Applying UML and Patterns

An Introduction to
Object-oriented Analysis
and Design
and Iterative Development

Part I - Introduction

Chapters

1. Object oriented analysis and design
2. ***Iterative, evolutionary, and agile***
3. Case study

Text book, page 3-44

★★★

回顾：软件工程

- 软件工程定义
  - IEEE：Software Engineering are（1）“the application of a systematic 系统化, disciplined 规范化, quantifiable 可度量 approach to the development, operation, and maintenance of software”；（2）在（1）中所述方法的研究。
- 软件工程知识体系（SWEBOK）
  - 以高质量、低成本为目标，研究软件生产的过程模型、方法与工具

http://en.wikipedia.org/wiki/Software_engineering

软件工程知识体系内涵

- 软件工程领域中的核心知识包括：
  1. 软件需求（Software requirements）
  2. 软件设计（Software design）
  3. 软件建构（Software construction）
  4. 软件测试（Software test）
  5. 软件维护与更新（Software maintenance）
  6. 软件配置管理（Software Configuration Management, SCM）
  7. 软件工程管理（Software Engineering Management）
  8. 软件开发过程（Software Development Process）
  9. 软件工程工具与方法（Software Engineering Tools and methods）
  10. 软件质量（Software Quality）

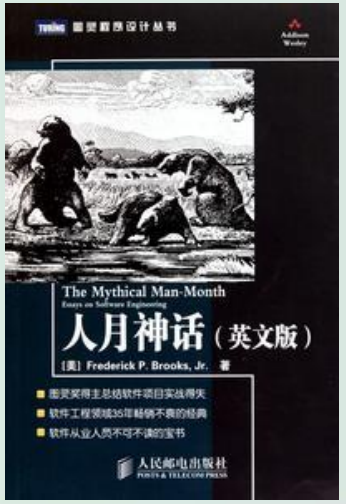
高品质程序标准

- 满足客户要求，客户**愿意使用**
- 良好的设计和代码，便于**维护、重用、扩展**

推荐读物

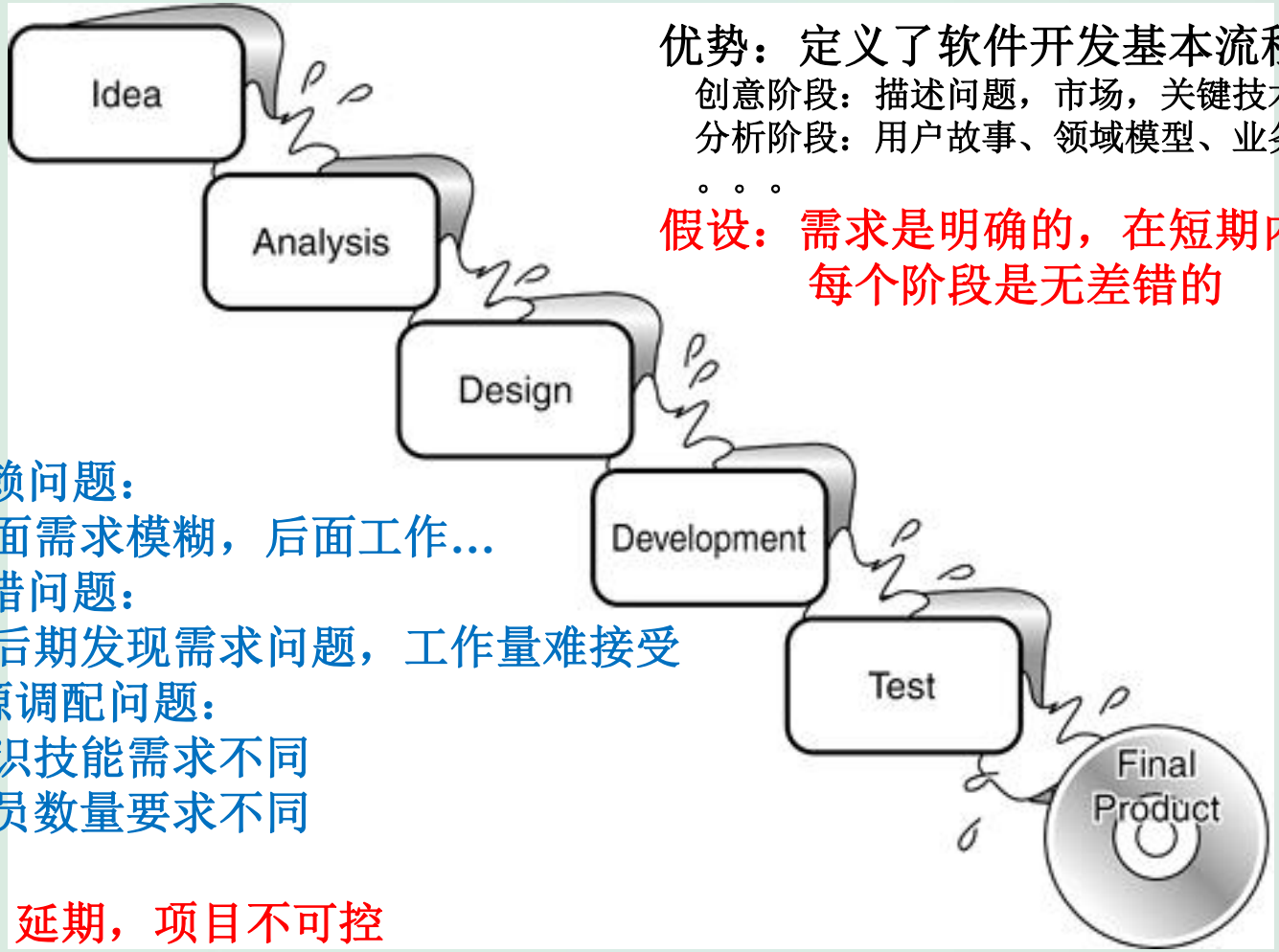
- 彼得·德鲁克（Peter F．Drucker)
  - 基本管理问题（知识型团队）
    - 重视目标与成果
    - 知识的权威（知识工作者即为管理者）
    - 时间的管理
    - 团队的组织与协作（技能互补团队）
  - 组织的目的是使平凡的人做出不平凡的事

