

Chapter 2



迭代、进化和敏捷

本章目标

- 定义迭代(**iterative**)过程和敏捷(**agile**)过程
 - 迭代/瀑布
 - 敏捷/重型
- 定义统一过程中的基本概念

软件过程

□ 什么是软件过程

软件过程定义了软件开发、部署和维护的步骤。

□ 软件过程本身就是软件

软件过程是一种被由人构成的虚拟机执行的软件。

□ 软件过程为什么重要（为什么不应该那么重要）

软件过程的谱系

□ 软件过程

软件过程描述开发、部署和维护软件系统的步骤。

□ 迭代式开发

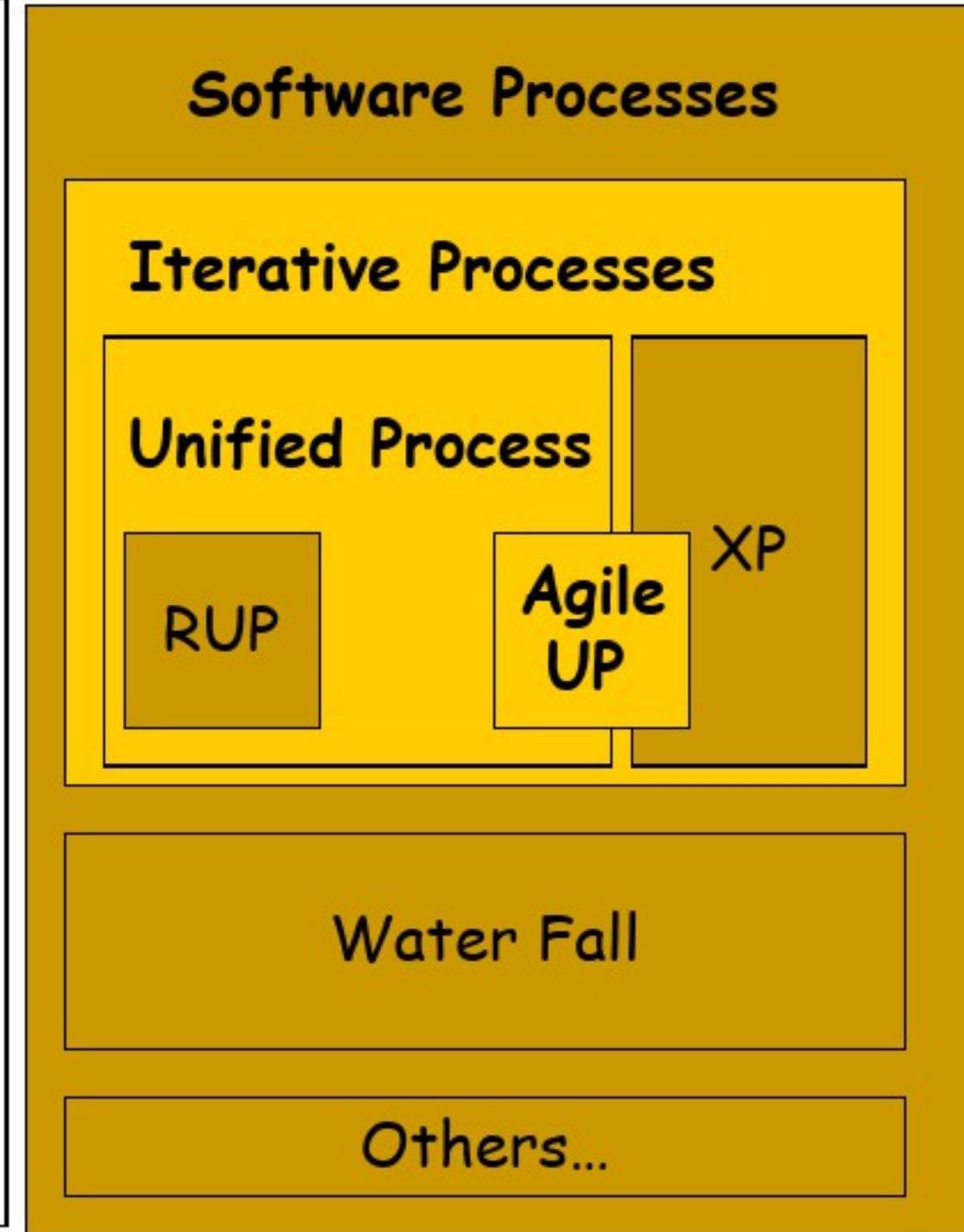
迭代式开发将软件开发过程分解为一系列小的，固定周期的(比如，4个星期)的小项目，每个小项目称为一个迭代。

