The Unified Process

Dror Feitelson

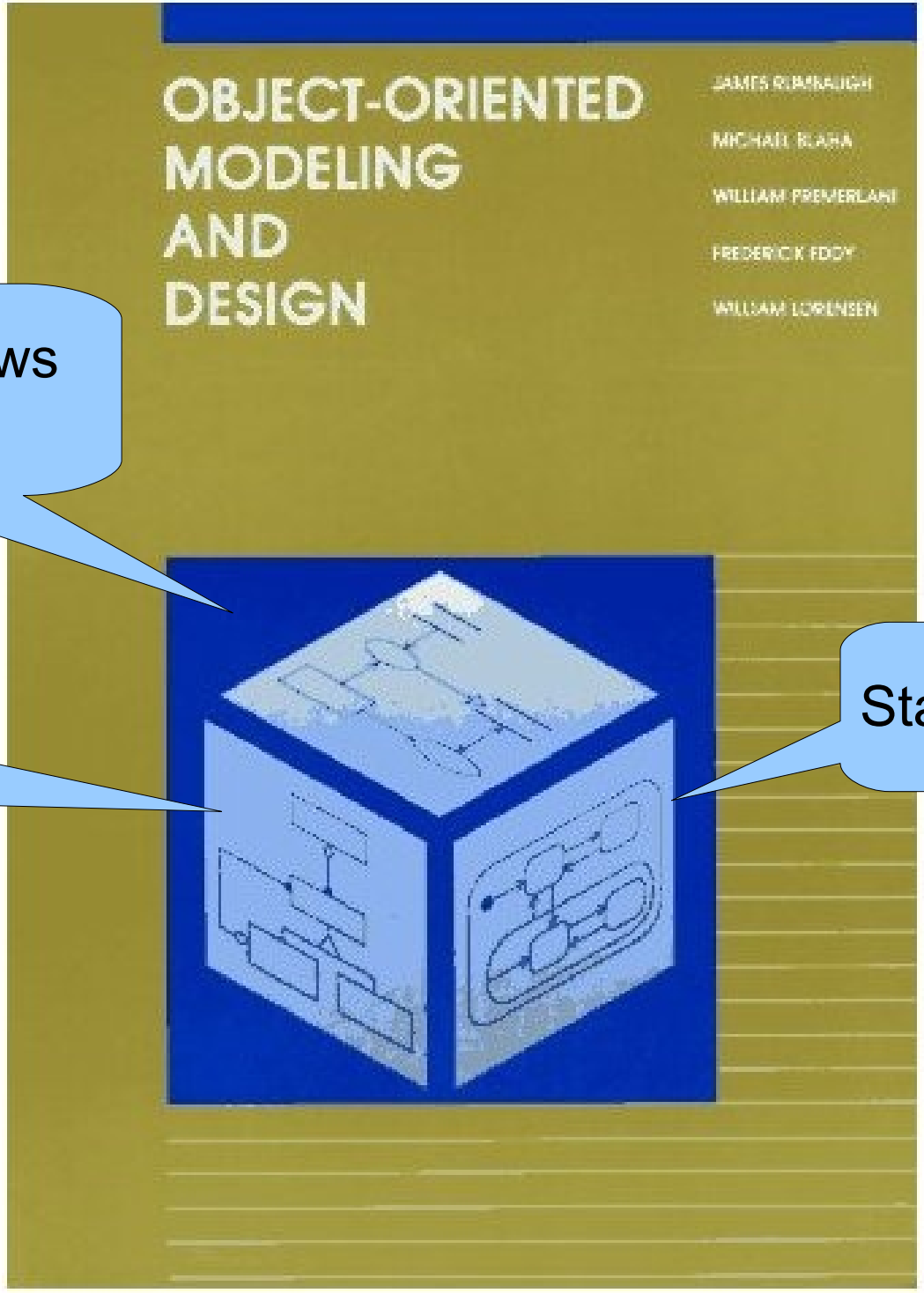
Basic Seminar on Software Engineering
Hebrew University
2009

Process

- Not really a process
  - Does not specify precisely what to do at each step
- More of a framework
- Needs to be adjusted to each project according to need
- Many refinements and extensions
  - Agile unified process
  - Enterprise unified process

History

- 1990: James Rumbaugh's OOAD
- 1992: Ivar Jacobson's Objectory
- 1993: Grady Booch's OOAD and diagrams
- 1995: Rational Software unites all three
  - Definition of UML
  - Definition of unified process
  - Rational Rose toolset
- 2003: IBM buys Rational