- 布鲁克斯(FrederickP.Brooks.Jr.)
  - 复杂软件的问题
    - 管理问题
    - 产品定义问题
    - 设计问题
  - 前人的经历是后人的财富



过程的重要性与CMMI

回顾：waterfall model 瀑布模型



优势：定义了软件开发基本流程与活动

创意阶段：描述问题，市场，关键技术等

分析阶段：用户故事、领域模型、业务流程等

。。。

假设：需求是明确的，在短期内可获取

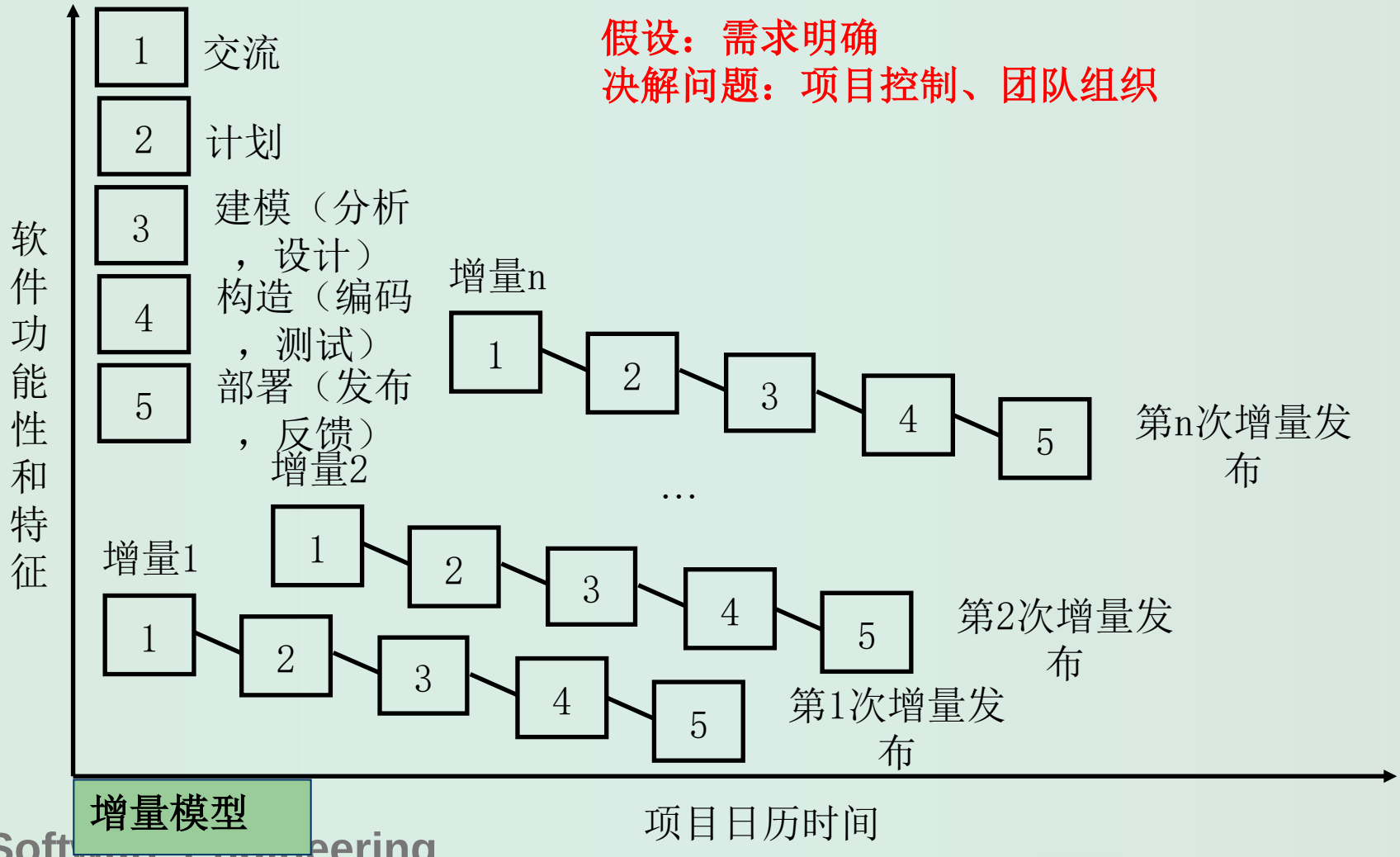
每个阶段是无差错的

劣势：

1. 依赖问题：
   -- 前面需求模糊，后面工作…
2. 容错问题：
   -- 在后期发现需求问题，工作量难接受
3. 资源调配问题：
   -- 知识技能需求不同
   -- 人员数量要求不同