□ 统一过程 (**Unified Process**)

一种采用OOA/D方法学开发项目的过程(Ivar Jacobson)。

□ 敏捷建模UP(**Agile UP**)

引入了敏捷概念的UP,是UP的一个简集。



迭代式开发

□ 瀑布生命周期

- 在瀑布生命周期过程中，试图在编写代码之前定义几乎所有的需求，以及明确详尽的时间表。

□ 迭代式的生命周期

- 通过多次的迭代获得周期性的反馈，以这些反馈为驱动力，对系统进行不断的扩展和精化。
- 迭代式开发将软件开发过程分解为一系列小的，固定周期的(比如，4个星期)的小项目，每个小项目称为一个迭代。

迭代式开发

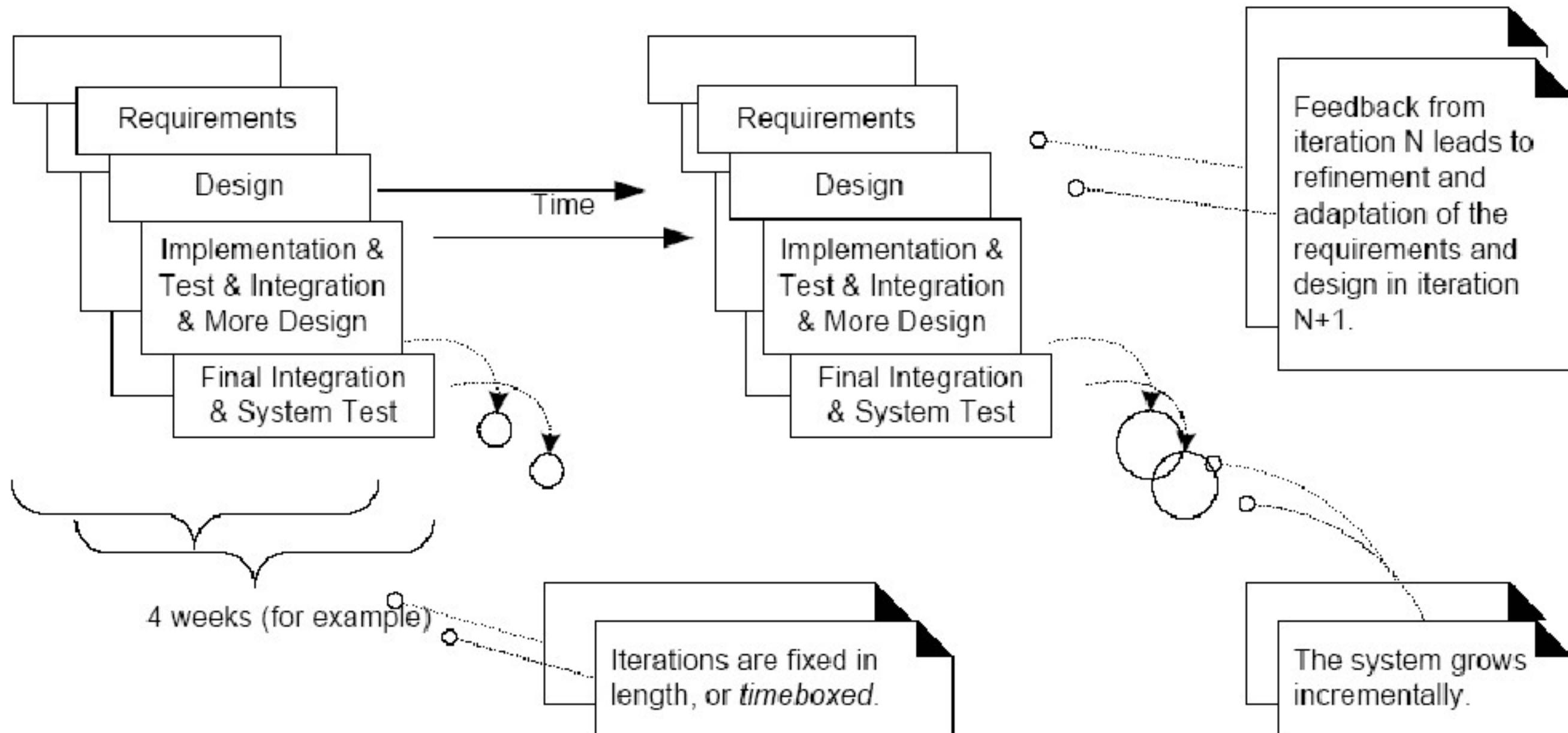


Figure 2.1 Iterative and incremental development.

每一次迭代的周期

