

The University of Iowa
22C:22 (CS:2820)
**Object-Oriented Software
Development**

Fall 2013

**Iterative Evolutionary
Development**

by

Cesare Tinelli

Iterative and Evolutionary Development

Software development approach
emphasizing

early programming and

testing of a partial system

in repeating cycles

Iterative and Evolutionary Development

- Development starts before all the requirements are defined in detail
- Feedback is used to clarify and improve evolving specifications
- Relies on short quick development steps, feedback, and adaptation to clarify the requirements and design

The Unified Process

- Popular iterative process for projects using OO analysis and design
- Combines commonly accepted best practices into a cohesive and well-documented process
- For us, it is an example how to do, and so explain, OO analysis and design
- It promotes iterative and evolutionary development

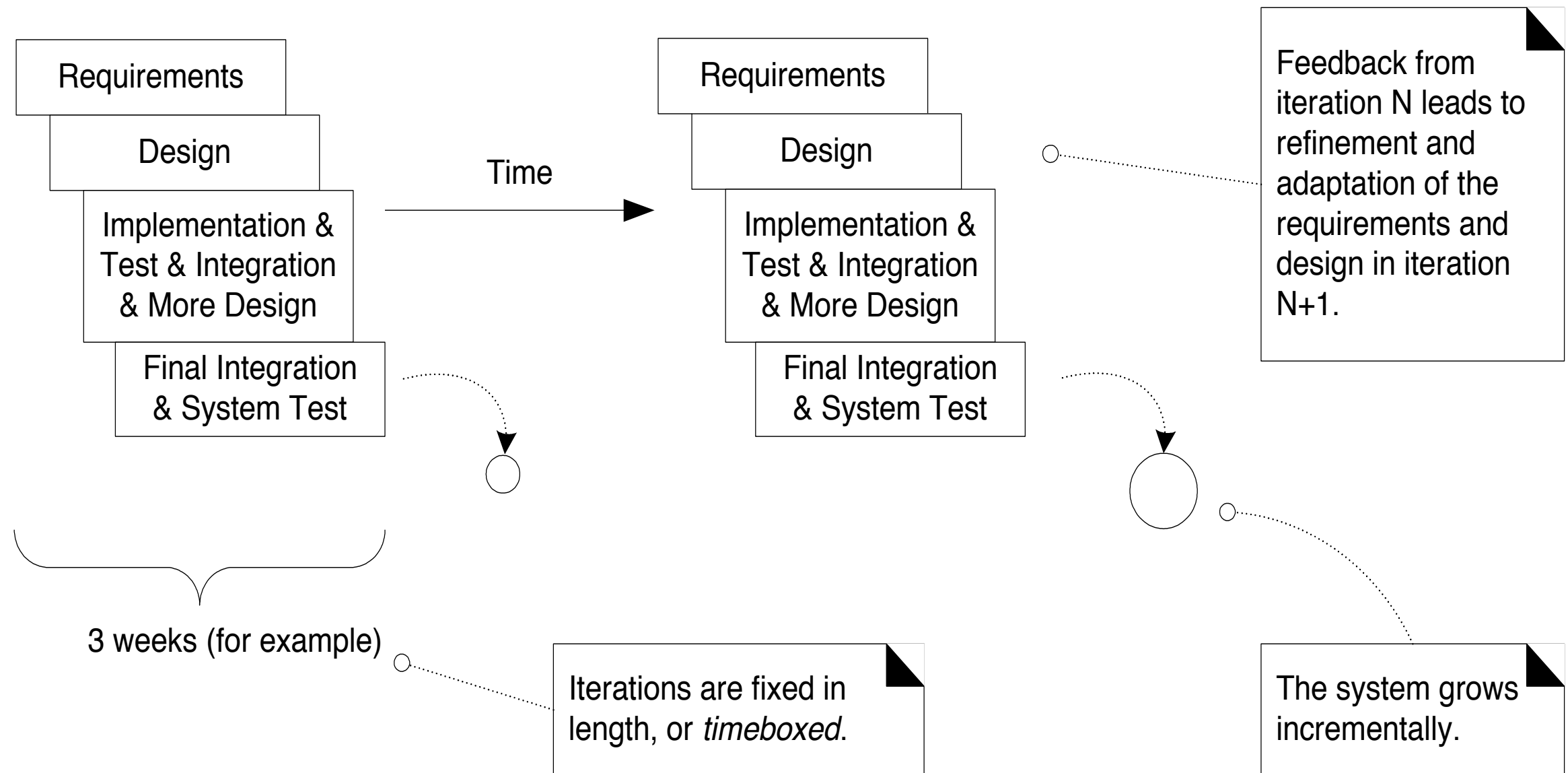
Iterative Development

- Development is organized into a series of short, fixed-length mini-projects called iterations
- The outcome of each iteration is a tested, integrated, and executable partial system
- Each iteration includes its own requirements analysis, design, implementation, and testing activities

Iterative Development

- The process lifecycle is based on the successive enlargement and refinement of a system through multiple iterations
- Cyclic feedback and adaptation are core drivers to converge upon a suitable system
