Aalto University
School of Science

Software Processes

CSE-C3610, Software Engineering, 5 cr

Prof. Casper Lassenius

Software Process

What?
Why?

Software Process Definitions

- Process
  - Webster:

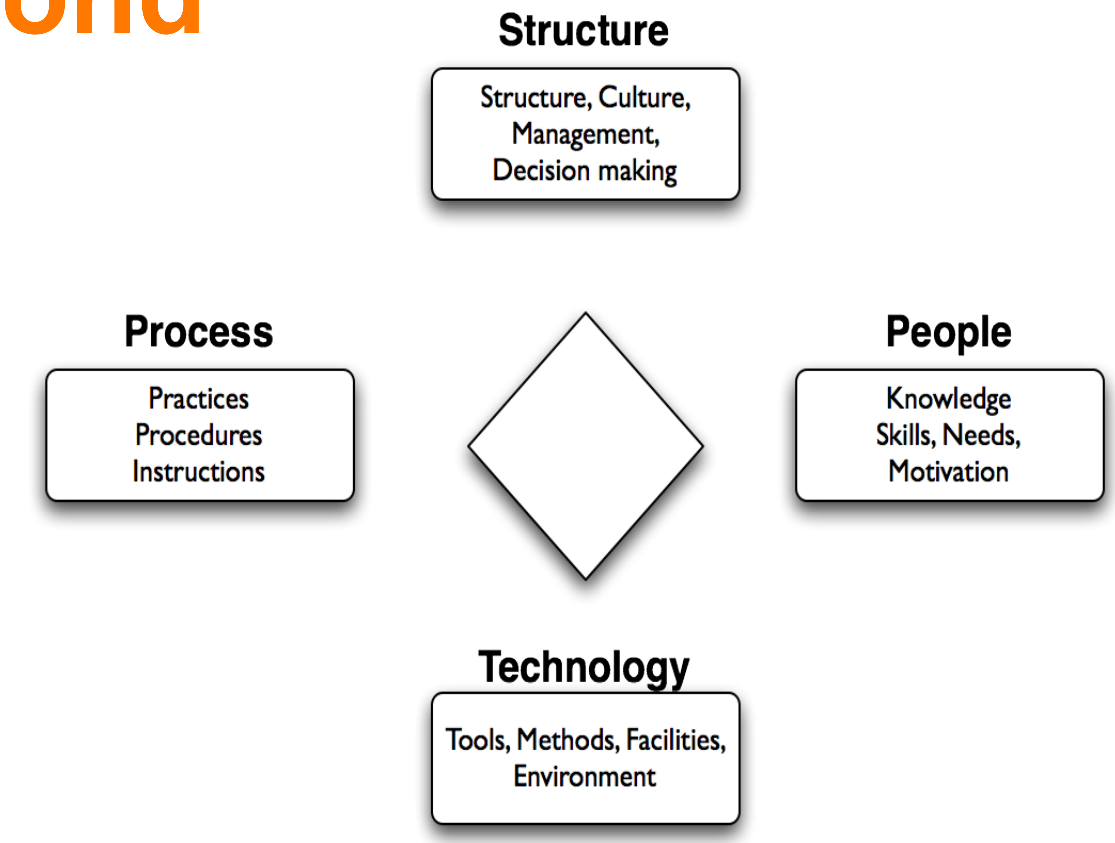
    1. A continuing development involving many changes.

    2. A particular method for doing something, usually involving a number of steps or operations.

  - IEEE: A sequence of steps performed for a given purpose.

- Software Process
  - CMM(I): a set of activities, methods, practices and transformations that people use to develop and maintain software and the associated products
  - Simply: the way an organization/team/individual develops software

Leavitt’s Organizational Diamond



Adapted from Leavitt, H.J. Applied organizational change in industry: Structural, technological and humanistic approaches. Handbook of Organizational. J.G. March. Chicago, Rand McNally. 1965