- 迭代的一个关键思想是时间定量，即时间长度固定。
- 大部分迭代方法建议迭代时间在**2**到**6**周之间。

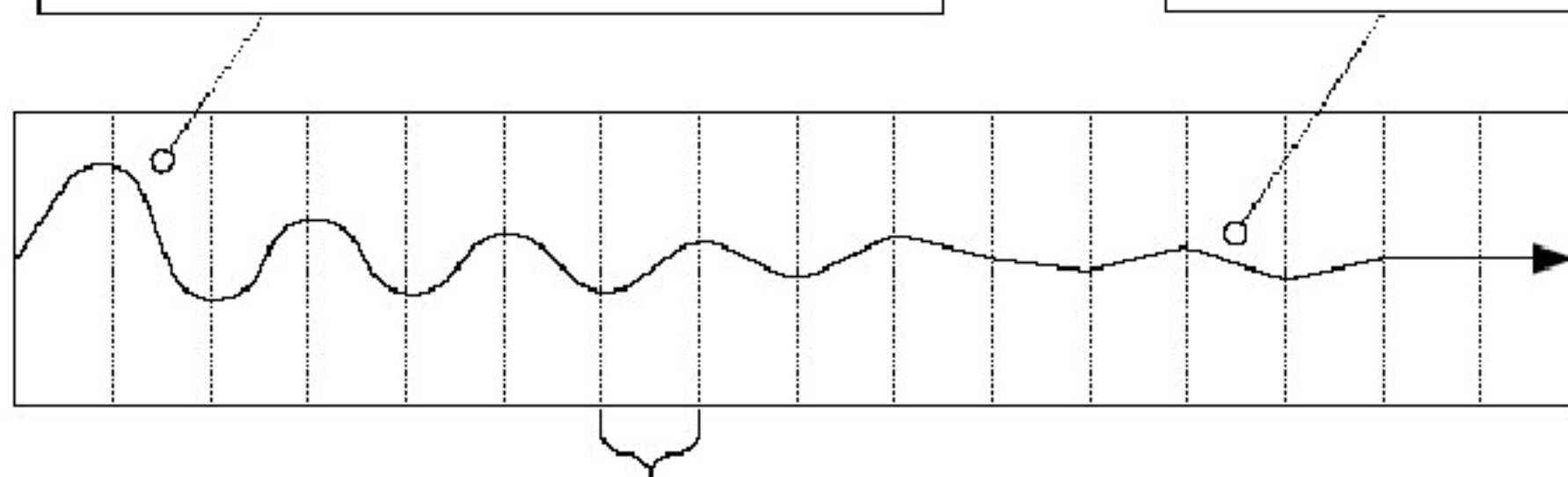
示例

- 在项目开始为期3周的迭代中
 - 周一启动会议，明确本次迭代的任务和目标。其间一小时制作**UML图**，打印最重要的部分。
 - 其他时间团队成员结对在白板上用**UML图建模**。
 - 开发，测试。
 - 发布，给客户**Review**本次迭代的成果，获取反馈。
 - 计划下一次的迭代。
- 注意：
 - 没有匆忙地开始编码，也没有长期的，试图完全定义系统的设计。
 - 迭代的成果不是用完后就抛弃的原型，而是最终产品的子集。
 - 获取用户反馈并不断改进是项目的主要驱动力量。

迭代的过程

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.