- The system grows incrementally over time, iteration by iteration
- Specification and design evolve as a result of feedback and adaptation

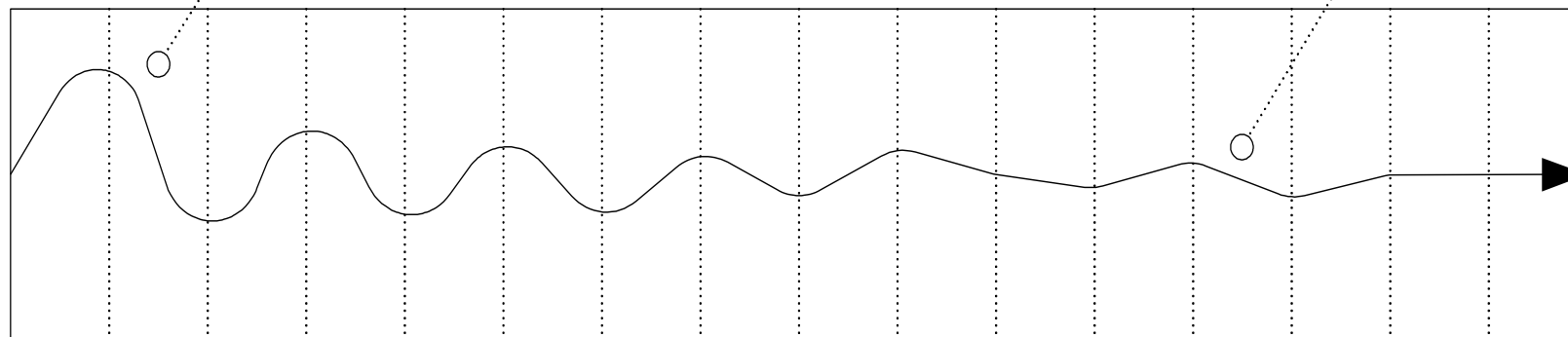
Iterative and evolutionary development



Iterative and evolutionary development

Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.

In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.



one iteration of design,
implement, integrate, and test

Benefits

- Fewer project failures, better productivity, and lower defect rates
- Early rather than late mitigation of high risks (technical, requirements, objectives, usability, ...)
- Early visible progress

Benefits

- A refined system that more closely meets the real needs of the stakeholders
- Managed complexity (the team is not overwhelmed by “analysis paralysis” or very long and complex steps)
- The learning within an iteration can be methodically used to improve the development process itself

Risk-Driven and Client-Driven Iterative Planning

- The UP encourages a combination of
 - risk-driven iterative planning and
 - client-driven iterative planning
- Early iterations aimed at
 1. identifying and reducing the highest risks
 2. building visible features the client cares most about

Other Critical UP Practices

- tackle high-risk and high-value issues in early iterations
- continuously engage users for evaluation, feedback, and requirements
- build a cohesive, core architecture in early iterations
- continuously verify quality; test early, often, and realistically