Use multiple views of the system

Class diagram

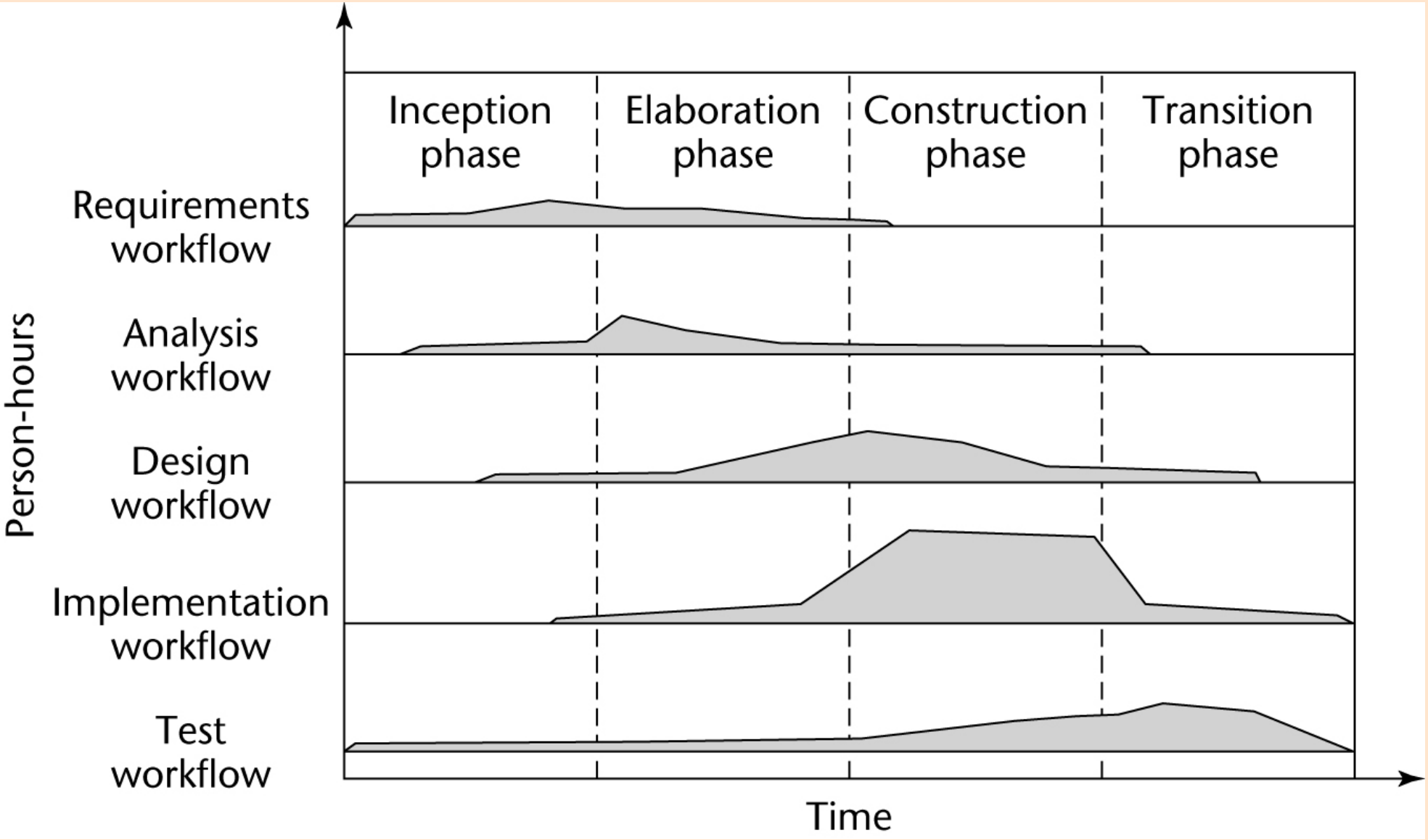
Statechart

Rambaugh et al.
Prentice-Hall
1990

Principles

- Iterative and incremental
  - Four phases divided into multiple iterations
- Use-case driven
  - Development is based on usage scenarios
- Architecture centric
  - Defining and refining the architecture is a major activity, and the baseline architecture a major milestone
- Risk focused
  - Activities in iterations prioritized to reduce risk

Phases and Workflows



Phase Milestones

- Inception: figure out what this is all about, and that it is feasible

  outcome: contract for the project
- Elaboration: figure out how to actually do it

  outcome: project architecture
- Construction: now do it

  outcome: initial running system installed
- Similar to Boehm's 3 anchor points

Continuous Workflows

- Exist in all iterations
- e.g. testing is done from the beginning
  - Even when there is no code to test
  - So test validity, completeness, and consistency of whatever artifact was produced
  - And plan relevant future code tests
- Relative weight may differ in different phases

UML

- Lots and lots of charts and repetitions
  - Use cases with detailed descriptions
  - Class diagrams with inheritance, attributes, and methods
  - Statecharts with system decomposition and state transitions
  - Sequence diagrams with interactions among components
  - Collaboration diagram with interactions superimposed on class diagram
- Serve as documentation
- But why do we need so many?

UML

- Lots and lots of charts and repetitions
  - Use cases with detailed descriptions
  - Class diagrams with inheritance, attributes, and methods
  - Statechart
    transiti
  - Seque
    compo
  - Collaboratio
    superimposed on class diag
- Serve as documentation
- But why do we need so many?

Constructing the diagrams forces the developers to really learn about the domain and the system requirements