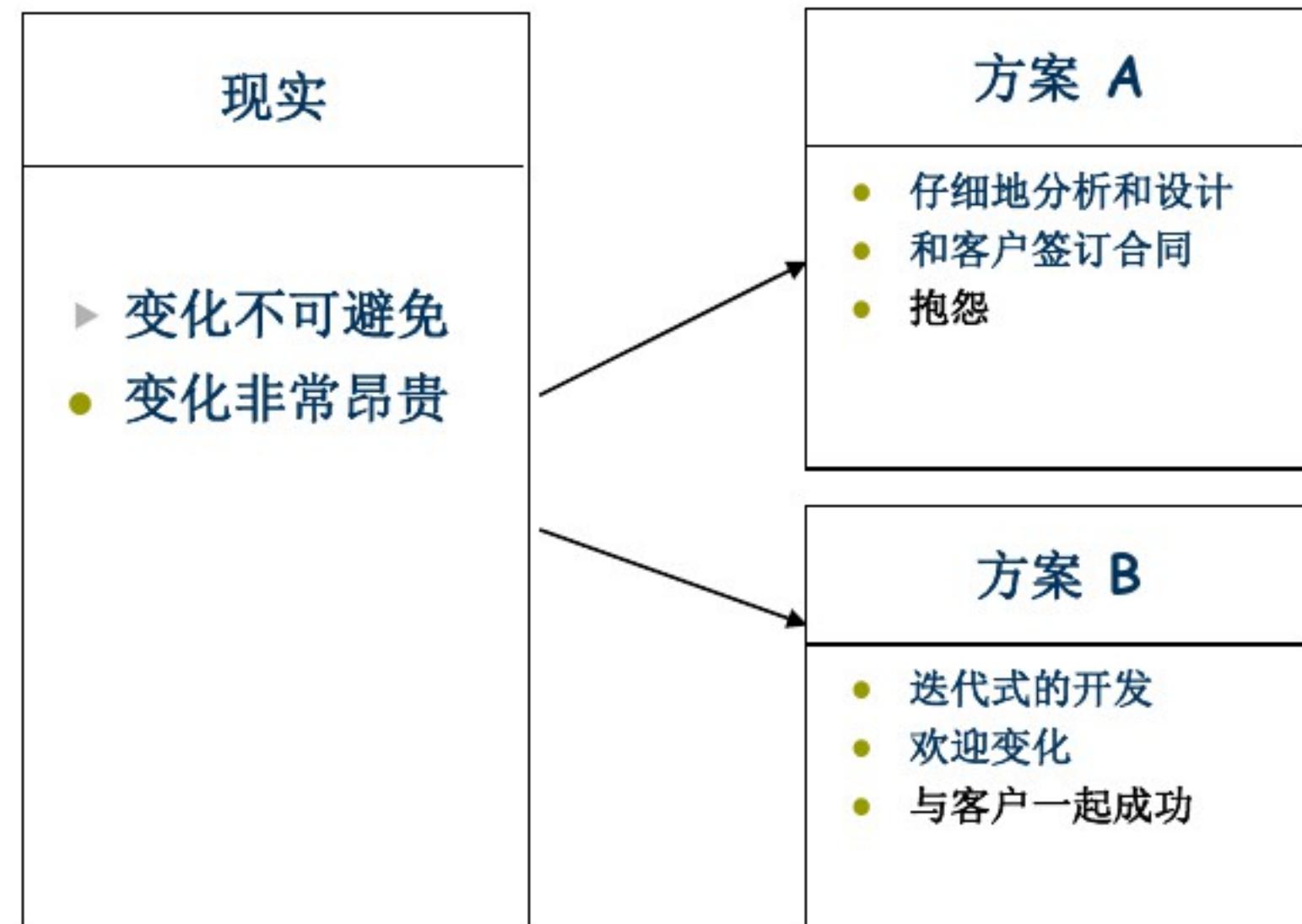
In late iterations, a significant change in requirements is rare, but can occur. Such late changes may give an organization a competitive business advantage.



one iteration of design,
implement, integrate, and test

After a series of structured, build-feedback-adapt cycles, the system will be stable.