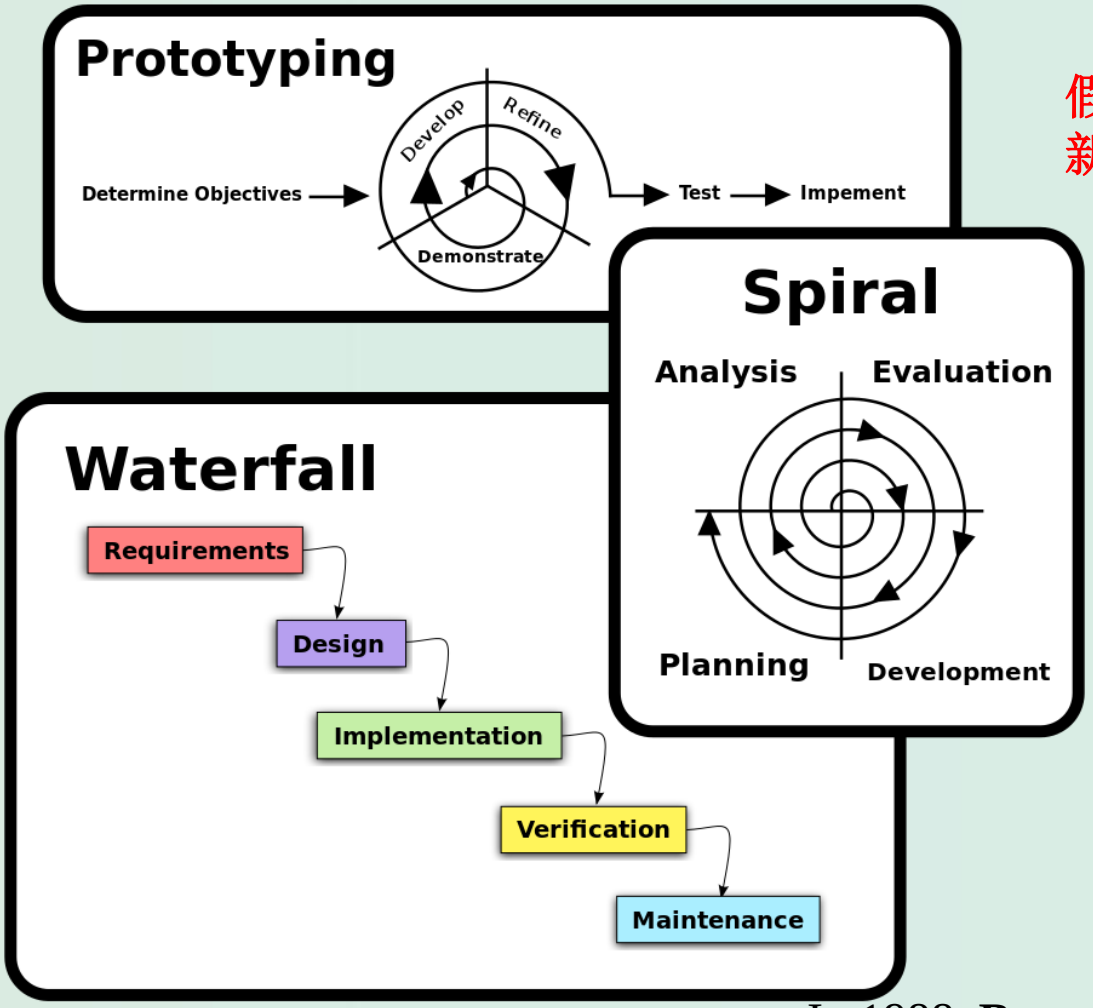
现象：延期，项目不可控

增量模型


假设：需求明确
决解问题：项目控制、团队组织
软件功能性和特征
1 交流
2 计划
3 建模（分析，设计）
4 构造（编码，测试）
5 部署（发布，反馈）
增量1
增量2
…
增量n
第1次增量发布
第2次增量发布
第n次增量发布
项目日历时间

增量模型

原型与螺旋模型



假设：需求随用户评估进化

新问题：

1. 迭代次数？
2. 不合适大团队
3. 无法确定发布日期

In 1988, **Barry Boehm** published a formal software system development "spiral model,"

Chap 2
Iterative, Evolutionary, and Agile

迭代、敏捷地开发

- 为什么迭代、敏捷开发是OOAD的关键“最佳实践”

Unified Process

- The Unified Process (UP) represents a mainstream approach for software development across the spectrum of project scales.
- The process is **scalable**: you need not use the entire framework of the process for every project, only those that are effective.
- The process is **effective**: it has been successfully employed on a large population of projects.
- Improves **productivity** through use of practical methods that you've probably used already (but didn't know it).
- Iterative and incremental approach allows start of work with incomplete, imperfect knowledge.

Unified Process Workflows

- Workflows define a set of activities that are performed
- Workflows cut across the phases, but with different levels of emphasis in each phase
- The core workflows
  - Business Modeling
  - Requirements analysis
  - Design
  - Implementation
  - Test and Integration

The Core Workflows 1

- Business Modeling
  - Develop and refine the problem definition
  - Identify stakeholder needs
  - Define system features to be considered
  - Define the system scope
  - Build the use-case model
- Requirements Analysis
  - Refine use-case model
  - Define the domain model
  - Define a candidate architecture (transitions to design)
  - Refine the architecture (transitions to design)

The Core Workflows 2

- Design
  - Design the physical realizations of the use cases
  - Develop the design model
  - Develop the deployment model
- Implementation
  - Plan subsystem implementation
  - Implement components: classes, objects, etc.
  - Perform unit-level testing
  - Perform component and system integration
- Test and Integration
  - Build the test model: test cases and expected results
  - Plan, design, implement, execute, and evaluate tests

Use Case Driven

- Use case
  - A prose representation of a sequence of actions
  - Actions are performed by one or more *actors* (human or non-human) and the system itself
  - These actions lead to valuable results for one or more of the actors—helping the actors to achieve their goals
- Use cases are expressed from the perspective of the users, in natural language, and should be understandable by all stakeholders
- ***Use-case-driven*** **means the development team employs the use cases from requirements gathering through code and test**