Software Process (Life-Cycle) Models

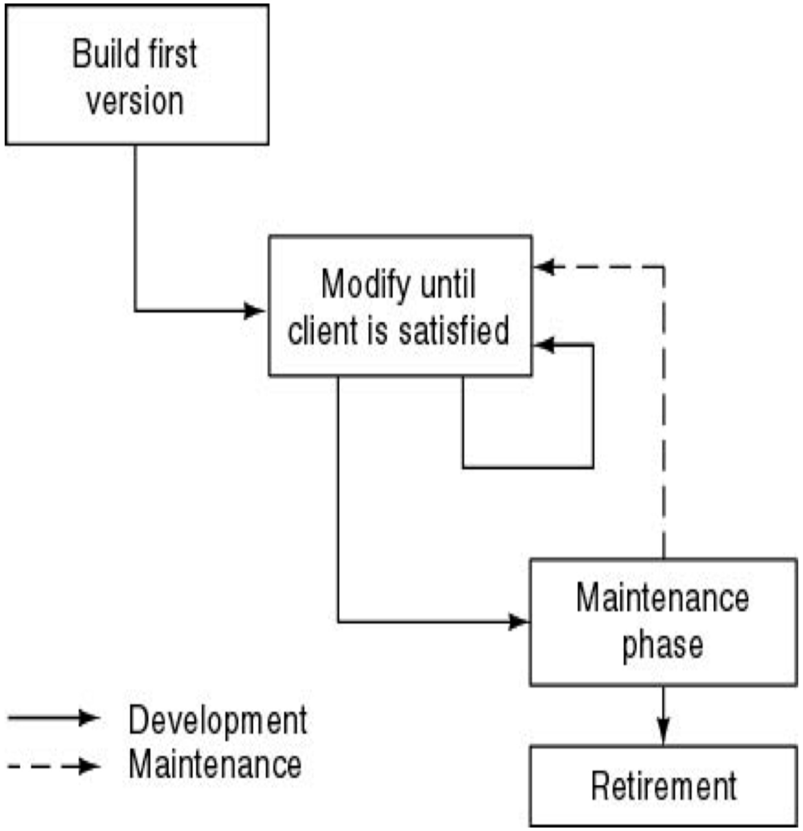
- Build-and-Fix
- Waterfall
- Rapid Prototyping

Software Process Models

- Order all or some of the basic software development activities in various ways

- Typical activities in a Life-Cycle model (LCM)
  - Requirements / specification
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance
  - Retirement

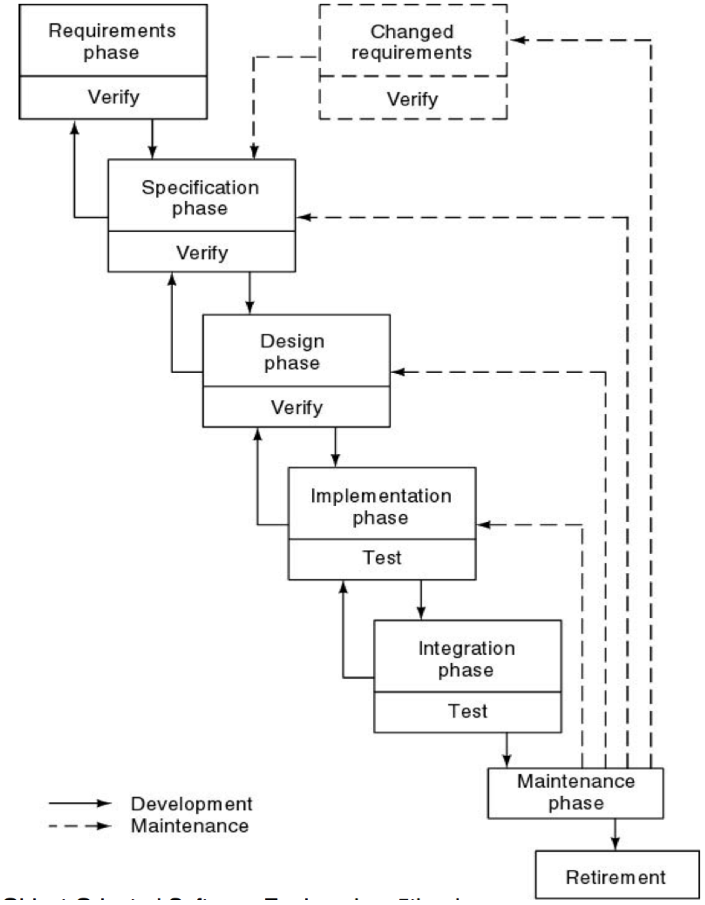
Build and Fix Model



- Problems
  - No specifications
  - No design
  - Lack of visibility
  - Easily leads to poorly structured systems
  - Totally unsatisfactory
  - Need life-cycle model

Picture from Schach: Classical and Object-Oriented Software Engineering, 5th ed.