拥抱变化



拥抱变化

- 仅仅有态度并不够：软件并不是想大多数人的直觉那样容易变化的。
- 迭代式的开发不比瀑布式开发容易。
- 我们应该构造能不断演化的软件系统。

迭代式开发的优势

- 能够较早地对付风险高的内容。
- 能够让人明确地看到进展，给客户信心，给开发队伍成就感。
- 能够较早获得反馈，鼓励用户参与开发，使系统能够更接近用户需求。
- 控制复杂性。

统一过程：Unified Process(UP)

- UP是迭代过程的一种。
提出人：**Ivar Jacobson**
- UP提供了如何实施OOA/D(和如何介绍OOA/D)的示范结构。这也形成了本书的结构。
- UP具有灵活性，可以应用于敏捷（轻量级）方法。

UP的阶段

□ UP项目将其工作和迭代组织为4个主要的阶段：

■ 初始(**Inception**)—

- 大体上的构想，业务用例，范围和初步的估计。

■ 细化(**Elaboration**)—

- 进一步细化的构想，以迭代的方式实现风险较高的核心架构，识别出大部分需求和范围，作更为准确地估计。

■ 构造(**Construction**)—

- 以迭代的方式实现剩下的低风险，易实现的部分，为发布做好准备。

■ 移交(**Transition**)—

- beta 测试，部署

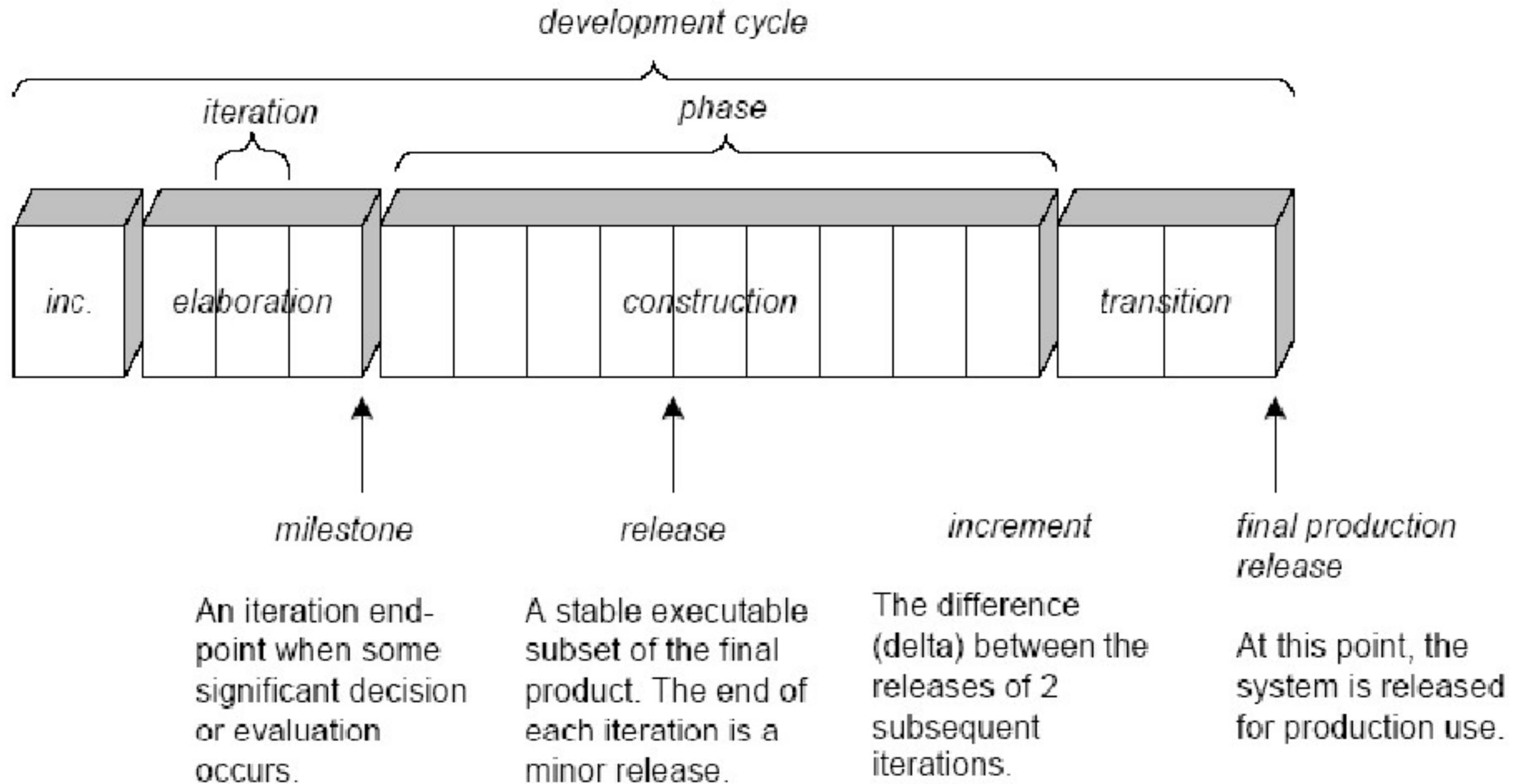


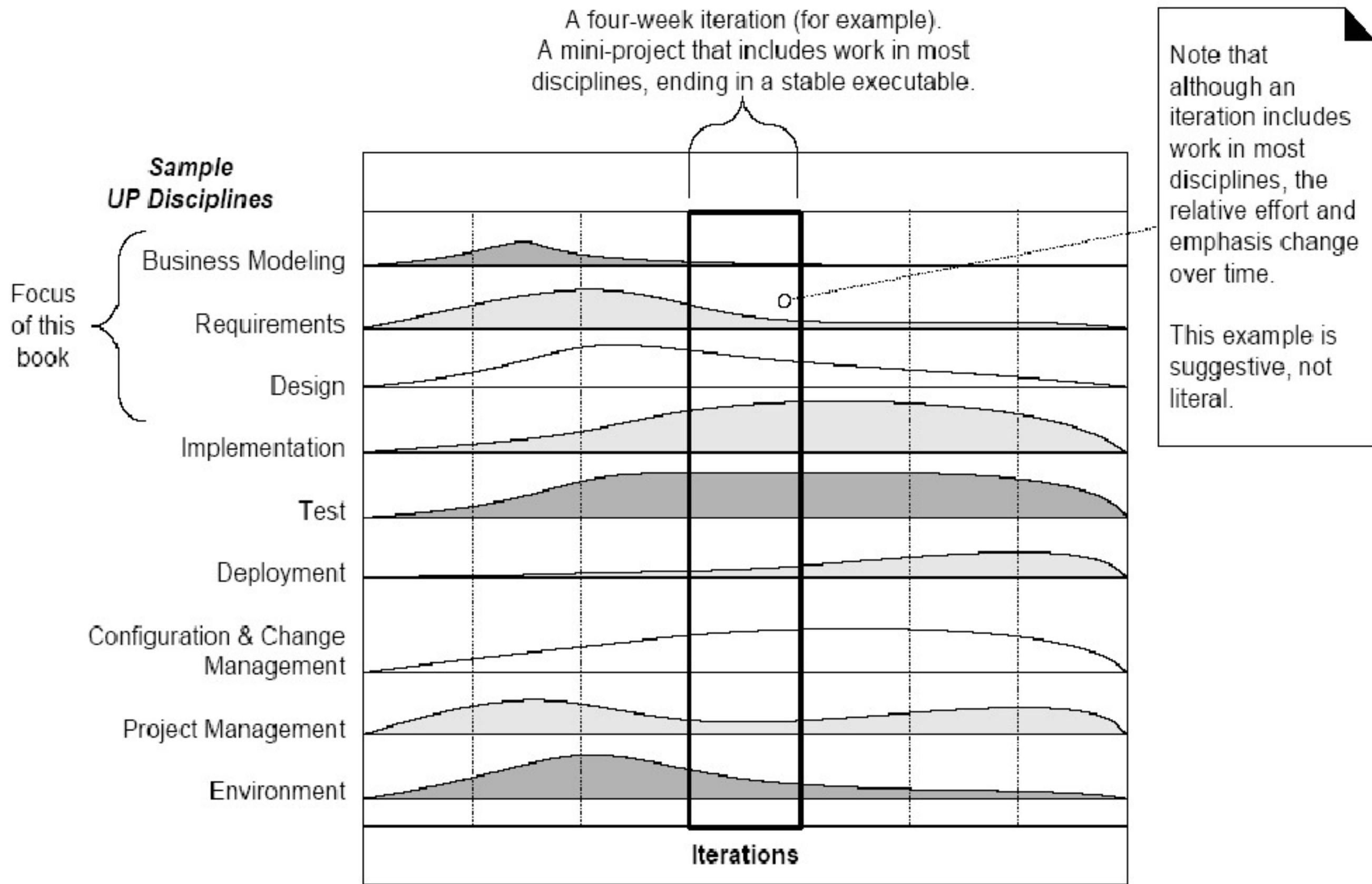
Figure 2.3 Schedule-oriented terms in the UP.