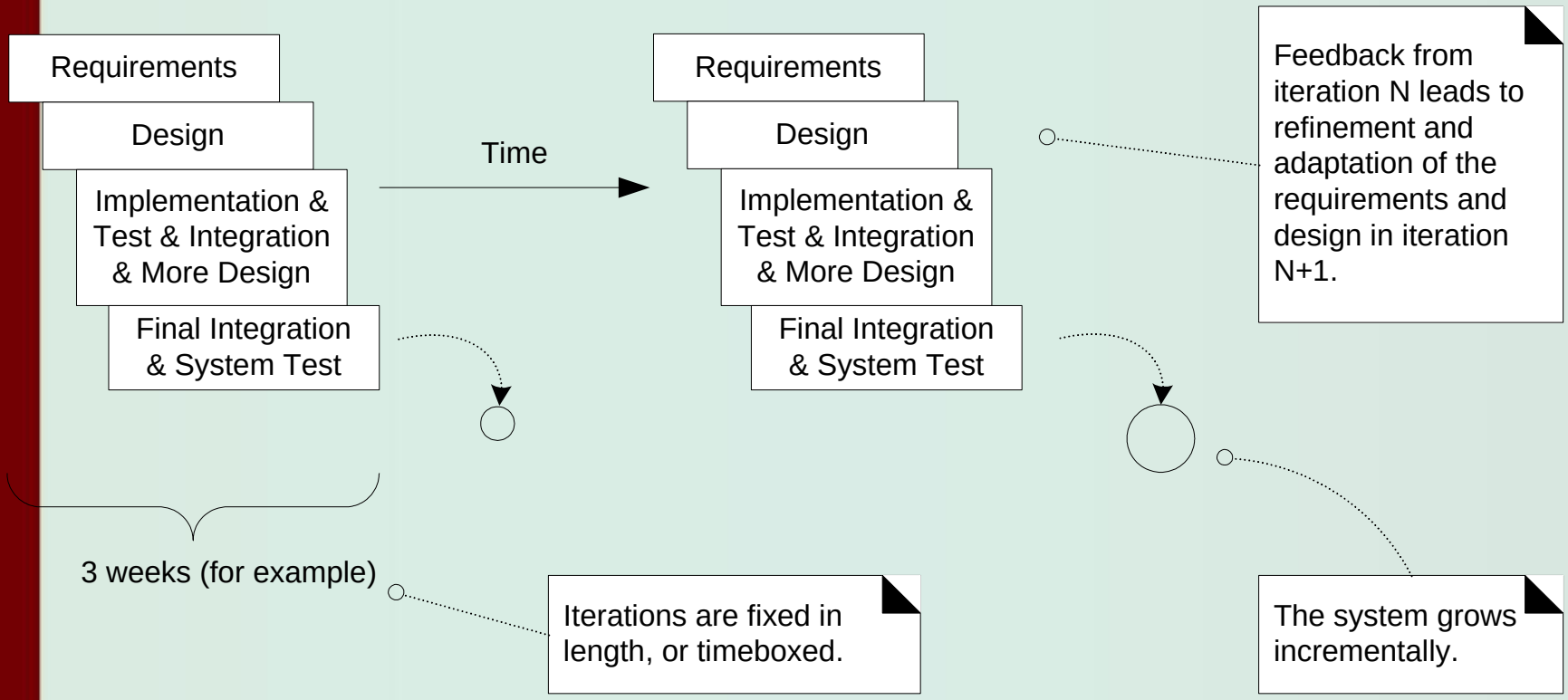
Architecture Centric

- Software architecture captures decisions about:
  - The **overall** structure of the software system
  - The structural elements of the system and their **interfaces**
  - The **collaborations** among these structural elements and their expected behavior
- Architecture-centric: software architecture provides the central point around which all other development evolves
  - Provides a 'big picture' of the system
  - Provides an organizational framework for development, evolving the system by attending to modifiability qualities of the system
  - Facilitates reuse

Iterative and Evolutionary 1

- An iterative and evolutionary approach allows start of development with **incomplete, imperfect knowledge**
- Iterative and evolutionary the following advantages:
  - Logical **progress** toward a robust architecture（逐步趋向稳定）
  - Effective management of **changing requirements**（有效管理需求变化）
  - Continuous integration（持续集成）
  - Early understanding of the system **(‘Hello world!’ effect)**（尽早接触整个系统）
  - Ongoing risk assessment（在线风险评估）

