Software Processes and Life-Cycle Models

CS-C3150, Software Engineering

Casper Lassenius

Software Process

- The way an organization/team/ individual does software engineering
- Can be construed as consisting of a set of
  - **activities** - what is done,
  - **artifacts** - what is produced and consumed,
  - **agents** - who does it,
  - and their **relationships**

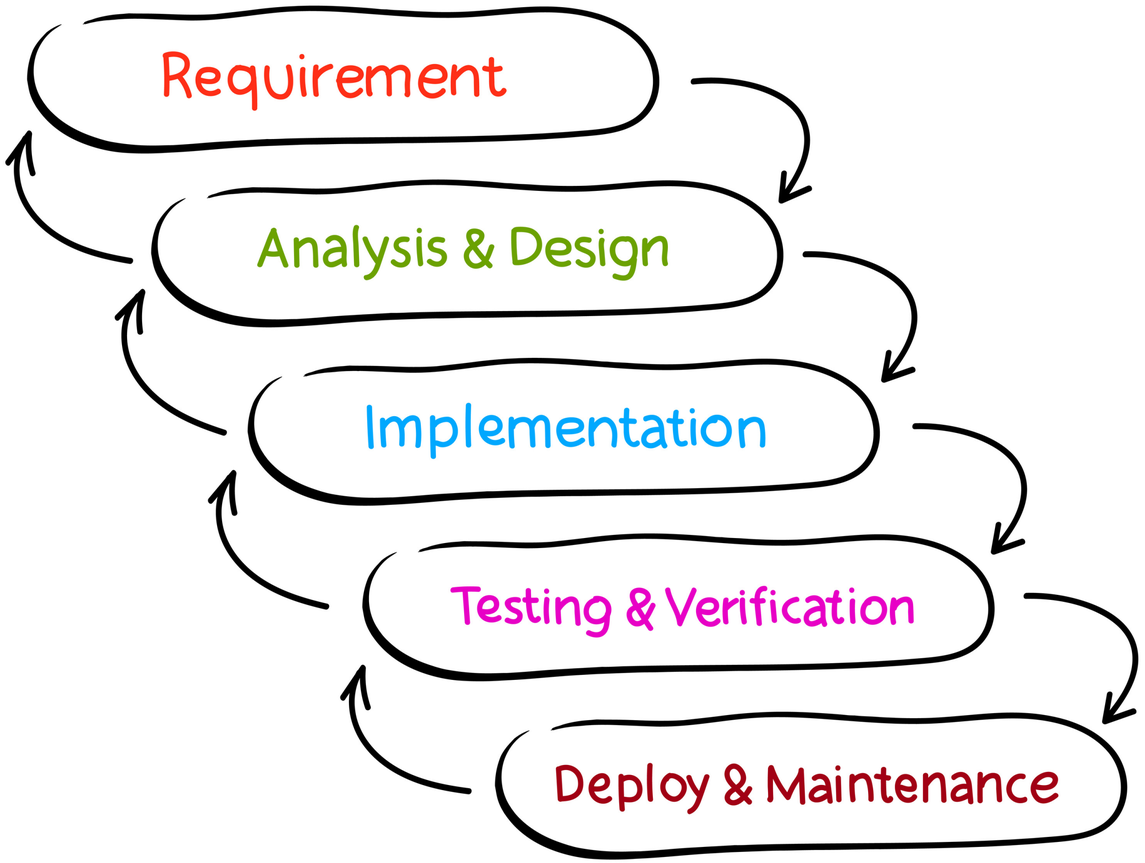
Software Life-Cycle Models

Life-Cycle Models

- Provide a high-level overview of the main activities of software engineering
- Different models order the activities in different ways
- Typical LCM activities
  - Requirements
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance
  - Retirement