UP 科目(Disciplines)

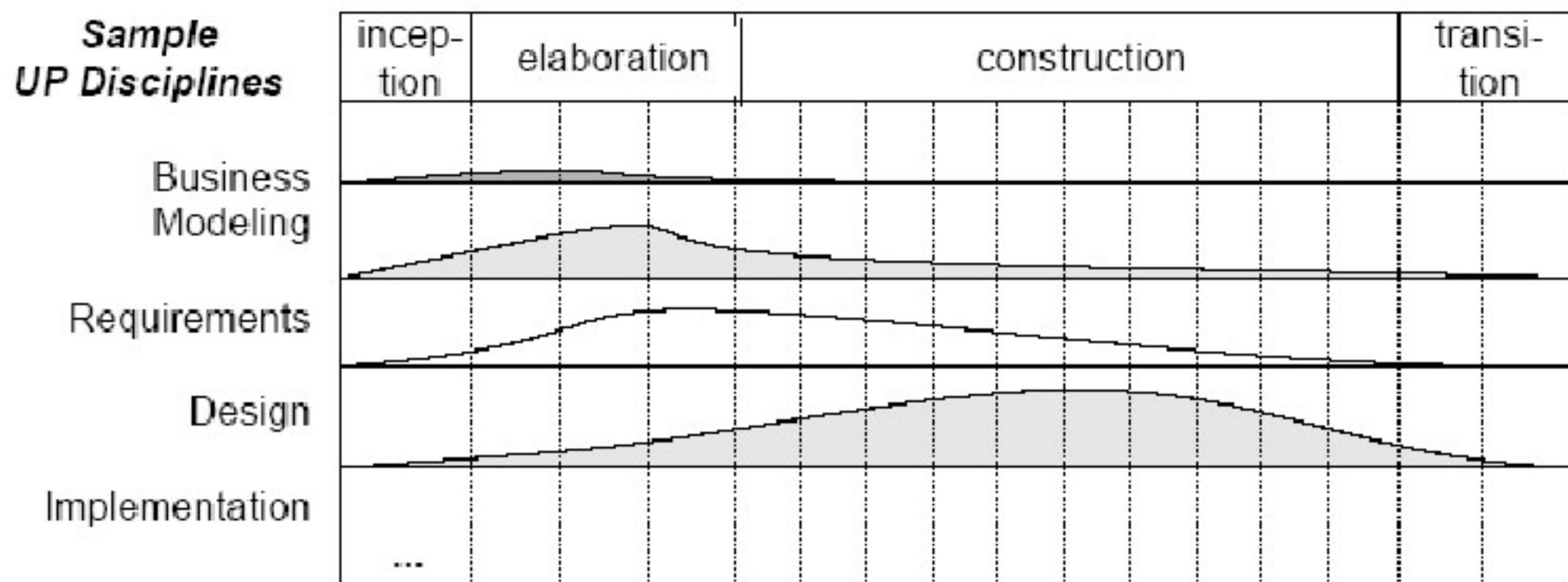
□ UP中定义了下列的科目：

- 业务建模(Business Modeling)
- 需求(Requirements)
- 设计(Design)
- 其他(实现/测试/部署....)

科目和迭代 (Disciplines and Iterations)



科目和阶段(Disciplines and Phases)



The relative effort in disciplines shifts across the phases.

This example is suggestive, not literal.

