SWEN30006
Software Modelling and Design

ITERATIVE, EVOLUTIONARY, AND AGILE

Larman Chapter 2

You should use iterative development only on projects that you want to succeed.

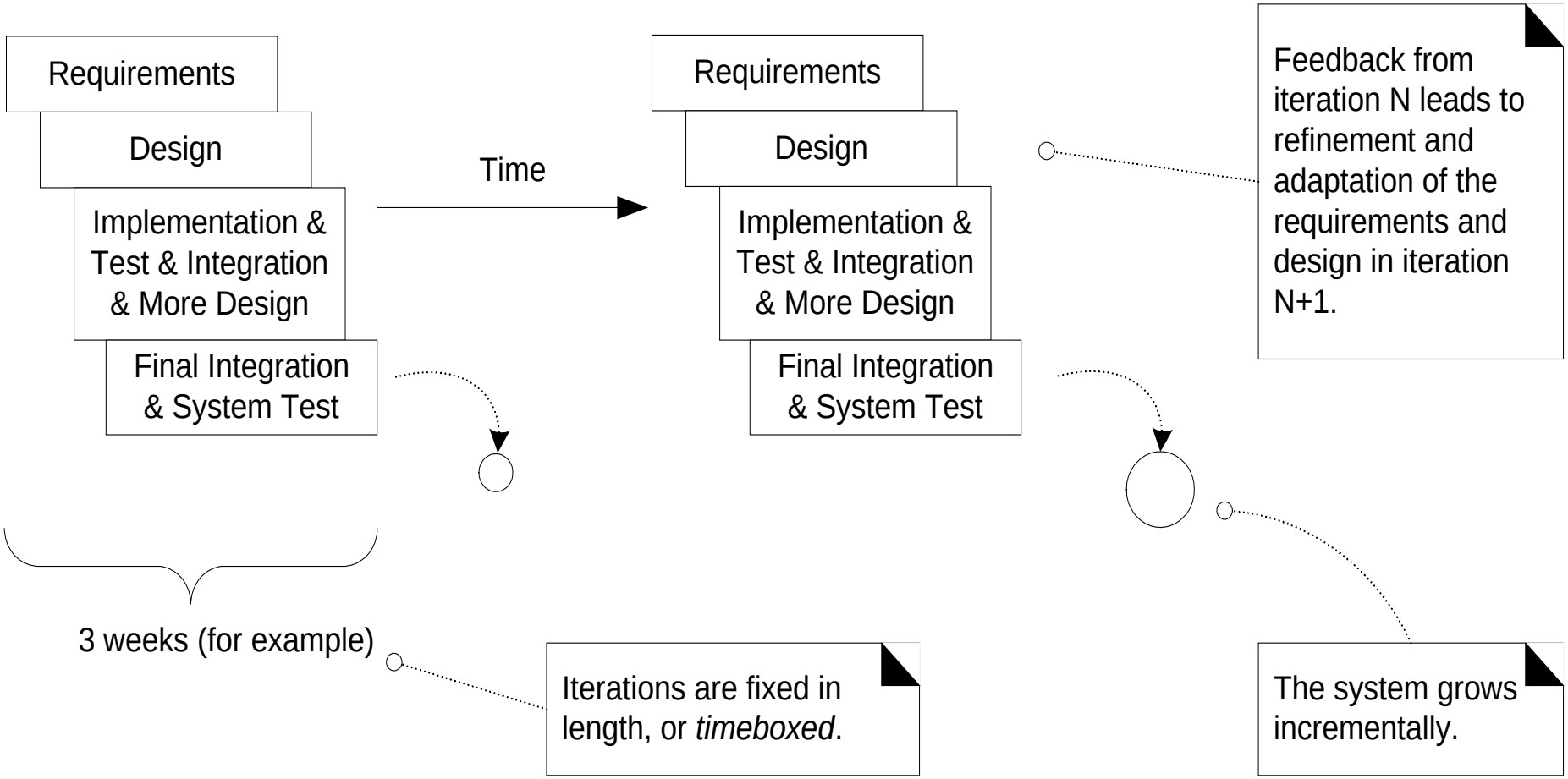
—Martin Fowler

Manifesto for Agile Software Development

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

That is, while there is value in the items on the right, we value the items on the left more.

