中南大学
CENTRAL SOUTH UNIVERSITY

软件工程

Software Engineering

龙 军

jlong@csu.edu.cn 18673197878

计算机学院 | 大数据研究院

第二章 软件过程

大 纲

01-软件过程概述

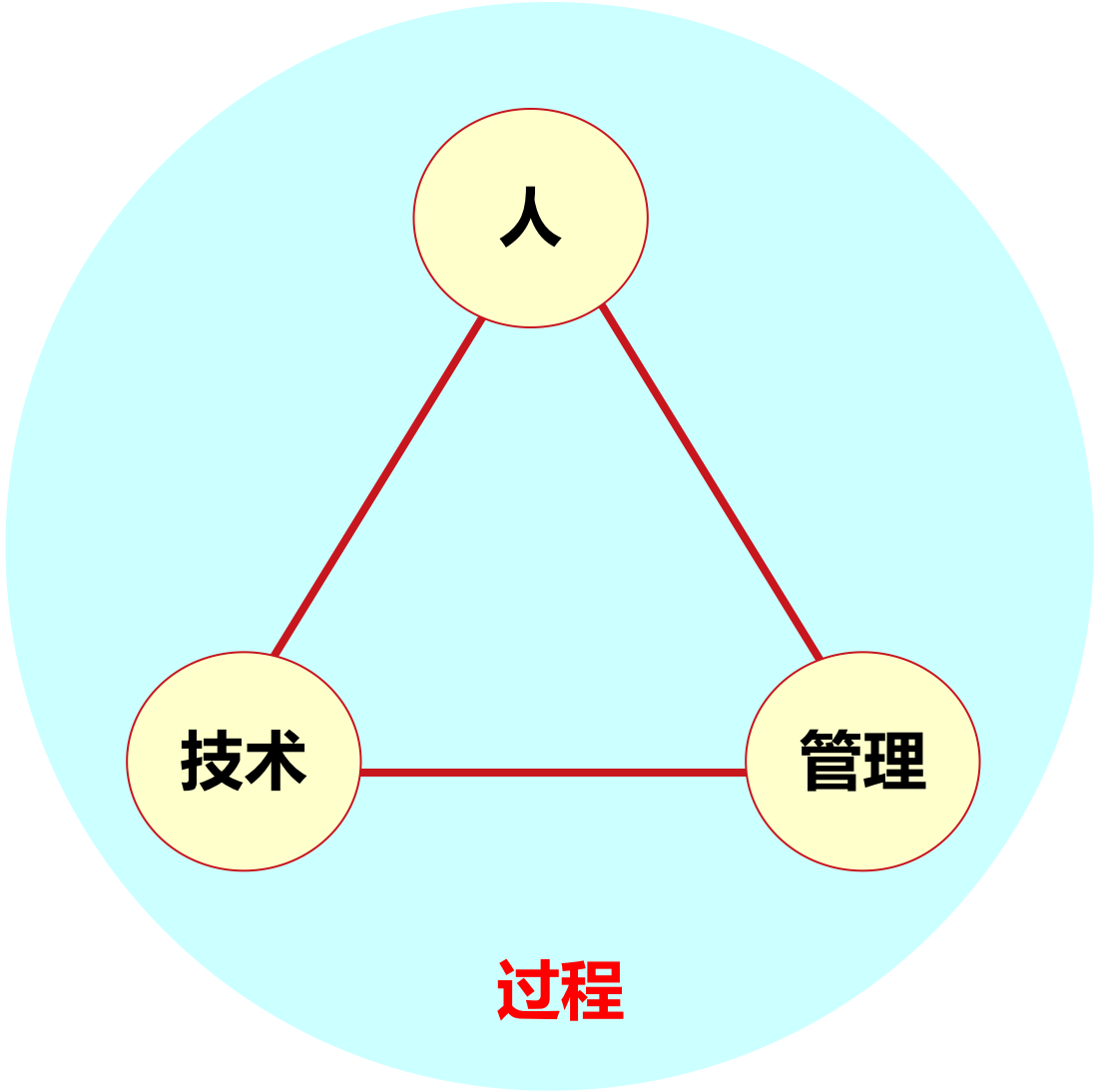
02-软件生命周期模型

03-统一软件过程 RUP

04-敏捷过程

@第2章.教材

软件工程的金三角



人

完成软件开发的主体

技术

提供了建造软件在技术上需要“如何做”的方法

管理

提供了质量管理、成本管理、时间管理、范围管理等知识和技能

过程

这是将人、技术、管理结合在一起的凝聚力

软件过程的杠杆作用点

- 每个人都体会到主动积极的优质劳动力的重要性，但是...
- ...如果不理解过程，或者过程不是在“最佳实践”下运行，即使我们的技术精英也无法使工作达到最佳的状态
- 过程是产品成本、进度和质量的主要决定因素

软件过程定义

软件过程：为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

软件过程（ISO9000）：使用资源将输入转化为输出的活动所构成的系统。

输入：如软件需求

输出：如软件产品

- Defines Who is doing What, When to do it, and How to reach a certain goal.

软件过程的组成

- 软件过程，也称为软件生存周期过程，是指软件生存周期中的一系列相关过程，其中过程就是活动的集合，活动是任务的集合，任务要起到把输入加工成输出的作用。
- 活动的执行可以是顺序的、迭代的（重复的）、并行的、嵌套的，或者是有条件地引发的。