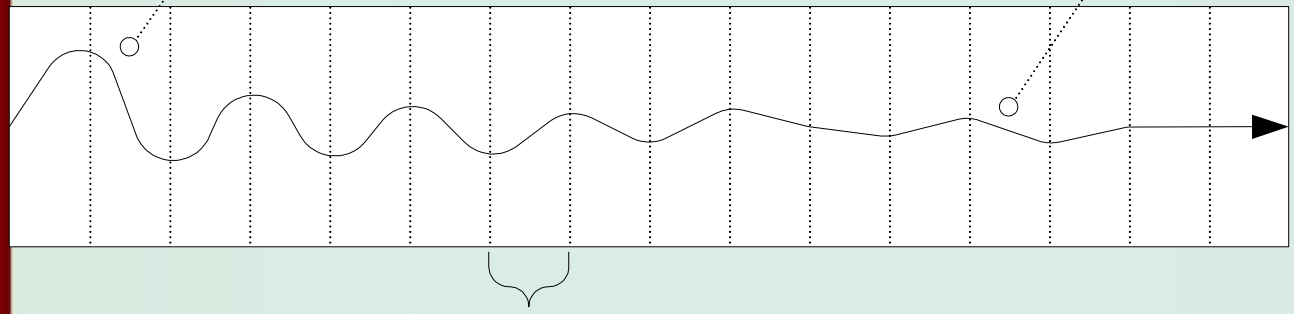
Iterative and Evolutionary 2



Iterative and Evolutionary 3

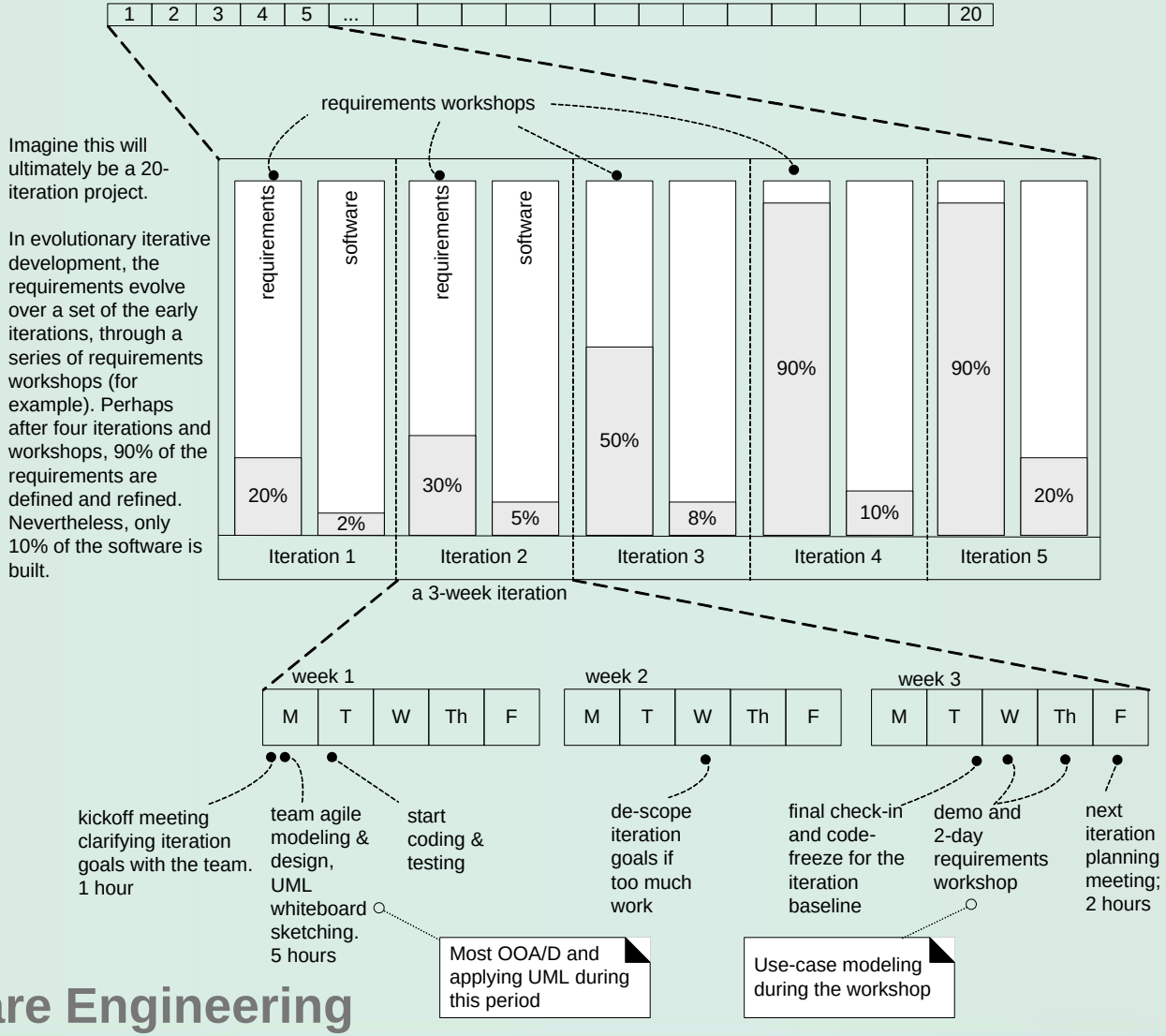
Early iterations are farther from the "true path" of the system. Via feedback and adaptation, the system converges towards the most appropriate requirements and design.

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

Iterative and Evolutionary 4



迭代UP vs. Scrum术语

Agile Iterative UP	Scrum
Iteration（迭代周期）	Sprint（冲刺）
Use cases（用例）或 Story（故事）	Backlog（产品条目，特征）
Kickoff meeting（启动会议）	Sprint plan meeting
Workshop（工作会议）	Daily meeting（每日例会）
Demo（演示）	Sprint demo
Review & next Plan（回顾与下个迭代计划）	Sprint review（冲刺回顾）
Coffee Time（私下交流时间）	
Phase（阶段）	
Disciplines（科目）	
Mile stone（里程碑）	

Agile Methods and Attitudes

- Agile Methods:
  - However, short timeboxed iterations with evolutionary refinement of plans, requirements, and design
  - Other:simplicity, lightness, communication, self-organizing teams, etc.
  - Scrum, XP……