Waterfall Model



- Planning and control
- Documentation-driven
- “Doing the homework”
- Formal change management

Picture from Schach: Classical and Object-Oriented Software Engineering, 5th ed.

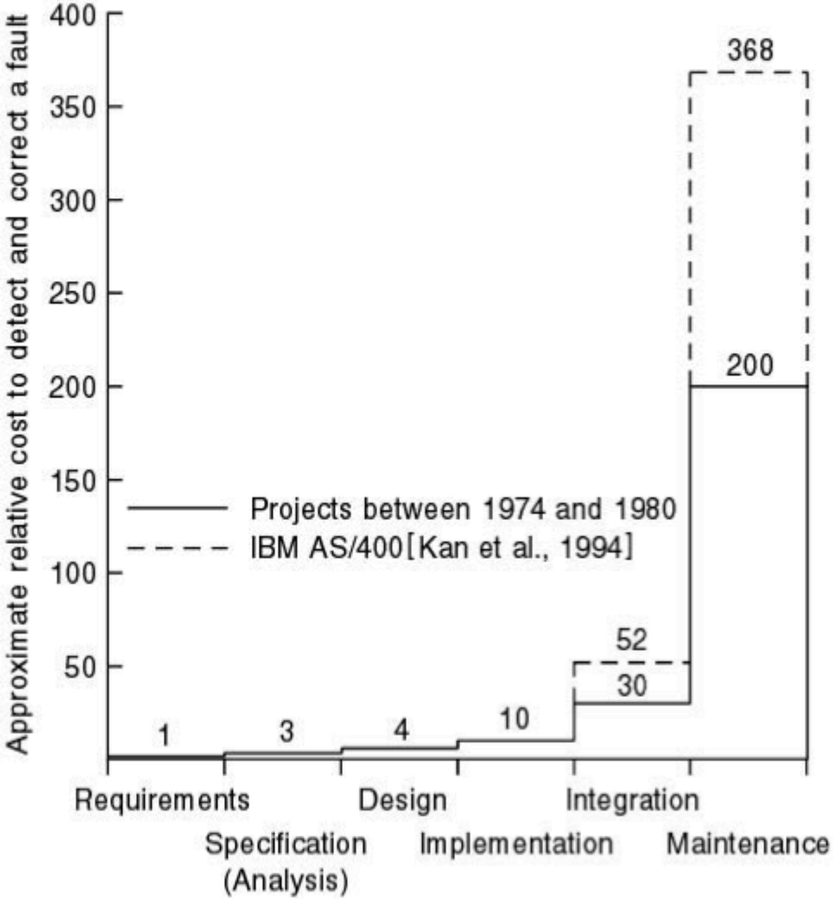
The Waterfall Model

- **Strengths**
  - Easily manageable process
  - Probably the most effective model, if you know the requirements
  - Extensive documentation

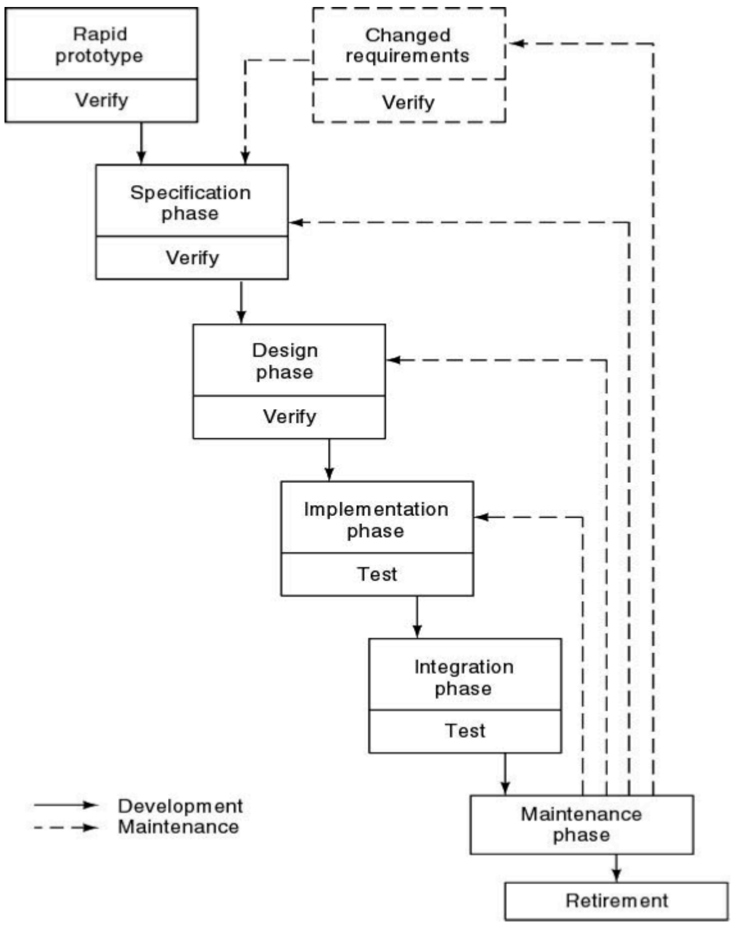
- **Weaknesses**
  - Inflexible partitioning of the project into distinct phases
  - Difficult to respond to changing customer requirements
  - Feedback on system performance available very late and changes can be very expensive