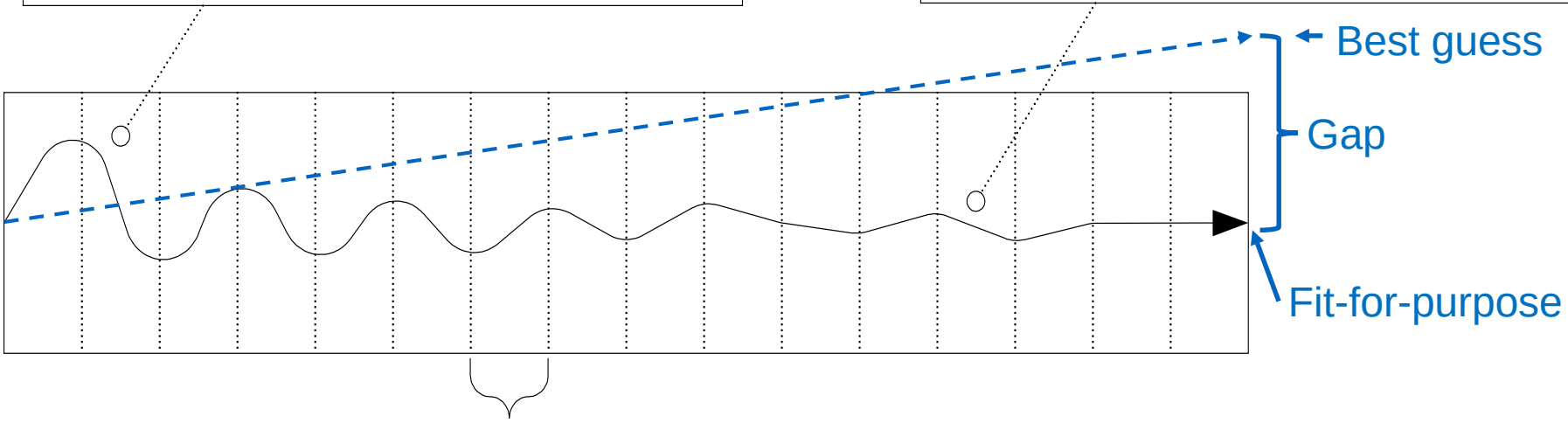
Iterative and Evolutionary Development



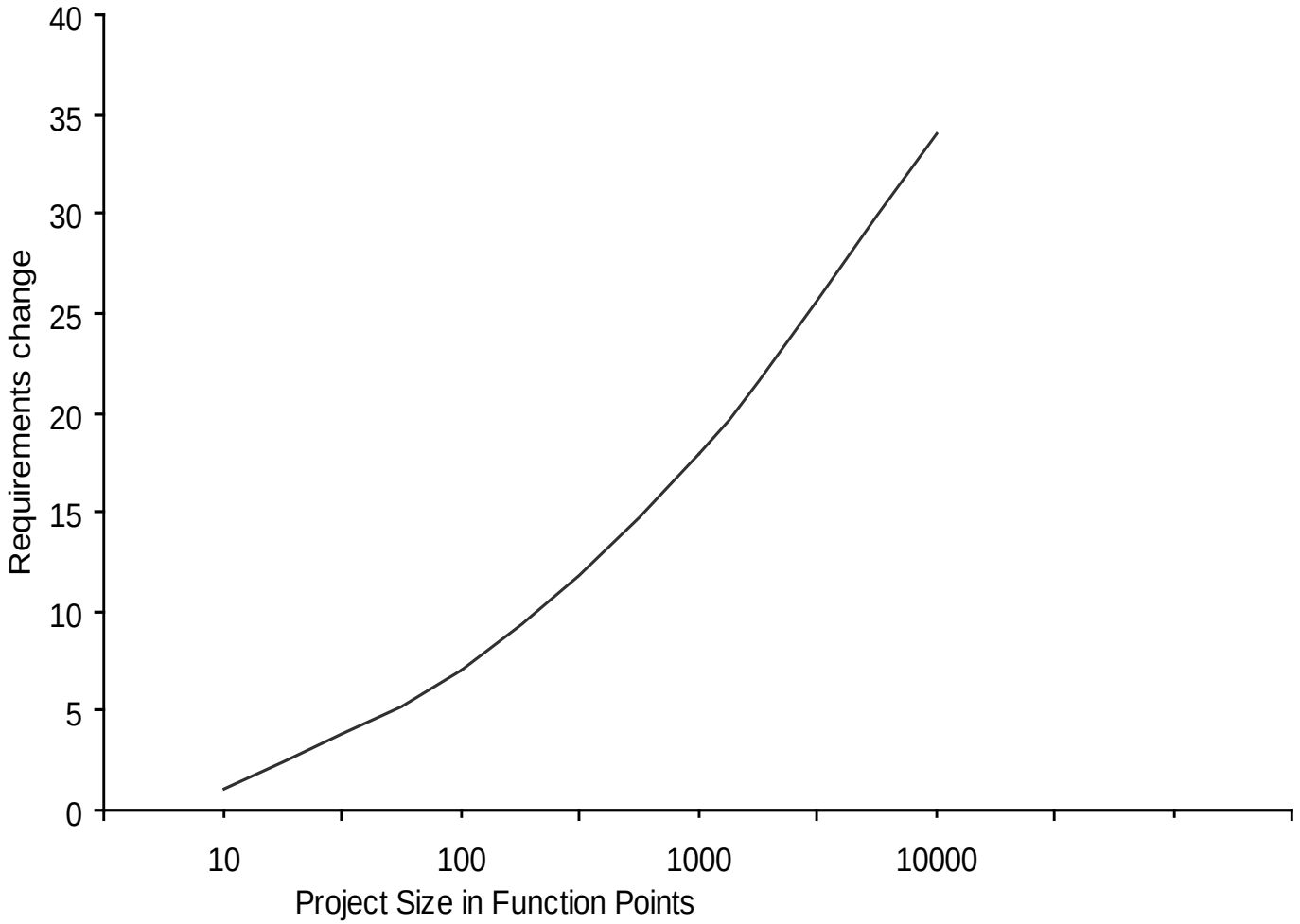
Increased Stability Through Feedback