★

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

Agile Methods and Attitudes

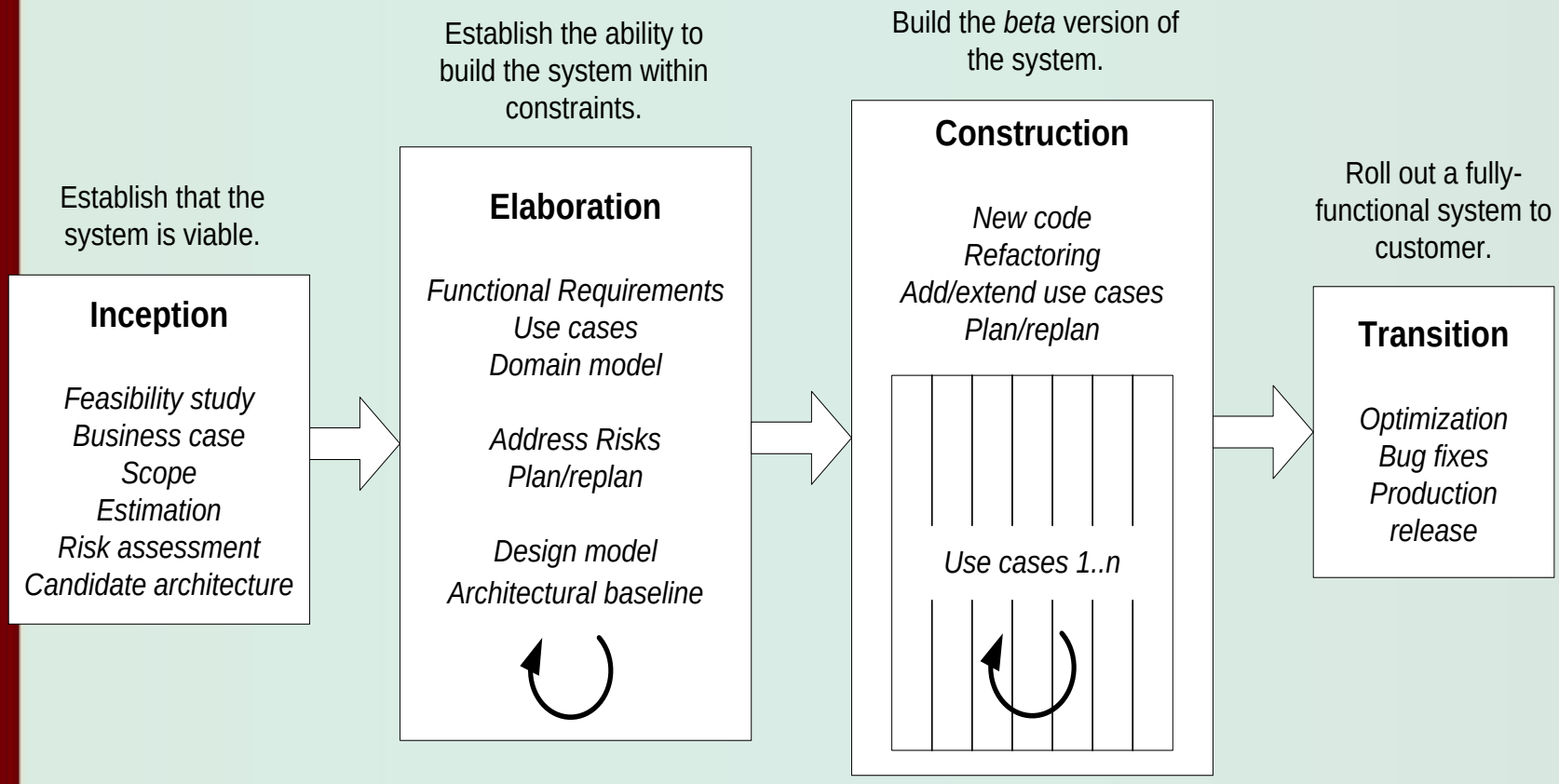
The Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development.
3. Deliver working software frequently.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals.
6. Face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development.
9. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
10. Continuous attention to technical excellence and good design enhances agility.
11. Simplicity-the art of maximizing the amount of work not doneis essential.
12. The best architectures, requirements, and designs emerge from self-organizing teams.
13. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

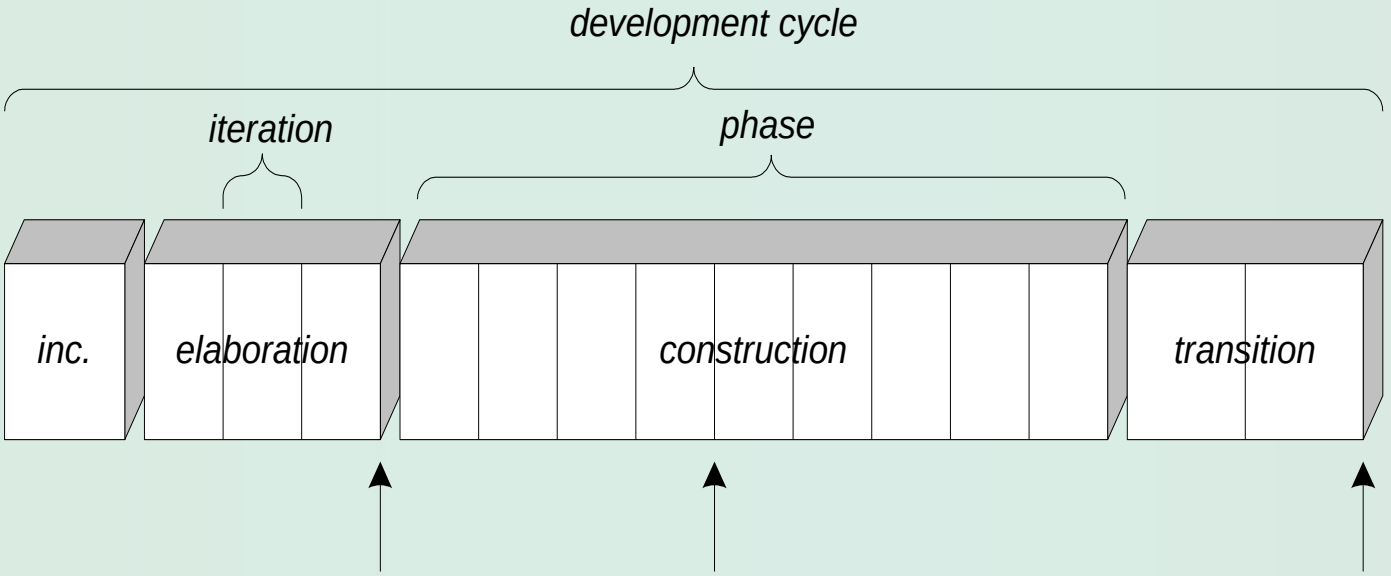
Agile Modeling

- Adopting an agile method does not mean avoiding any modeling
- The purpose of modeling and models is primarily to support understanding and communication, not documentation.
- Don't model or apply the UML to all or most of the software design.
- Use the simplest tool possible.
- Don't model alone, model in pairs (or triads) at the whiteboard.
- Create models in parallel.
- Use "good enough" simple notation while sketching with a pen on whiteboards.
- Know that all models will be inaccurate, and the final code or design different sometimes dramatically different than the model.
- Developers themselves should do the OO design modeling, for themselves.

