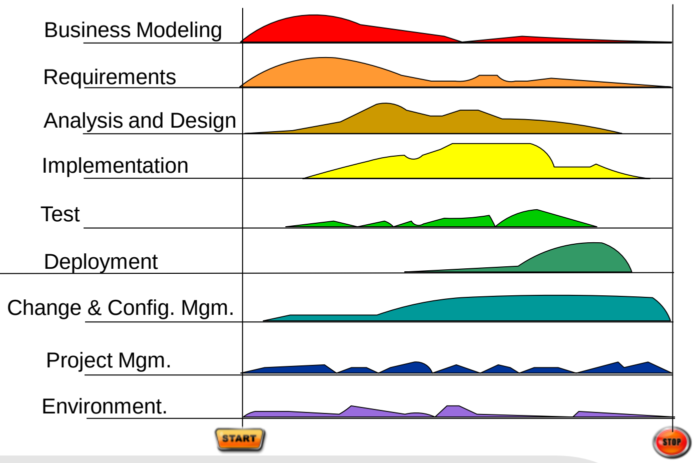


Forerunner – the spiral model



Disciplines

Core Technical Disciplines



Core Supporting Disciplines

Part I
Life Cycles and
Process Models

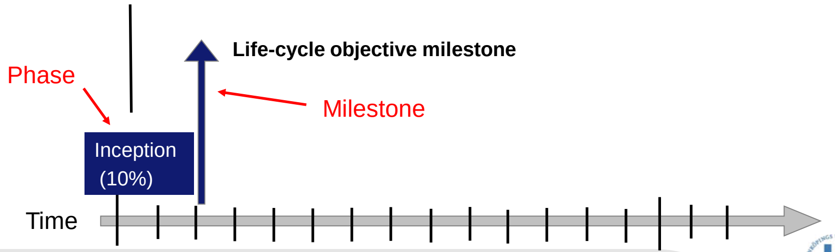
Part II
Methodologies and Processes

RUP- Phases and Milestones

Inception

- Formulate scope
- Capture most important requirements
- Plan, risk, staffing, project plan
- Synthesize a candidate architecture
- The project may be cancelled after this phase

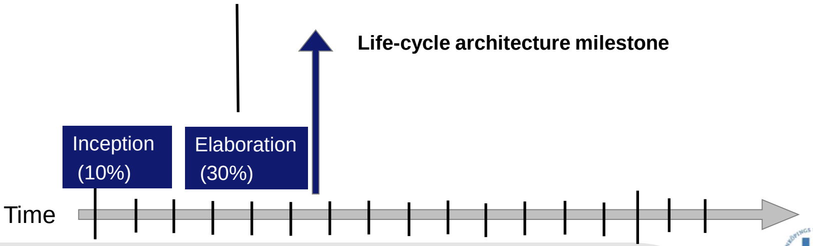
similar to a "Pre-study"



RUP- Phases and Milestones

Elaboration

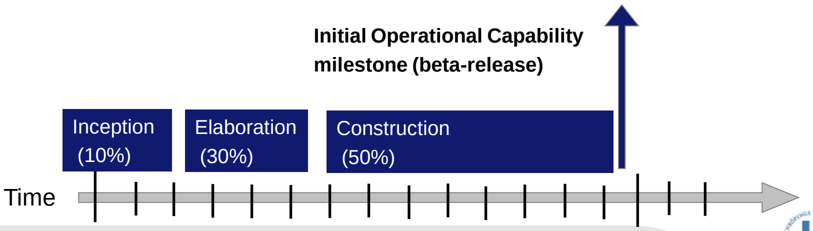
- Define architecture
- Specify requirements more precisely
- Executable architecture prototype
- Define project plan



RUP- Phases and Milestones

Construction

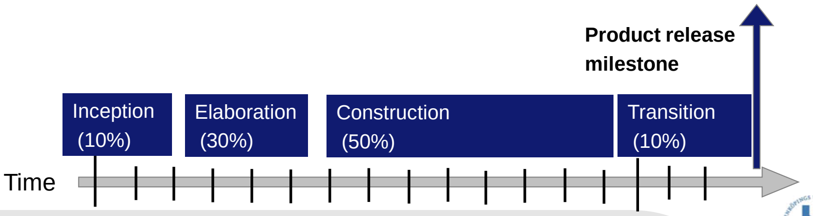
- Resource management and control
- Design, Implementation, and Testing
- Output (software + documentation) ready for users.