第二章 软件过程

大 纲

01-软件过程概述

02-软件生命周期模型

03-统一软件过程 RUP

04-敏捷过程

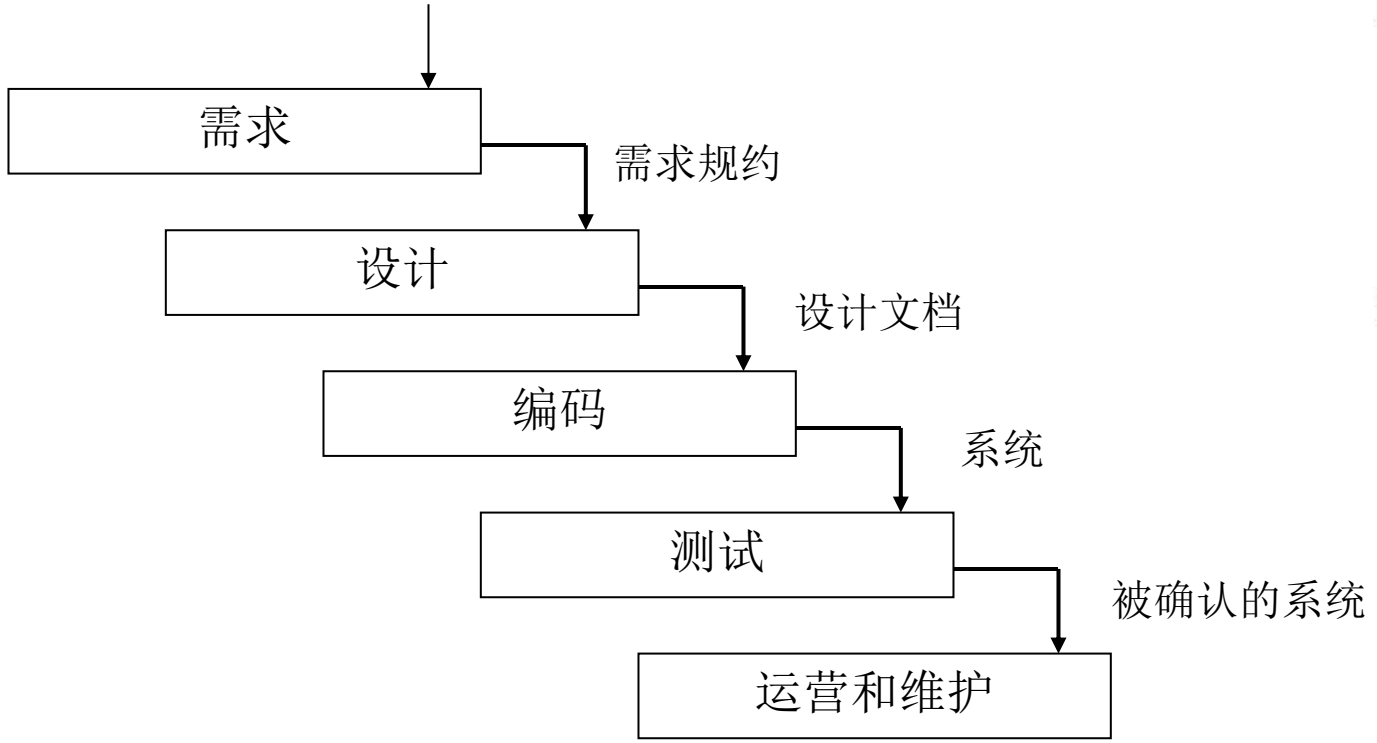
@第2章.教材

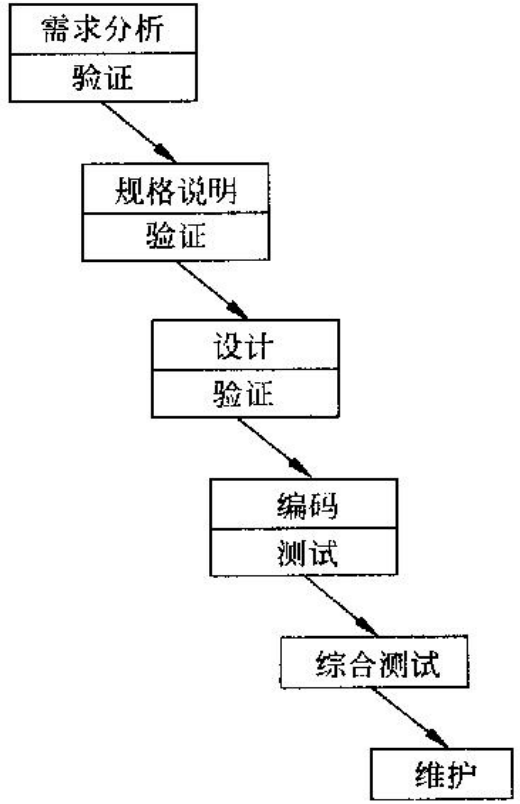
软件生存周期模型

- 又称软件开发模型，是软件生命周期的一个框架，规定了软件开发、运作和维护等所需的过程、活动和任务。
- 软件生存周期模型分类：
  - 瀑布模型 Waterfall Model
  - 增量模型 Incremental Model
  - 演化模型 Evolutionary Model

瀑布(Waterfall)模型

- 最早的软件开发模型
- 1970年W. Royce提出
- 又称为线性顺序模型





传统的瀑布模型

瀑布模型特点

本质要求：
1. 阶段间具有顺序性和依赖性
2. 推迟实现的观点
3. 质量保证的观点

- 特点
  - 强调阶段的划分及其顺序性
  - 强调各阶段工作及其文档的完备性
  - 每个阶段结束之前，都从技术和管理两个角度进行严格的审查
  - 是一种严格线性的、按阶段顺序的、逐步细化的开发模式
- 适用时机
  - 所有功能、性能等要求能一次理解和描述时
  - 所有的系统功能一次交付时
  - 必须同时淘汰全部老系统时