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.

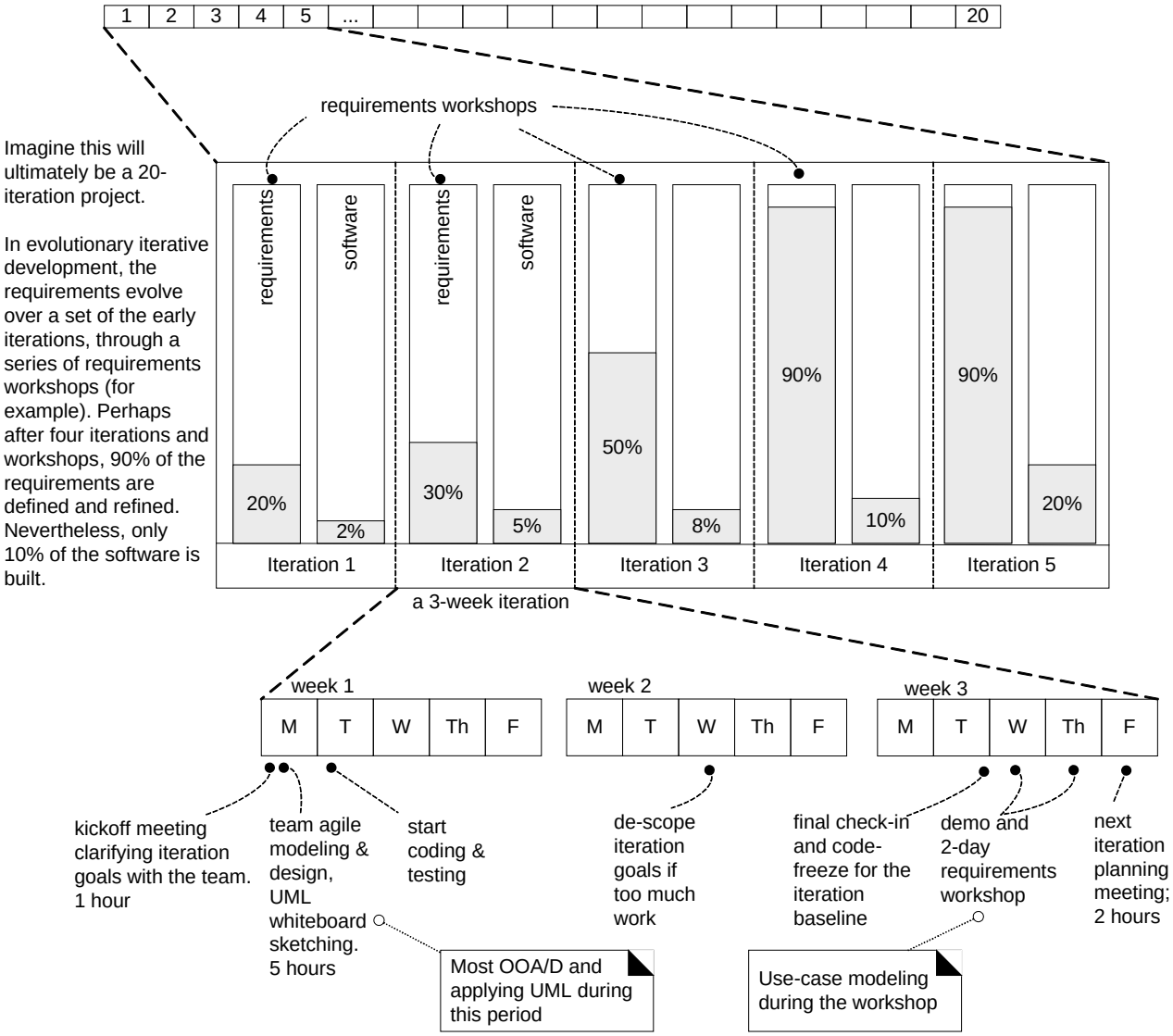
In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.