- **Applicability**
  - Appropriate when the requirements are well understood
  - Short, clearly definable projects (e.g. maintenance)
  - Very large, complex system development that requires extensive documentation. Safety critical systems.

Cost to Detect and Correct a Fault



Rapid Prototyping Model



- Linear
- “Rapid”
- Exploratory vs. throw-away prototypes

Aalto University
School of Science

Picture from Schach: Classical and Object-Oriented Software Engineering, 5th ed.

Software Process (Life-Cycle) Models

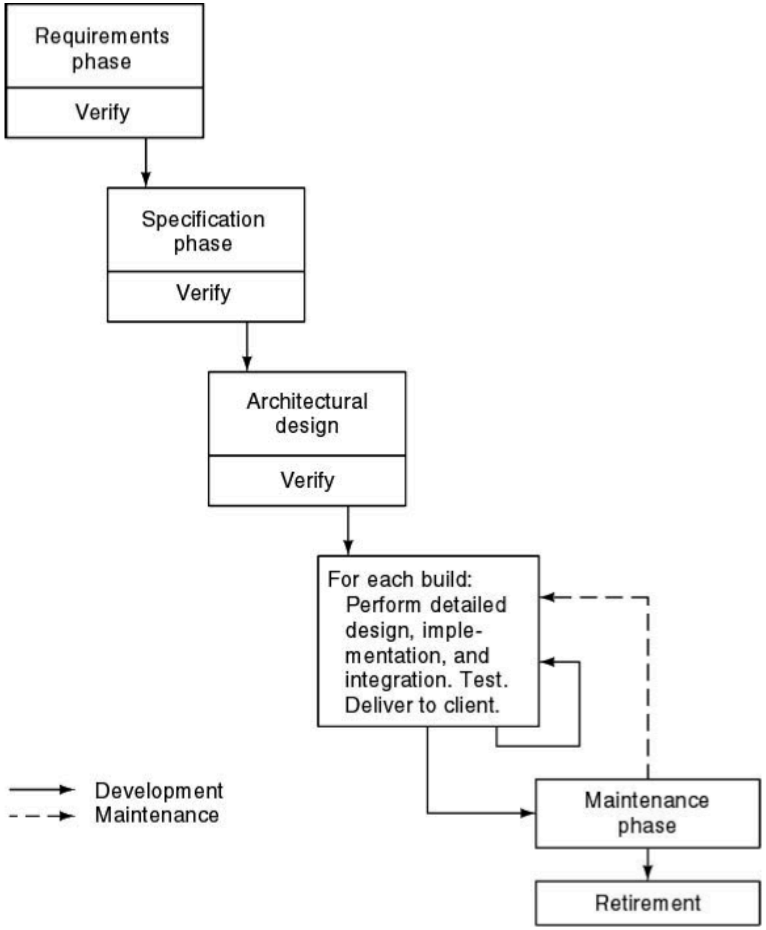
- Incremental development
- Rational Unified Process (RUP)
- Microsoft Sync-and-Stabilize

Incremental Model

- The concept of growing a system via iterations: iterative and incremental development (IID)
  - Divide the project into increments
  - Each increment adds functionality
  - Each iteration is a self-contained mini project composed of activities such as requirements analysis, design, programming and test
  - At the end of the iteration an iteration release: a stable, integrated and tested partially complete system
  - Most releases internal, final iteration release is the complete product

- Prioritize user requirements
  - MOSCOW priorities: must, should, could, want
  - High-priority requirements into early increments
  - Freeze requirements during each increment

Incremental Model



Picture from Schach: Classical and Object-Oriented Software Engineering, 5th ed.