The Waterfall Model

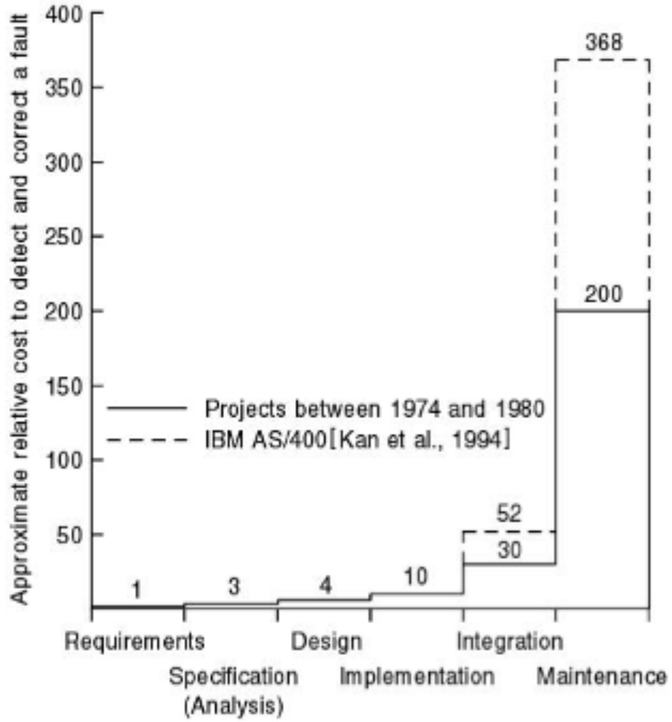
Waterfall-Model

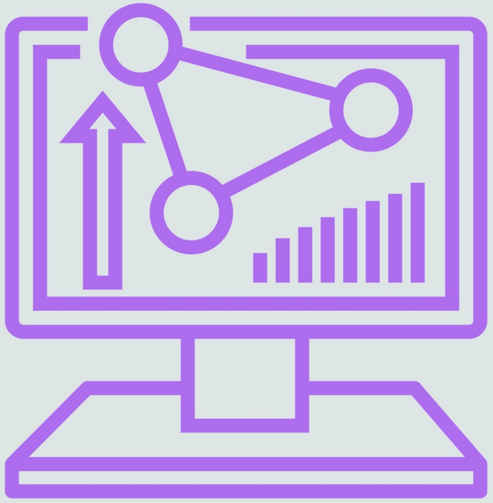


- Sequential stages
- Document-driven
- Focus on “doing the homework”
- Formal change management
- Iterations are avoided and discouraged
- Based on a misunderstanding of a classic paper by Winston Royce (in the readings)
- The arguably most influential model in the history of software engineering

Pros and Cons

- Pros
  - Easy to understand
  - Clear phases, milestones, and deliverables - “easy” to use as a basis for contracts
  - Provides extensive documentation
  - Less (total) planning than in iterative models
  - Many contractual models are build on this model
- Cons
  - Difficult to respond to changing requirements
  - Feedback on system features and performance available very late
  - Changes can be very expensive
- Applicability
  - Well-understood requirements that are unlikely to change
  - When the customer demands it :)