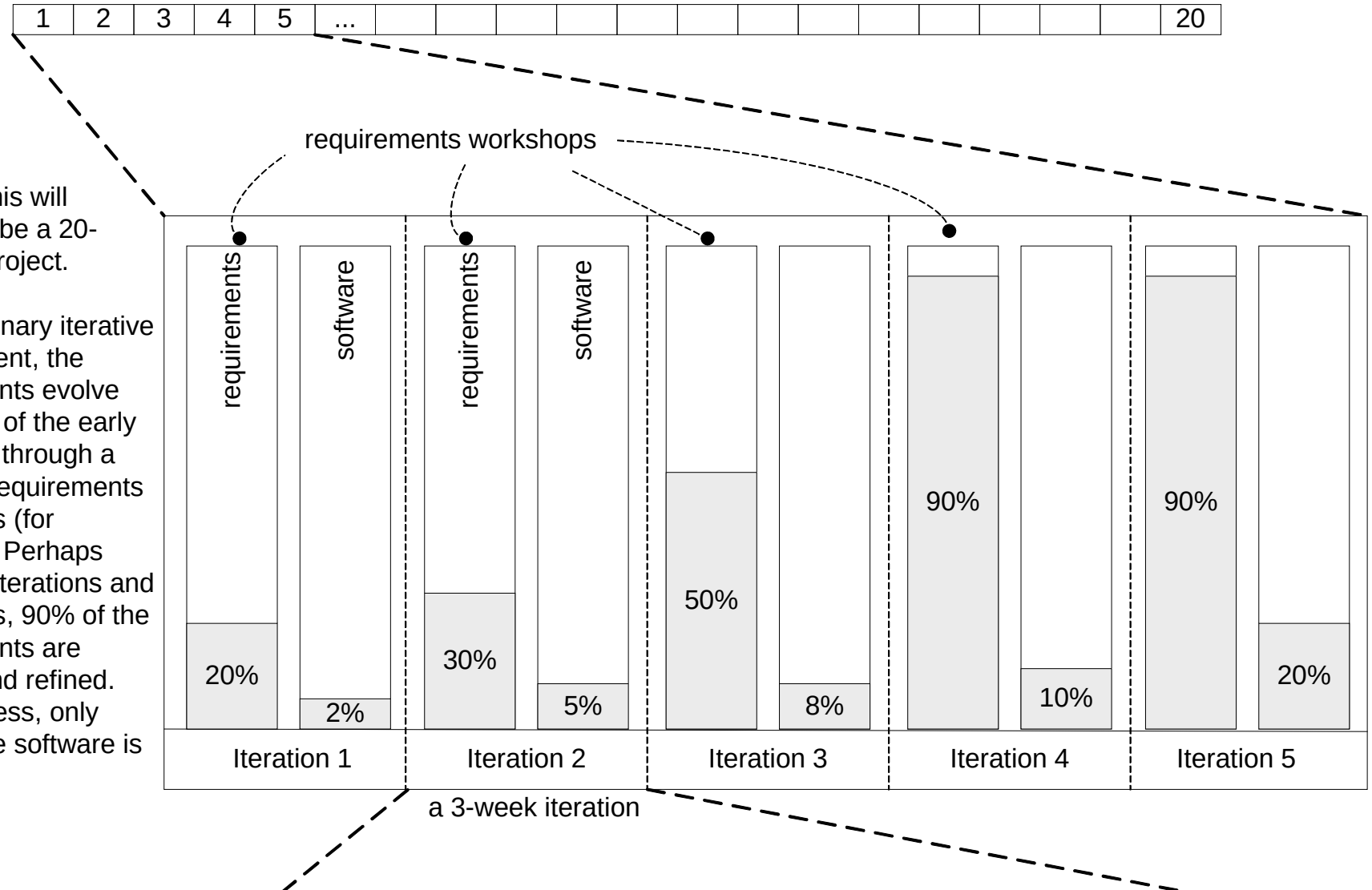
% Req. Change on S/W Projects



Evolutionary Analysis and Design



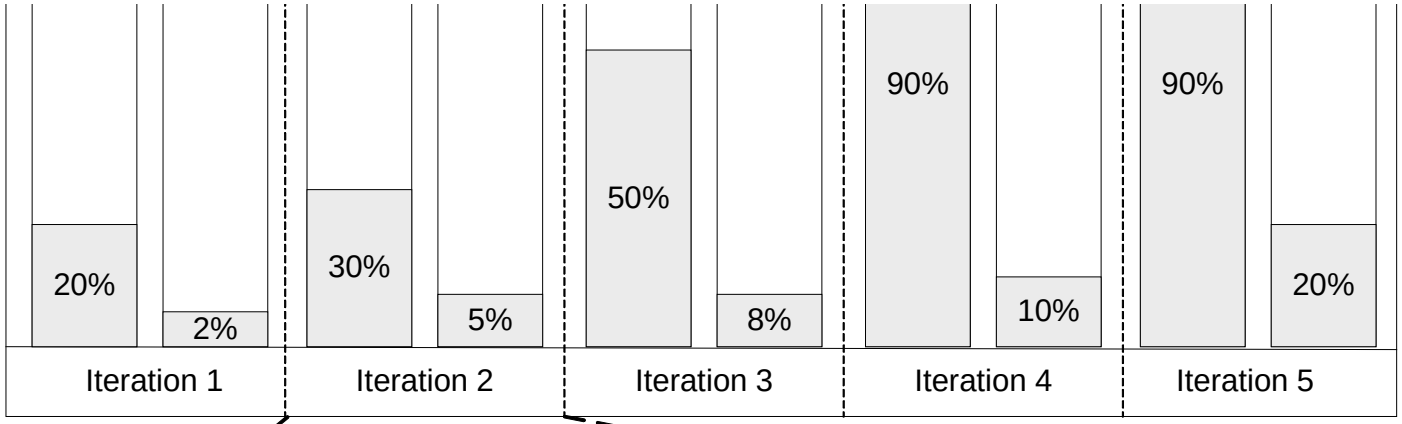
Evolutionary Analysis and Design



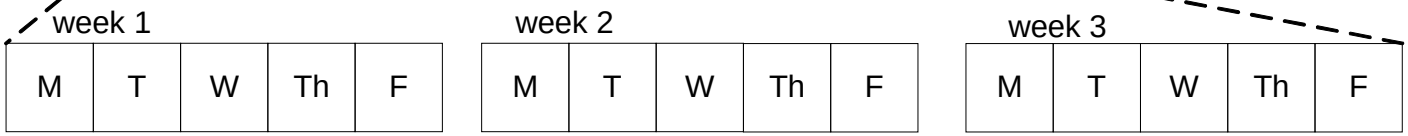
Imagine this will ultimately be a 20-iteration project.

In evolutionary iterative development, the requirements evolve over a set of the early iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.

iterations, through a series of requirements workshops (for example). Perhaps after four iterations and workshops, 90% of the requirements are defined and refined. Nevertheless, only 10% of the software is built.