优点：
采用规范的方法；严格规定每个阶段提交的文档；要求每个阶段交出的产品必须经过验证。

瀑布模型的价值和风险

• 瀑布模型的价值

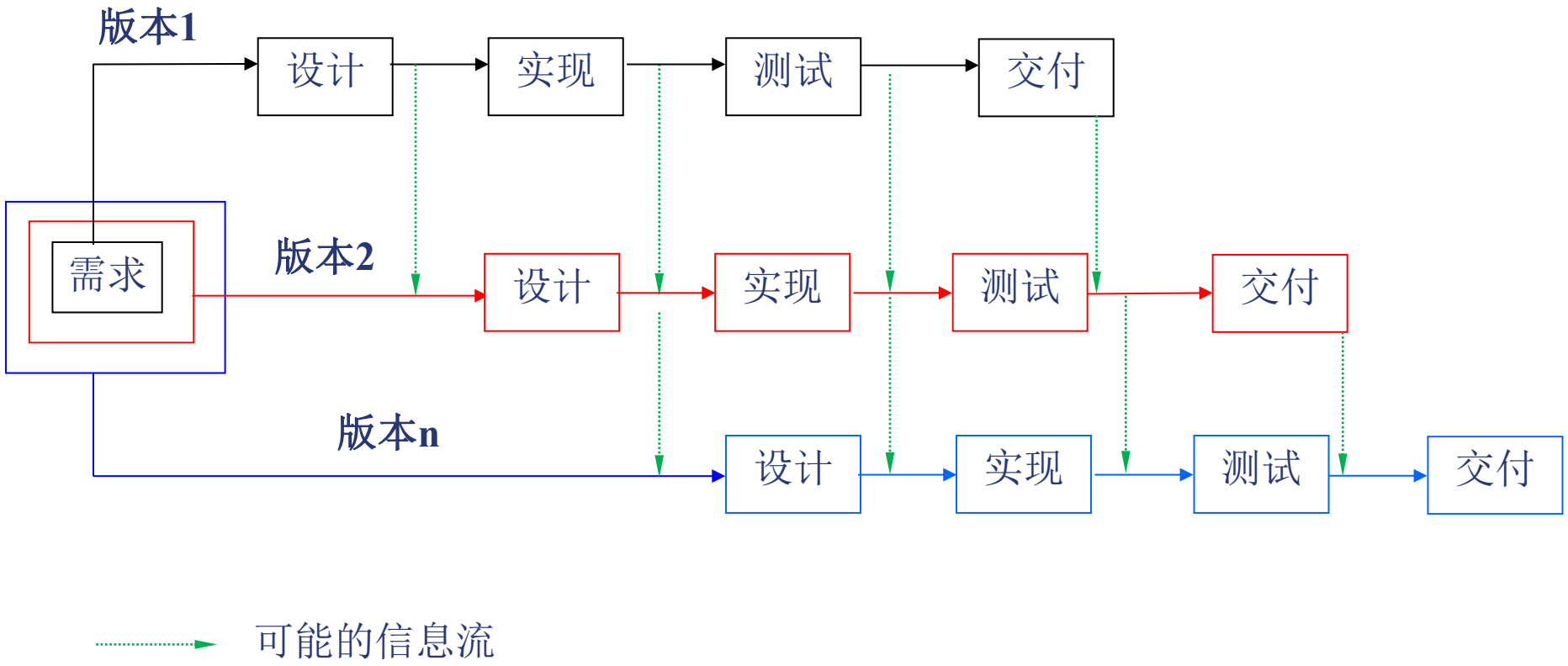
- 结构简单明了；历史较长、应用面广泛、为广大软件工作者所熟悉；已有与之配套的一组十分成熟的开发方法和丰富的支撑工具。
- 一种较为有效的管理模式：订计划、成本预算、组织开发人员,阶段评审,文档管理,从而对软件质量有一定的保证。

• 瀑布模型的风险

- 获得完善的需求规约是非常困难的；
- 难以适应快速变化需求；
- 系统太大时,难以一次做完；
- 反馈信息慢；
- 极可能引起开发后期的大量返工，如返工到需求、设计等早期活动。

增量（Incremental）模型

- 需求和架构确定后，增量式进行开发，构造一系列可执行的版本(Version by Version)