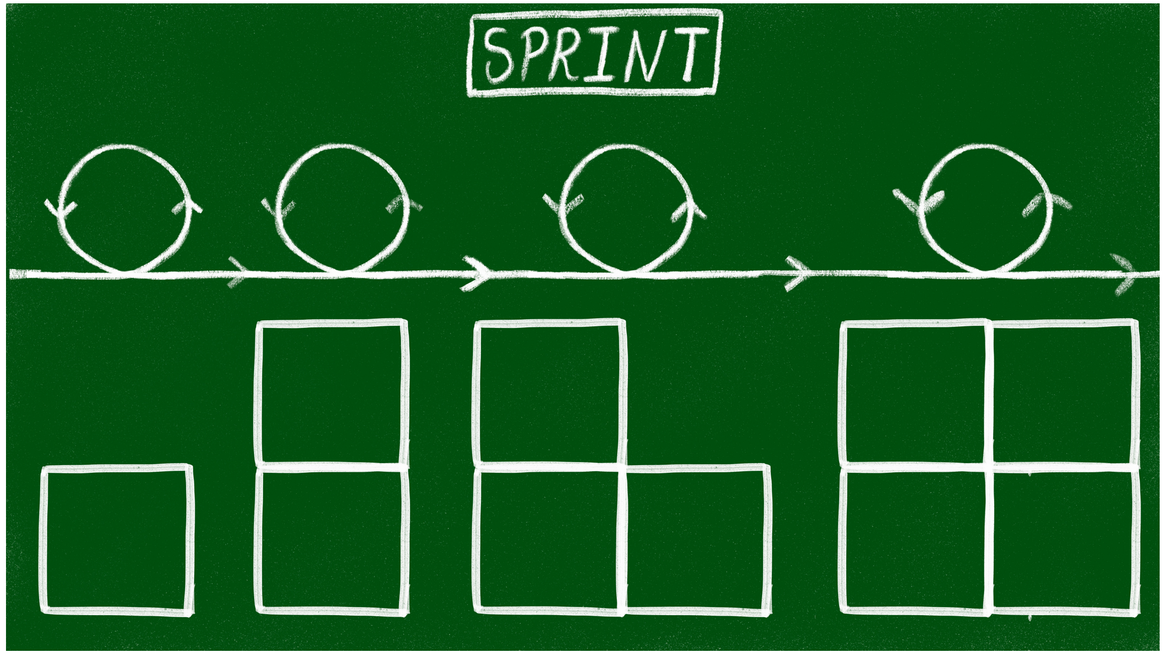




ITERATIVE AND INCREMENTAL DEVELOPMENT

Iterative and Incremental Development

- Growing a system via iterations
  - Divide the project into increments, each adding functionality
    - Each iteration is a self-contained mini-project with activities, such as requirements, design, implementation, and test
    - Each iteration releases a software version: a stable, integrated and tested partially complete system
    - The final iteration release is the final product
  - User requirements prioritization helps with allocation to iterations
    - MOSCOW: Must, Should, Could, Want
    - High-priority requirements in early increments
    - Each increment works with frozen requirements (cannot change during the iteration)



- Example: Scrum
  - Sprint = Iteration
  - Agile processes are iterative and incremental
  - Each iteration produces a part of the system

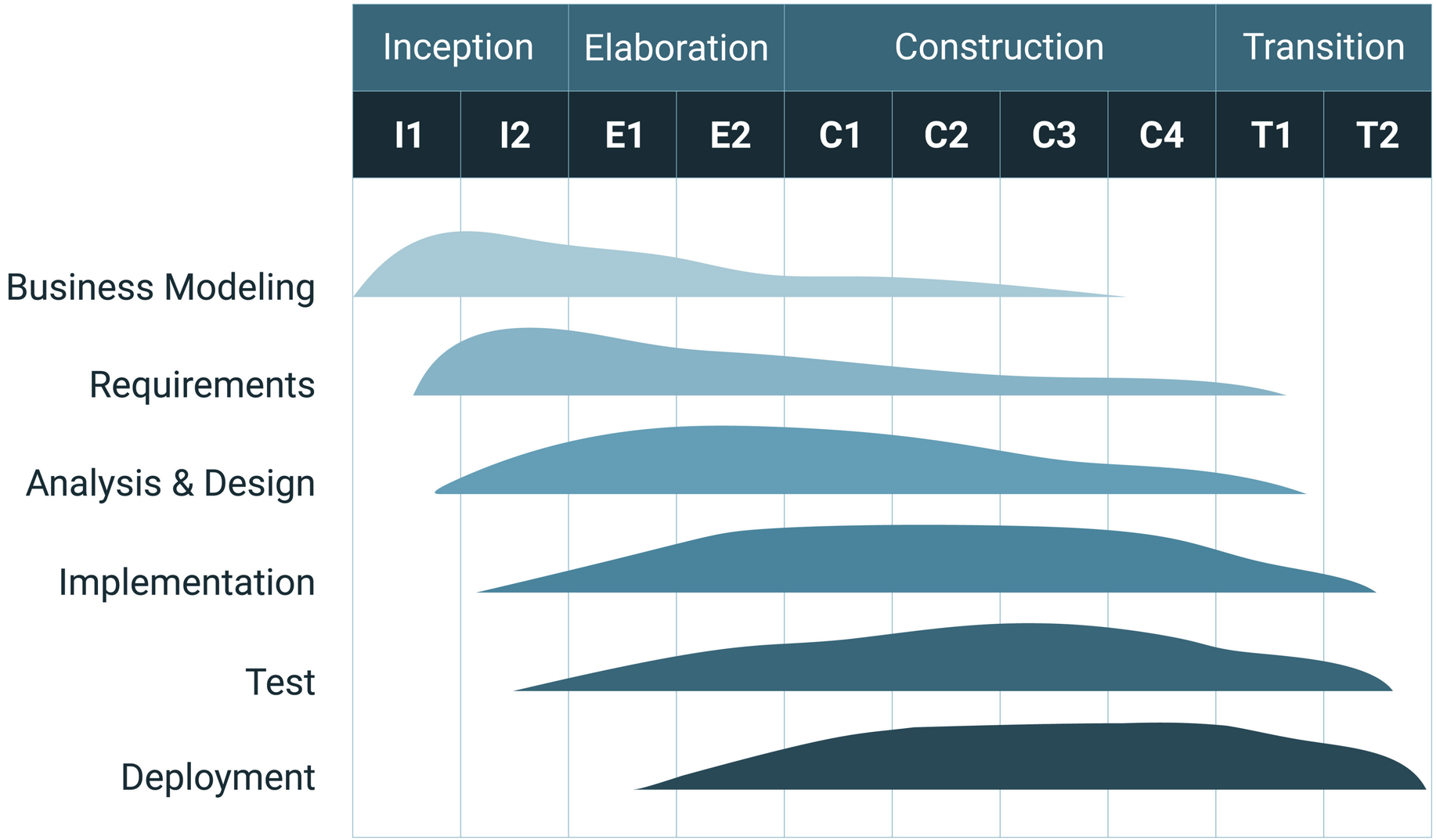
Incremental Development Advantages

- Customer value can be delivered at the end of each increment making system functionality available earlier
- The 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