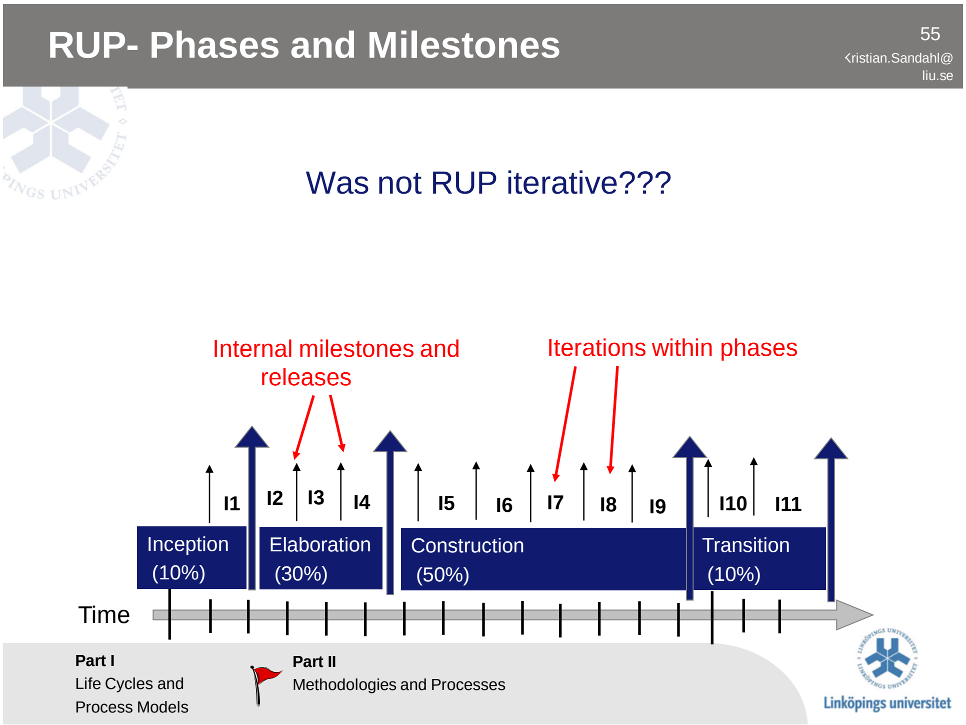


RUP- Phases and Milestones

Transition

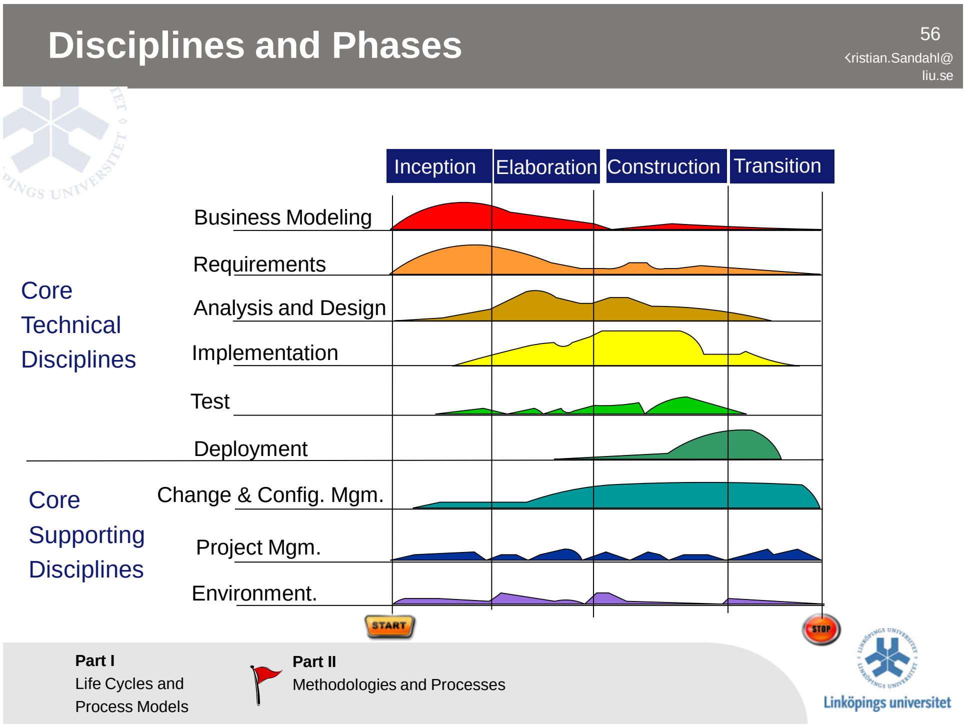
- Transition of the product to users
- Beta-testing
- Training of users and maintainers
- Rollout of the product to operational environment





RUP- Phases and Milestones

Was not RUP iterative???



Disciplines and Phases

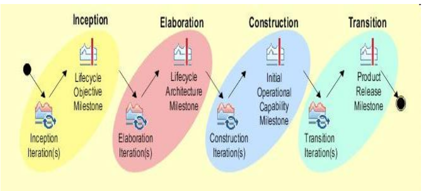
OpenUP vs. RUP

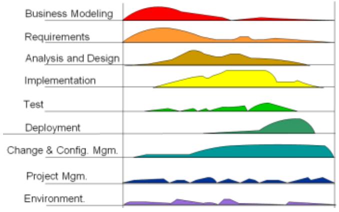
Differences to RUP

- Minimal – smaller than RUP
- Free and available
- Do not include some diciplinces, e.g. Configuration management

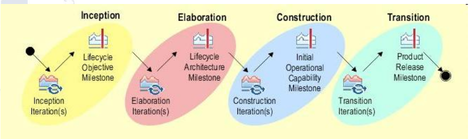
Similarities to RUP

- The 4 faces (inception, elaboration, construction, transition)
- Several defined artifacts: Arcitecture, project plan, requirements etc.


Inception
Elaboration
Construction
Transition
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release Milestone
Inception Iteration(s)
Elaboration Iteration(s)
Construction Iteration(s)
Transition Iteration(s)



OpenUP vs. Scrum


Inception
Elaboration
Construction
Transition
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release Milestone
Inception Iteration(s)
Elaboration Iteration(s)
Construction Iteration(s)
Transition Iteration(s)

Differences to Scrum

- Use cases used to elicit requirements
- Stabilized artifacts, e.g. Architecture in construction phase
- Defined milestones after phases
- Include practices (e.g. test driven development (TDD), continuous integration etc.)
- More roles: Analyst, architect, developer, project manager, stageholder, tester, any role

Similarities to Scrum

- Self-organized teams
- Time-boxed iterations
- Daily stand-up meetings
- Work Item list (similar to PB)
- Testing within iterations