Unified Process Phases 1



Unified Process Phases 2



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.

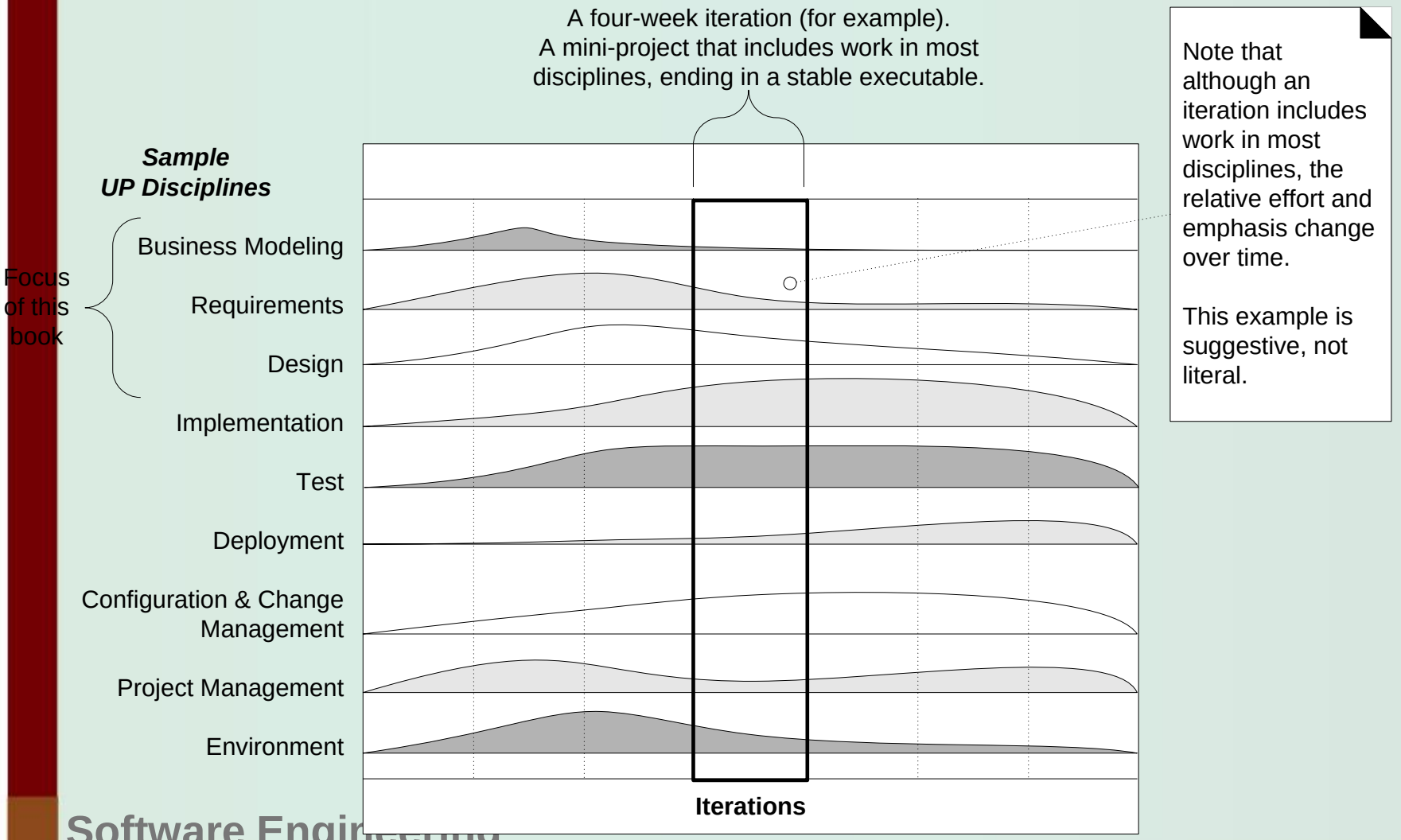
increment

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

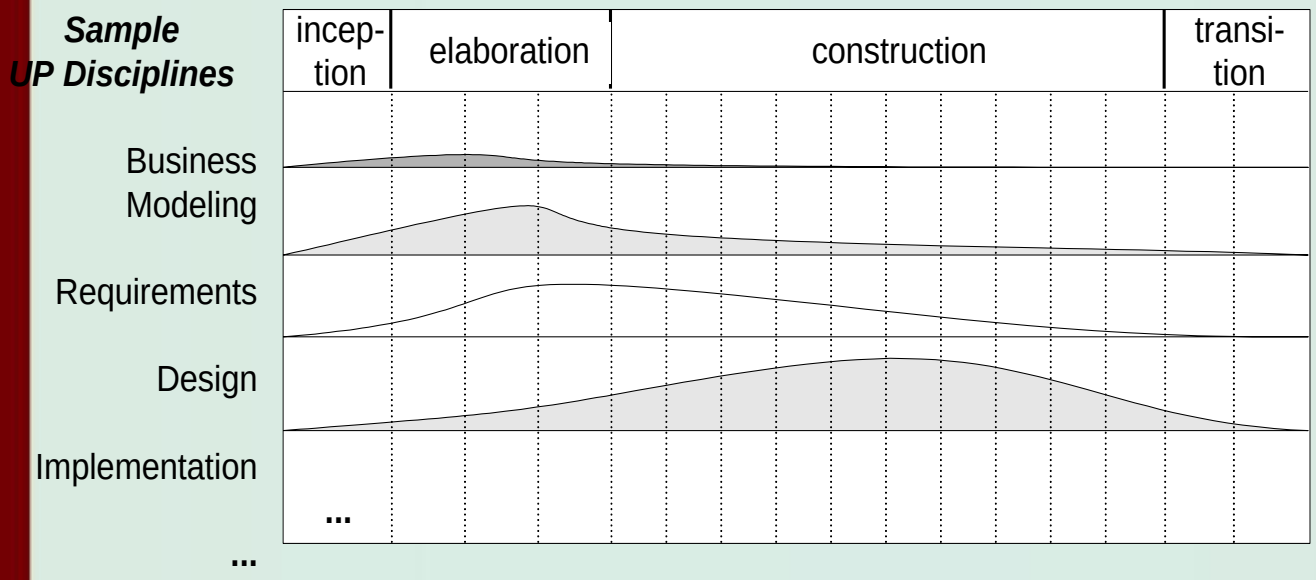
final production release

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

Core Workflows and Phases 1



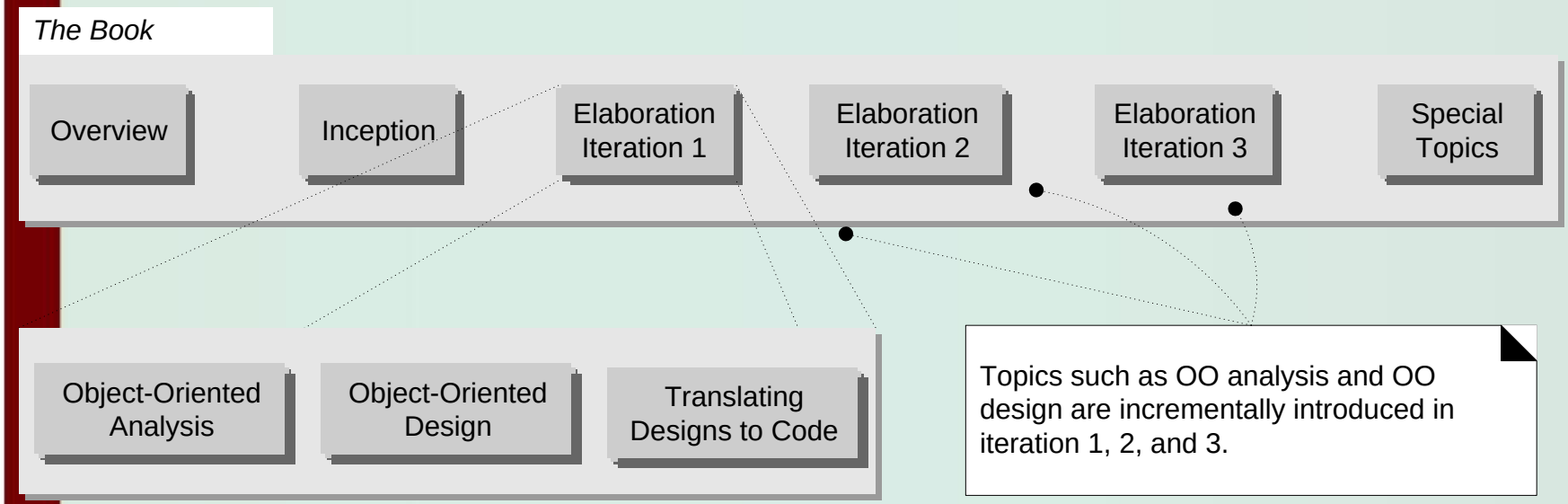
Core Workflows and Phases 2



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

Core Workflows and Phases 3



Artifacts, Workers, and Activities 1

- An artifact is a piece of information that is used as input to, changed by, or output from a process
- Examples include:
  - Models — use-case, domain, and design
  - Model elements—use case, domain class, design class
  - Diagrams and documents
  - Source code
  - Executable elements

Artifacts, Workers, and Activities 2

- Workers define the behavior and responsibilities of an individual or a team
  - Examples: Architect, use-case engineer, component engineer, system integrator
- Some important distinctions:
  - Workers participate in the development of the system
  - Actors are outside the system and have usage relationships with the system
  - Stakeholders encompass both actors and workers, as well as others involved with the project

Artifacts, Workers, and Activities 3

- Activities are the tasks performed within a workflow