a 3-week iteration



kickoff meeting clarifying iteration goals with the team. 1 hour

team agile modeling & design, UML whiteboard sketching. 5 hours

start coding & testing

de-scope iteration goals if too much work

final check-in and code-freeze for the iteration baseline

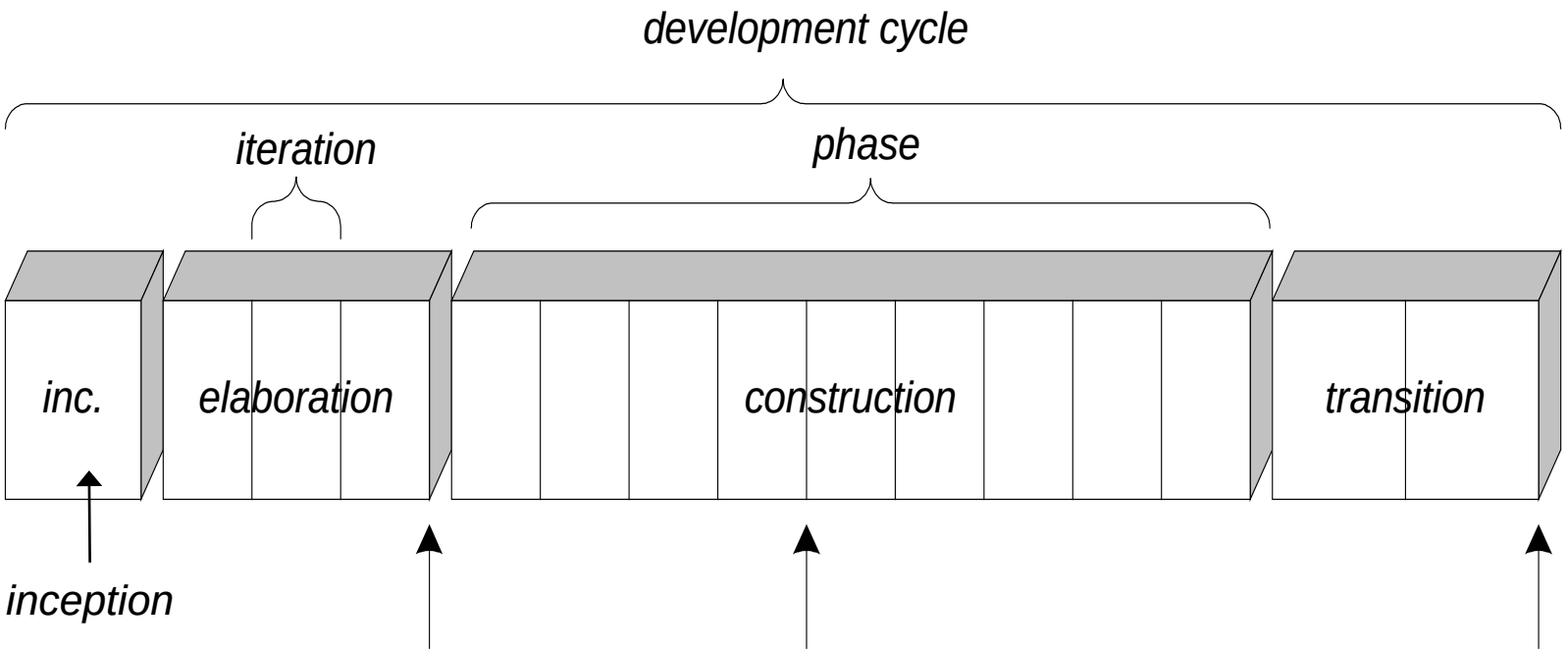
demo and 2-day requirements workshop

next iteration planning meeting; 2 hours

Most OOA/D and applying UML during this period

Use-case modeling during the workshop

Schedule-Oriented Terms in UP



milestone

An iteration end-point when some significant decision or evaluation occurs.

release

A stable executable subset of the final product. The end of each iteration is a minor release.