Other Critical UP Practices

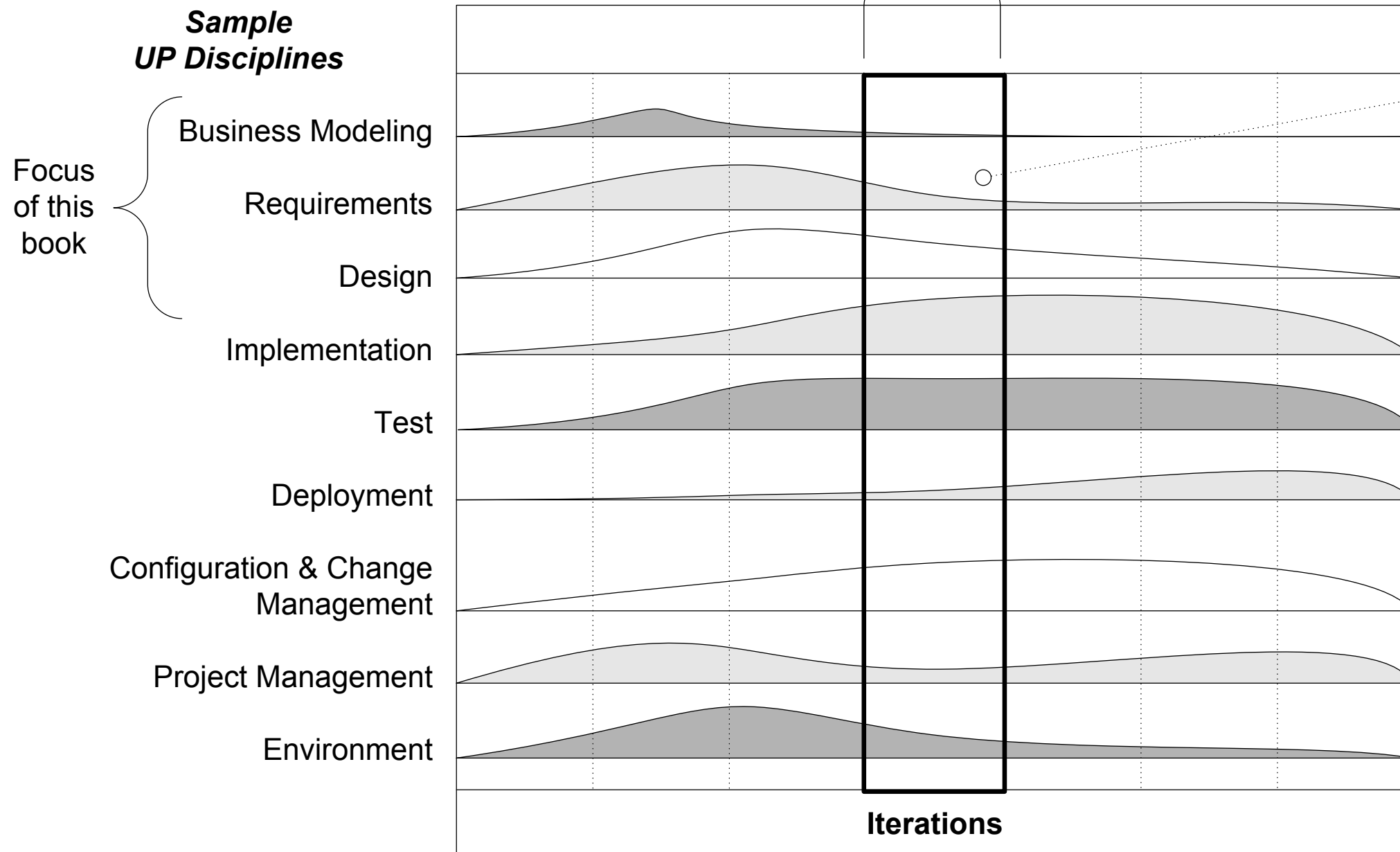
- apply use cases where appropriate
- do some visual modeling (with UML)
- carefully manage requirements
- practice change request and configuration management