判断你是否理解迭代开发或UP

- 你是否认为
 - 初始 = 需求
 - 细化 = 设计
 - 构造 = 实现
- 你是否认为制作**UML**图的设计过程是用来精确地定义系统，而开发和编码只不过是将他们机械地变换为源程序的过程

根据UP的科目和阶段设计的课程结构

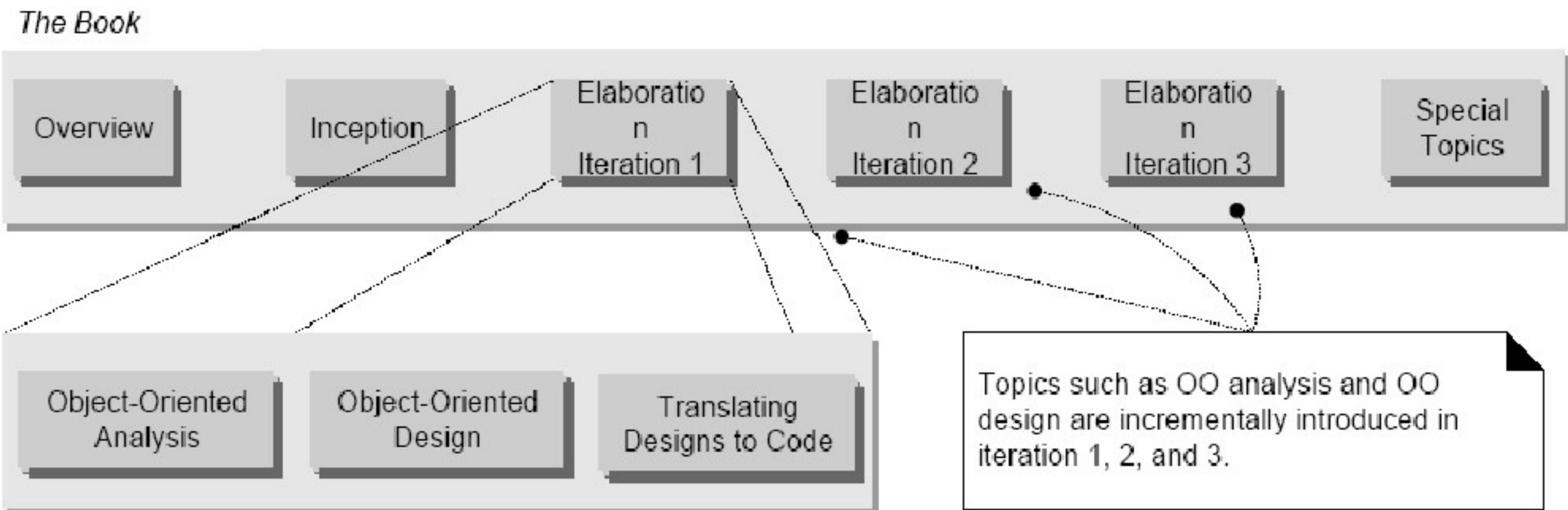


Figure 2.6 Book organization is related to the UP phases and iterations.

敏捷宣言

- | | |
|---------------------------------------|---------------------------------------|
| □ 个体和交流(Individuals and interactions) | □ 过程和工具(processes and tools) |
| □ 工作的软件(Working software) | □ 完善的文档(comprehensive documentation) |
| □ 与客户协作(Customer collaboration) | □ 合同谈判(contract negotiation) |
| □ 积极响应变更(Responding to change) | □ 严格履行计划(following a plan) |

敏捷原则

- 通过早期和持续交付有价值的软件来满足客户
- 欢迎变更需求，即使在开发的后期提出。敏捷过程为客户的竞争优势而控制变更。
- 以两周到两月为周期，频繁地交付可运行的软件。
- 在整个项目的过程中，每一天开发人员都要和来自客户的业务人员合作。
- 依靠有干劲的个体推动项目的开发，为他们提供所需的开发环境、支持和信任。
- 在开发团队中获开发团队间传递信息的最为有效和高效的方法是面对面的交流。
- 衡量进度的重要尺度是可运行的软件。
- 敏捷过程提倡持续开发和集成。
- 发起人、开发者和用户应该步调一致。
- 关注技术和设计技能的提高。
- 简洁，这门减少工作量的艺术至关重要。
- 团队要定期反省如何使工作更有效，然后相应地调整行为。

敏捷的UP

□ 敏捷的 UP:

- 从标准的UP活动中选取了一小部分活动和成果物，是UP的一个简集。
- 敏捷建模：建模的主要目的是为理解，而非文档。
- 不需要一个对于项目整体的详细计划。
- 测试驱动
- 重构
- 持续集成