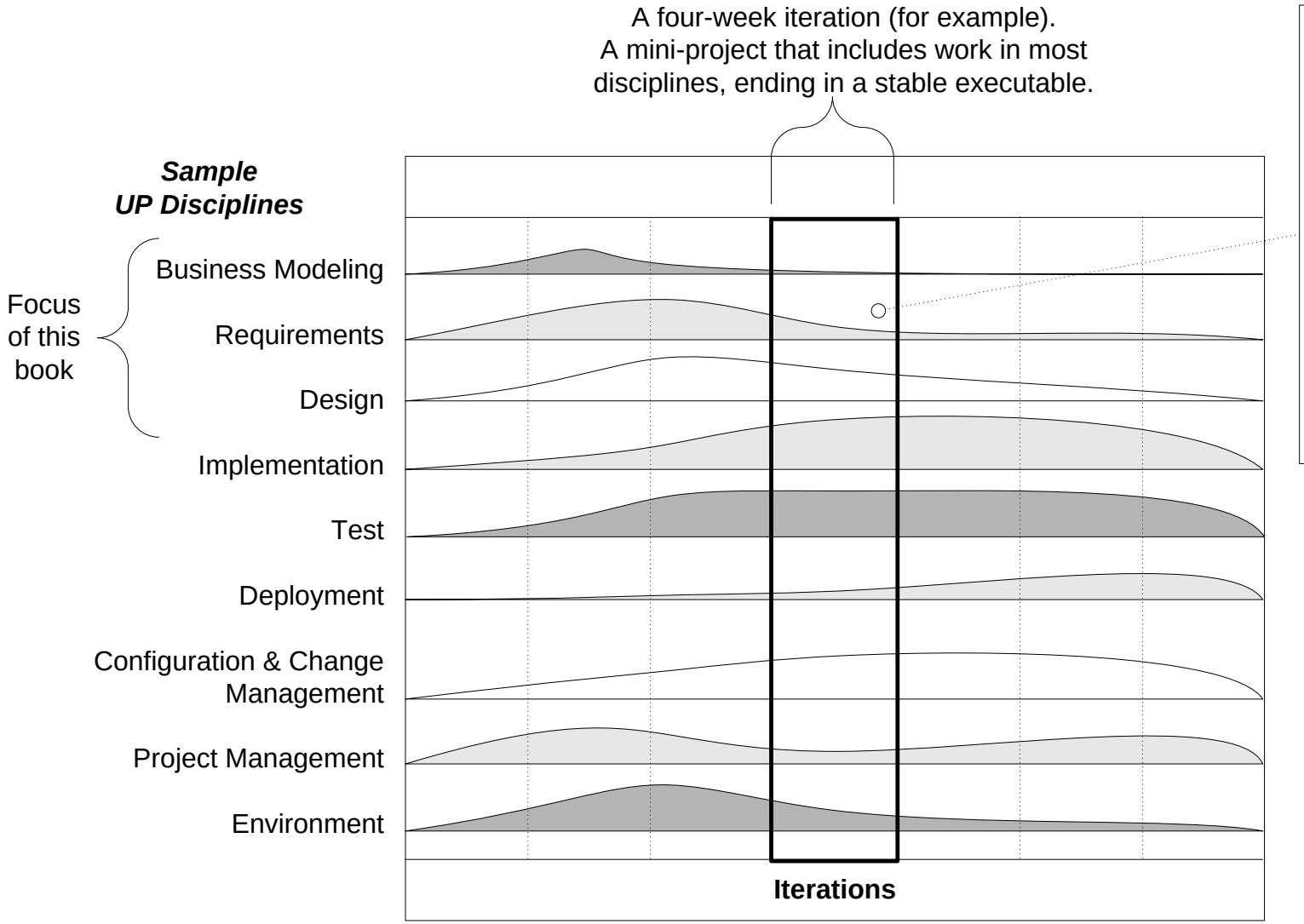
increment

The difference (delta) between the releases of 2 subsequent iterations.

final production release

At this point, the system is released for production use.

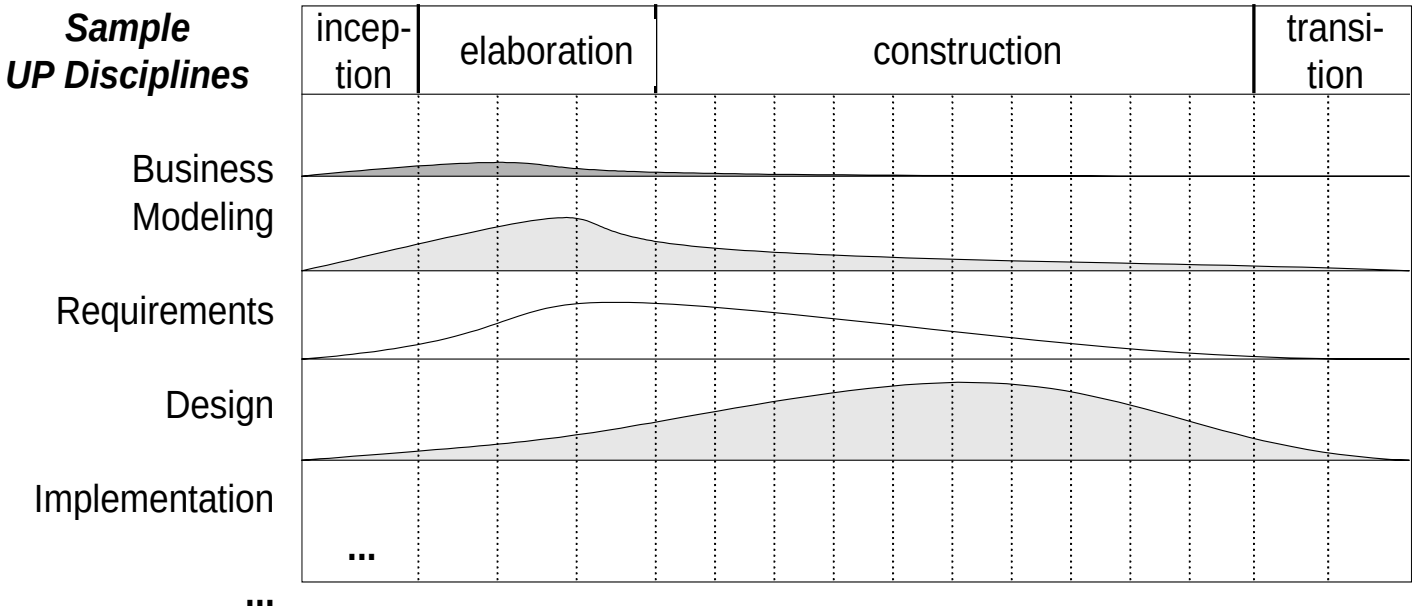
UP Disciplines



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

This example is suggestive, not literal.