增量模型的风险和适用时机

• 增量模型的风险

- 需求未被很好地理解
- 一次要求所有功能
- 需求迅速发生变化
- 事先打算采用的技术迅速发生变化
- 长时期内仅有有限的资源（人员/资金）

• 增量模型的适用时机

- 需要早期获得所有需求；
- 根据需求建立稳定的软件架构；
- 中间产品可以提供使用；
- 系统被自然地分割成增量；
- 工作人员/资金可以逐步增加。

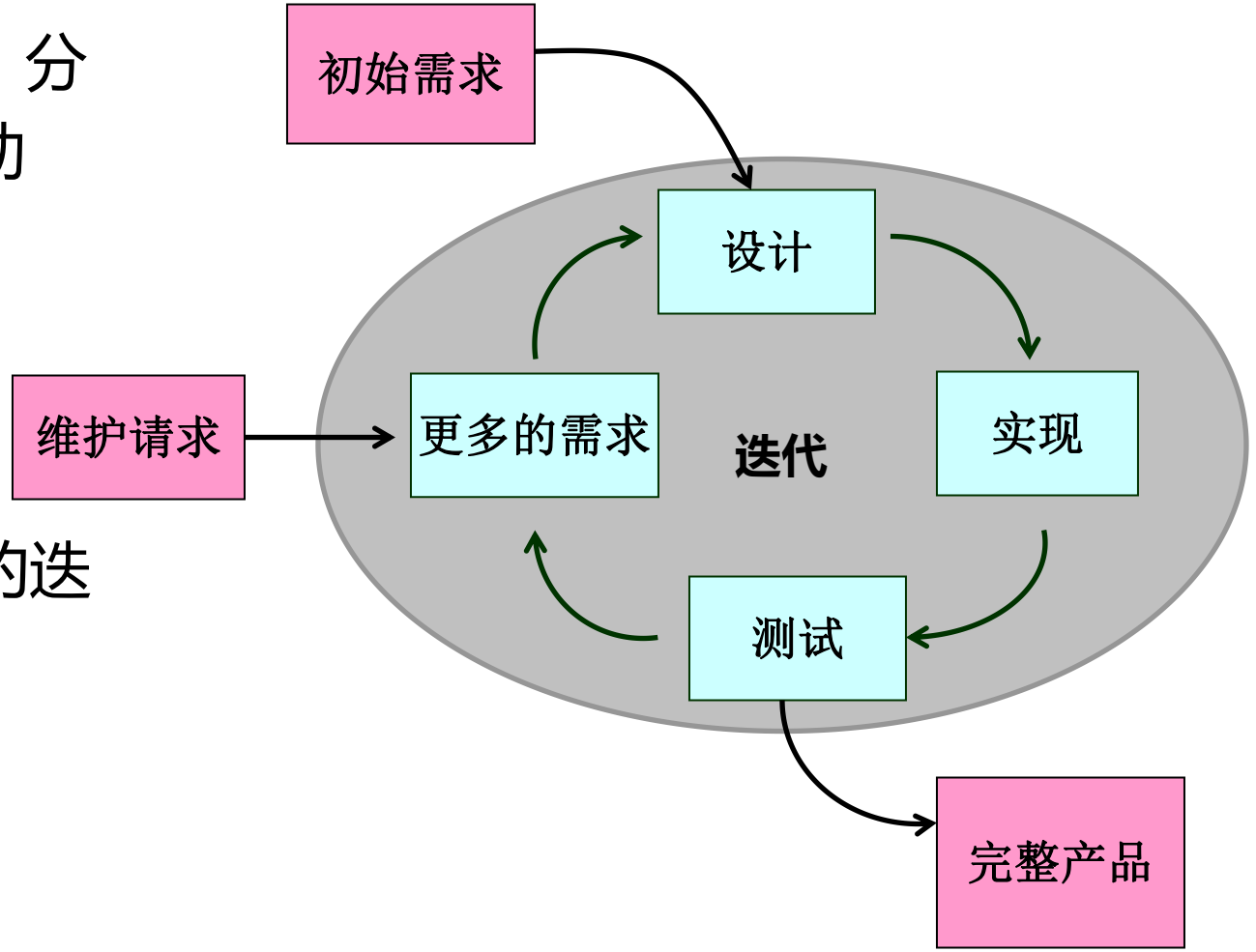
演化(Evolutionary)模型

- 现状：
  - 软件需求在软件开发过程中常常发生改变，想要一次迭代就开发出最终产品是不可能的
  - 紧迫的市场期限使得难以一下子完成一个完善的软件产品
- 解决方案：演化模型
  - 只要核心需求能够被很好地理解，就可以进行渐进式开发，其余需求可以在后续的迭代中进一步定义和实现。这种过程模型称为演化模型，它能很好地适应随时间演化的产品的开发。