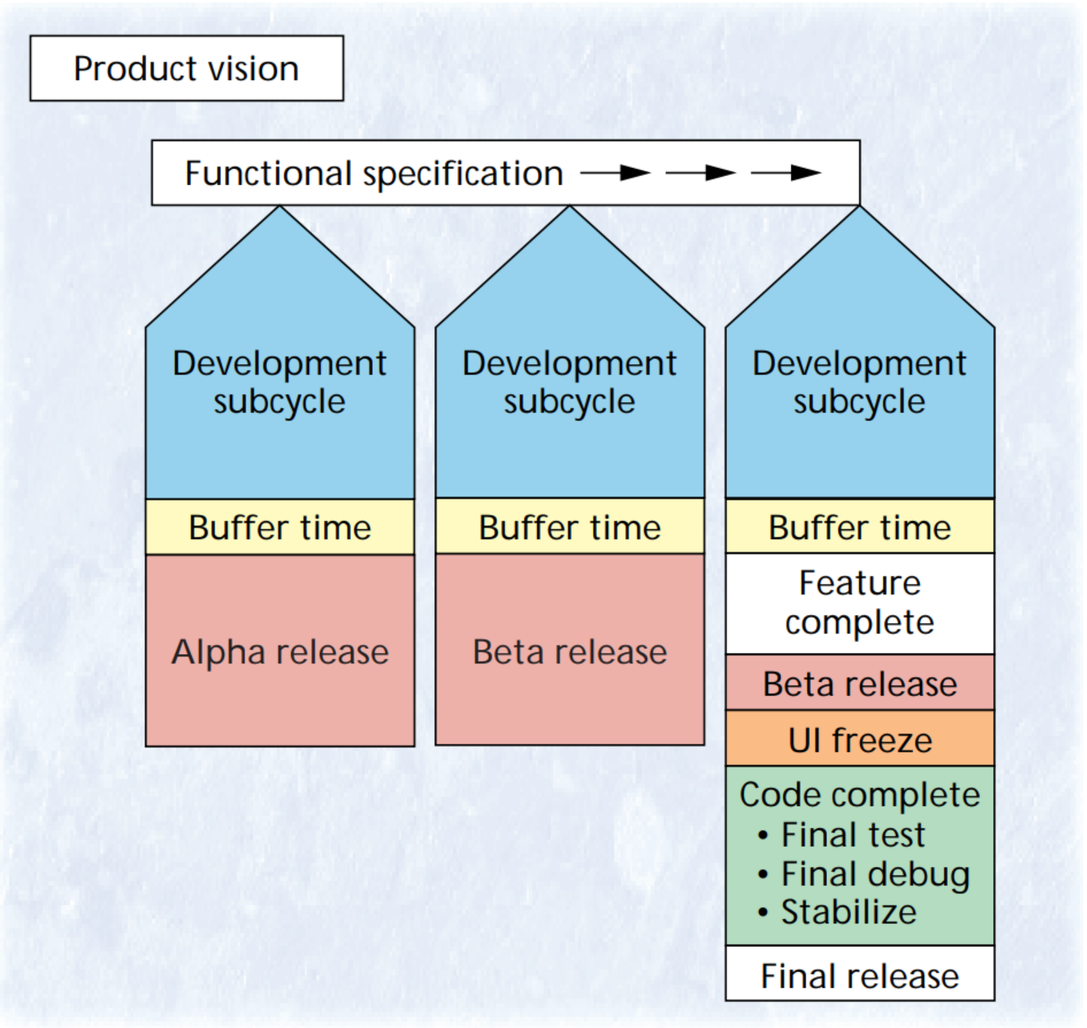
- Can be harder to plan and control than waterfall development
- Can be more expensive than waterfall development
- May require more experienced staff
- System architecture must be adaptive to change
- Contracts are still mostly drawn up according to the waterfall model, and changes might require renegotiations

RATIONAL UNIFIED PROCESS (RUP)



Microsoft Synch-and-Stabilize

- Requirements analysis—interview potential customers - create a Product Vision
- Draw up a high-level specification
- Divide the project into 3 or 4 builds
- Teams work in parallel on the builds
  - At the end of the day—synchronize (test and debug)
  - At the end of the build—stabilize (freeze build)



Others

- Spiral model
- Component-based development
- Formal methods
- Model-based software engineering
- Agile methods
- AI assisted software engineering

Thanks & Have Fun!