UP Disciplines and Phases



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

SWEN30006
Software Modelling and Design

Inception is Not the Requirements Phase

Larman Chapter 4

Le mieux est l'ennemi du bien
(The best is the enemy of the good).

—Voltaire

What is Inception?

Initial short project phase answering questions like:

- What is the vision and business case for this project?
- Feasible?
- Buy and/or build?
- Rough unreliable range of cost: $10K–100K or $xM?
- Should we proceed or stop?

Do the stakeholders have basic agreement on the vision of the project, and is it worth investing in serious investigation?

Outcome of Inception

- Common vision and basic scope for the project
- Creation of a business case (addressing cost)
- Analysis of ~10% of use cases
- Analysis of critical non-functional requirements
- Preparation of the development environment
- (Maybe) Prototypes: clarify req's or tech. questions
- Go or no go decision.

"In preparing for battle I have always found that plans are useless, but planning indispensable"-Eisenhower

Inception Artefacts (Some, Partial)

Artefact	Comment
Vision & Business Case	Describes *high-level goals and constraints, business case*, and provides an executive summary.
Use-Case Model	Describes functional requirements. During inception, *names of most use cases will be identified; ~10% of use cases analysed in detail*.
Supplementary Specification	Describes other requirements, mostly non-functional. During inception, useful to have some idea of *key non-functional requirements with major impact on the architecture*.
Glossary	Key domain terminology, and data dictionary.
Risk List & Risk Management Plan	Describes risks (business, technical, resource, schedule) and ideas for their mitigation or response.
Prototypes & Proof-of-concepts	To clarify the vision, and validate technical ideas.
Iteration Plan	Describes what to do in the *first elaboration* iteration.
Phase Plan & Software Development Plan	Low-precision guess for elaboration phase duration and effort. Tools, people, education, and other resources.
Development Case	A description of the customized UP steps and artefacts for this project. In the UP, one always customizes for the project.

U Know U Didn't Understand Inception When...

1. You usually take more than “a few” weeks for inception.
2. You attempted to define most of the requirements.
3. You expect estimates or plans to be reliable.
4. You defined the architecture.
5. You planned the sequence of work: 1) define the requirements; 2) design the architecture; 3) implement.
6. You don’t produce a Business Case or Vision artefact.
7. You wrote all the use cases were written in detail.
8. You didn’t write any use cases in detail.

SWEN30006
Software Modelling and Design

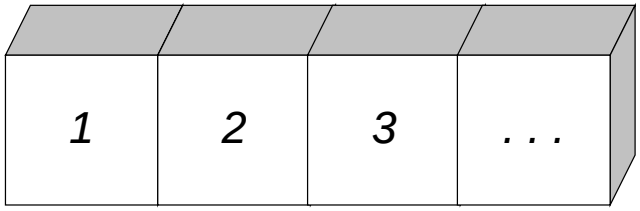
Iteration 1—Basics

Larman Chapter 8

The hard and stiff breaks. The supple prevails.

—Tao Te Ching

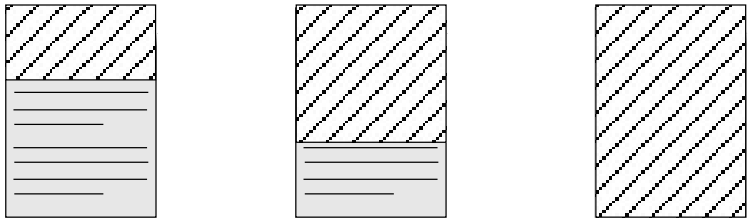
Spreading Use Cases Across Iterations



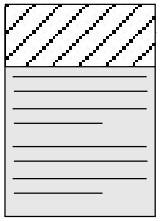
Use Case
Process Sale

Use Case
Process Sale

Use Case
Process Sale



Use Case
Process Rentals



Feature:
Logging

A use case or feature is often too complex to complete in one short iteration.

Therefore, different parts or scenarios must be allocated to different iterations.

SWEN30006
Software Modelling and Design

Requirements to Design—Iteratively

Larman Chapter 12

***Hardware**, n.: The parts of a computer system that can be kicked.*

—anonymous

Requirements to Design—Iteratively

- Iteratively Do the Right Thing, Do the Thing Right
  - Requirements: do the right thing (c.f. validation)
  - Design: do the thing right (c.f. verification)
- Provoking Early (inevitable) Change
  - Don’t just passively “embrace change”
- Didn't All That Analysis and Modelling Take Weeks To Do?
  - A few hours or days: use case writing, domain modelling
  - A few weeks: proof-of-concept dev., finding resources, planning, setting up the environment