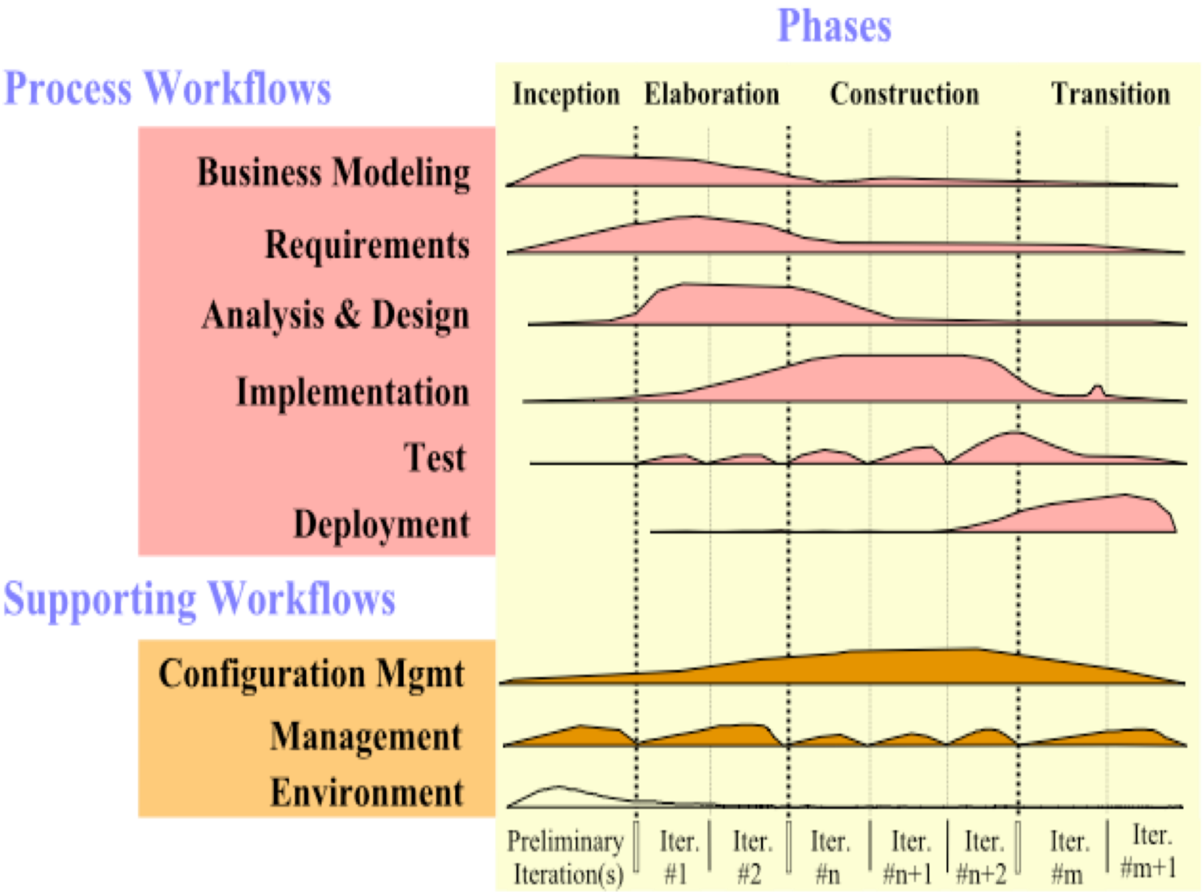
Incremental Development Advantages

- Customer value can be delivered at the end of each increment making system functionality available earlier
- Final product better matches true customer needs
- Early increments act as a prototype to help
  - elicit requirements for later increments
  - get feedback on system performance
- Lower risk of overall project failure
- Smaller sub-projects are easier to control and manage
  - A meaningful progress indicator: tested software
- The highest priority features tend to receive the most testing
- Job satisfaction is increased for developers who can see early results of their work