- 特点：
  - 迭代的开发方法，渐进地开发各个可执行版本，逐步完善软件产品。每个版本在开发时，开发过程中的活动和任务顺序地或部分重叠平行地被采用。
  - 与增量模型的区别是：需求在开发早期不能被完全了解和确定，在一部分被定义后开发就开始了，然后在每个相继的版本中逐步完善。

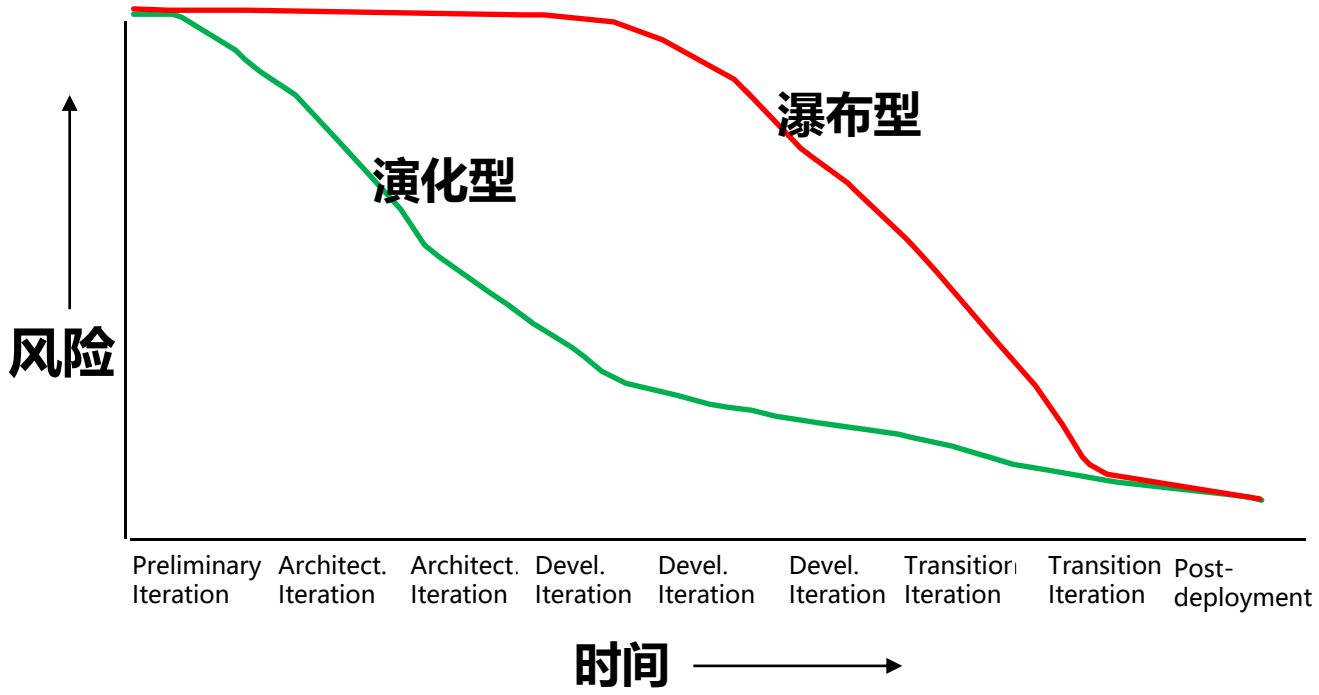
迭代开发原则

- 每次迭代都应产生一个可执行的软件版本。
  - 每次迭代都包括计划、建模、需求、分析和设计、实现、测试、评估等活动
- 要求有计划的迭代。
  - 通常由3~9个迭代组成
  - 风险驱动：项目的风险越高，迭代就越多；风险越高的工作，越在早期的迭代中执行
  - 迭代可以并行、重叠、串行
  - 迭代内部的活动可以交叉并行