UP Phases

1. **Inception**—approximate vision, business case, scope, vague estimates
2. **Elaboration**—refined vision, iterative implementation of the core architecture, resolution of high risks, identification of most requirements and scope, more realistic estimates
3. **Construction**—iterative implementation of the remaining lower risk and easier elements, and preparation for deployment
4. **Transition**—beta tests, deployment

UP Disciplines

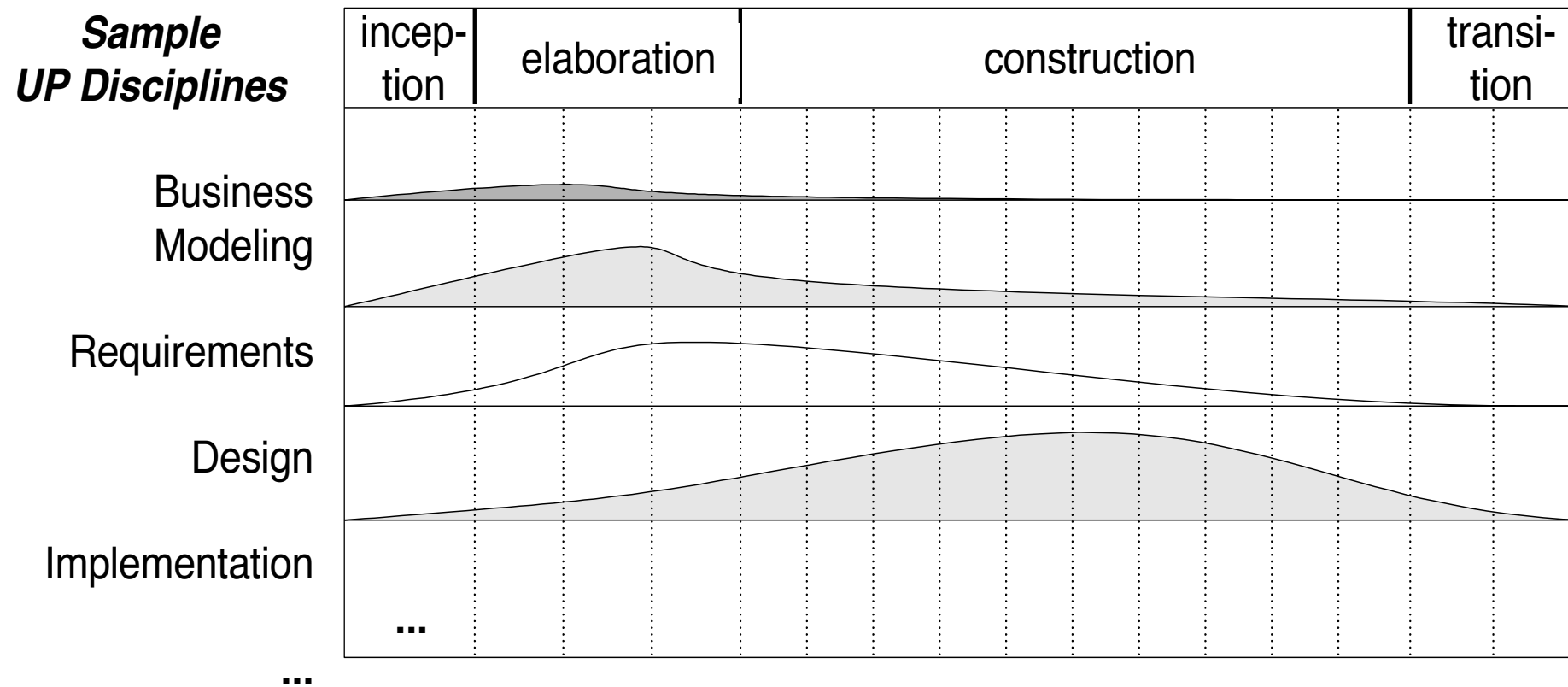
A four-week iteration (for example).
A mini-project that includes work in most disciplines, ending in a stable executable.



Note that although an iteration includes work in most disciplines, the relative effort and emphasis change over time.

This example is suggestive, not literal.

Disciplines across Phases



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

Credits

Notes and figures adapted from

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development by C. Larman. 3rd edition.
Prentice Hall/Pearson, 2005.