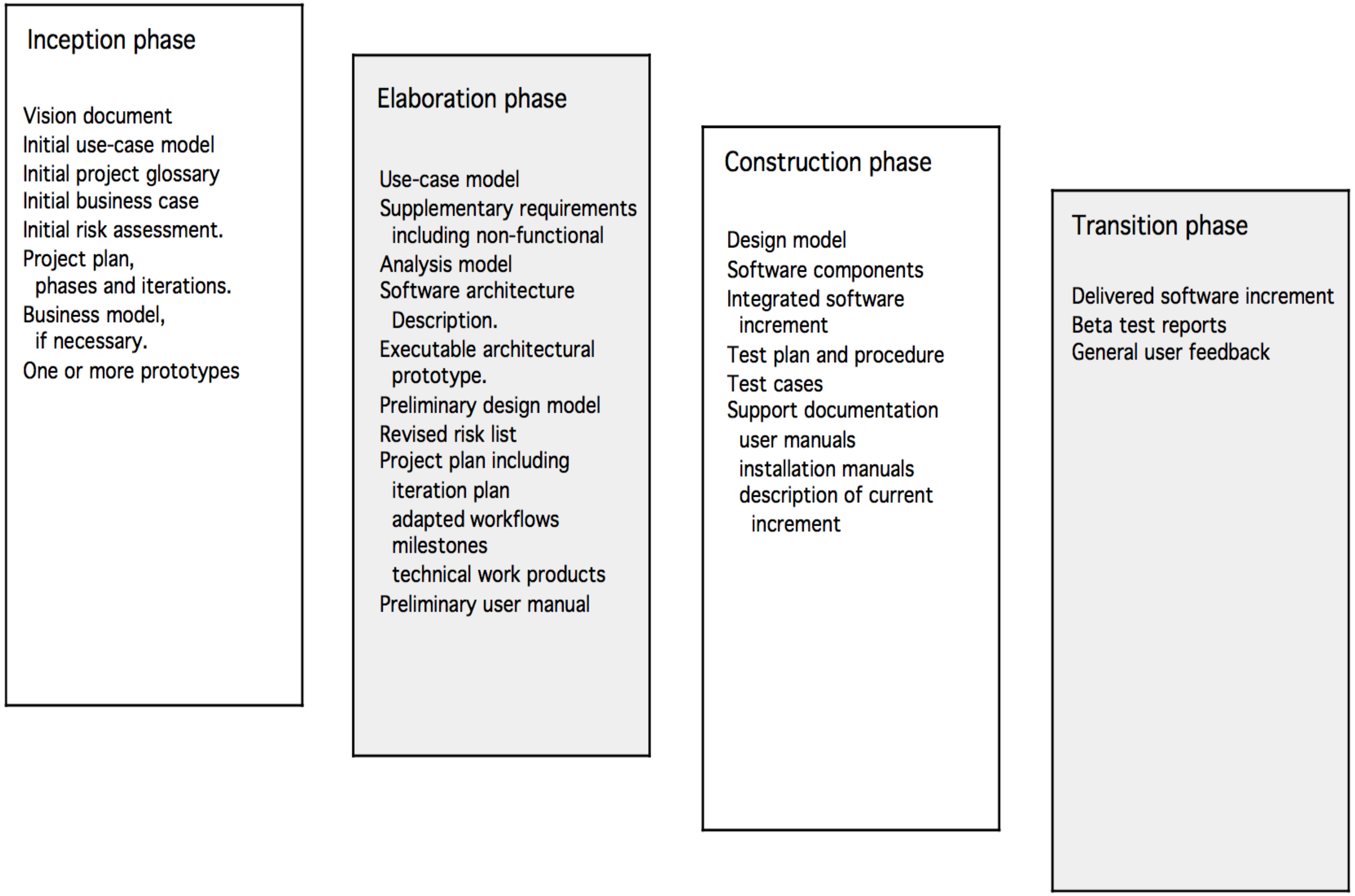
Incremental Development Disadvantages

- Can be harder to plan and control than waterfall development
- Can be more expensive than waterfall development
- May require more experienced staff
- System architecture must be adaptive to change
- Software project contracts are still mostly drawn up according to the waterfall model and all changes cause renegotiations