- Activities can describe a wide range of abstraction levels, from high-level ('construct domain model') to low-level ('implement class')
  - Examples include:
  - Plan iteration
  - Find use cases and actors
  - Execute integration test
  - Review test results

The Agile Unified Process

- The Unified Process has been designed from the outset as:
  - *Lightweight*: 'Pay as you go.' Use only the parts that are essential and effective for your project. When in doubt, leave it out.
  - *Non-predictive*: Requirements and design build gradually as development proceeds rather than being completed before any work can begin.
  - *Adaptable*: Planning and risk analysis/assessment are on-going and process can be adapted accordingly.

Sample Development Case

Discipline	Practice	Artifact	Incep.	Elab.	Const.	Trans.
		Iteration	I1	E1..En	C1..Cn	T1..T2
Business Modeling	agile modeling req. workshop	Domain Model		s		
Requirements	req. workshop vision box exercise dot voting	Use-Case Model	s	r		
		Vision	s	r		
		Supplementary Specification	s	r		
		Glossary	s	r		
Design	agile modeling test-driven dev.	Design Model		s	r	
		SW Architecture Document		s		
		Data Model		s	r	
Implementation	test-driven dev. pair programming continuous integration coding standards	…				
Project Management	agile PM daily Scrum meeting	…				
…						

过程总结

- 过程（资产）建设在软件企业中的重要性
  - ISO-9000，CMMI 认证
- 瀑布模型、增量模型、螺旋模型的优缺点
  - 风险（需求），规模（工作量），工期的控制分析
- UP 工作流与瀑布模型的关系
- UP 的三大特征
- UP 的四个阶段，阶段的划分与增量模型、螺旋模型的关联，它是如何克服基本模型的缺陷的？
- UP 与敏捷实践的关系
- UP 是固定周期软件生产模型？
- UP 的术语