Rational Unified Process (RUP)



UP Work Products



Inception phase

- Vision document
- Initial use-case model
- Initial project glossary
- Initial business case
- Initial risk assessment.
- Project plan, phases and iterations.
- Business model, if necessary.
- One or more prototypes

Elaboration phase

- Use-case model
- Supplementary requirements including non-functional
- Analysis model
- Software architecture Description.
- Executable architectural prototype.
- Preliminary design model
- Revised risk list
- Project plan including
  - iteration plan
  - adapted workflows
  - milestones
  - technical work products
- Preliminary user manual

Construction phase

- Design model
- Software components
- Integrated software increment
- Test plan and procedure
- Test cases
- Support documentation
  - user manuals
  - installation manuals
  - description of current increment

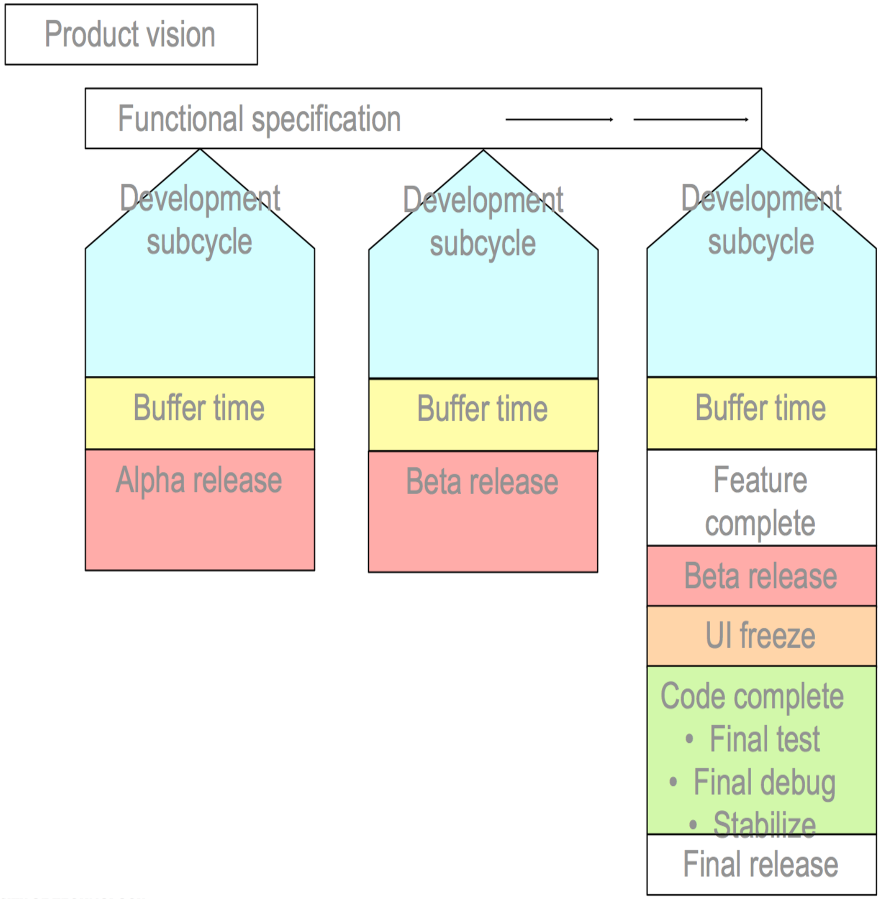
Transition phase

- Delivered software increment
- Beta test reports
- General user feedback

Aalto University
School of Science

Synchronize-and-Stabilize Model

- Microsoft' s life-cycle model
- Requirements analysis—interviev potential customers
- Draw up specifications
- Divide project into 3 or 4 builds
- Each build is carried out by small teams working in parallel



Product vision

Functional specification

Development subcycle | Buffer time | Alpha release

Development subcycle | Buffer time | Beta release

Development subcycle | Buffer time | Feature complete | Beta release | UI freeze | Code complete

- Final test
- Final debug
- Stabilize

Final release

Sync-and-Stabilize

- At the end of the day—synchronize (test and debug)
- At the end of the build—stabilize (freeze build)
- Components always work together
  - Get early insights into operation of product

Still Other Process Models

- Spiral model—a risk-driven meta-model
- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing and constructing aspects

Questions?