示例: 迭代顺序执行的演化模型



演化模型价值



不断探索

持续改进

拥抱变化

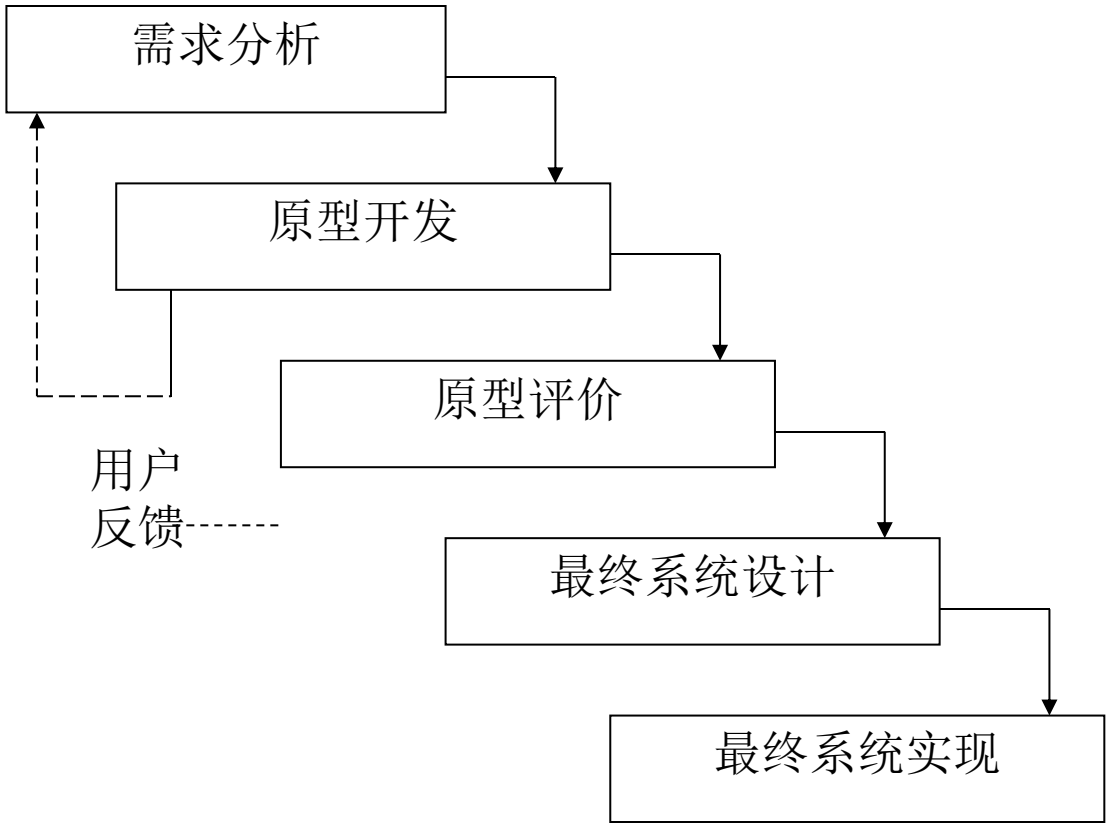
降低风险

演化模型是目前采用最广泛的模型

演化模型已成为主流

- 现代软件过程都采用演化模型
  - 统一软件过程RUP
  - 敏捷过程 （SCRUM、XP等）
  - 净室（Cleanroom）软件过程
- 演化模型的“子类”
  - 原型 Prototyping
  - 螺旋模型 Spiral Model

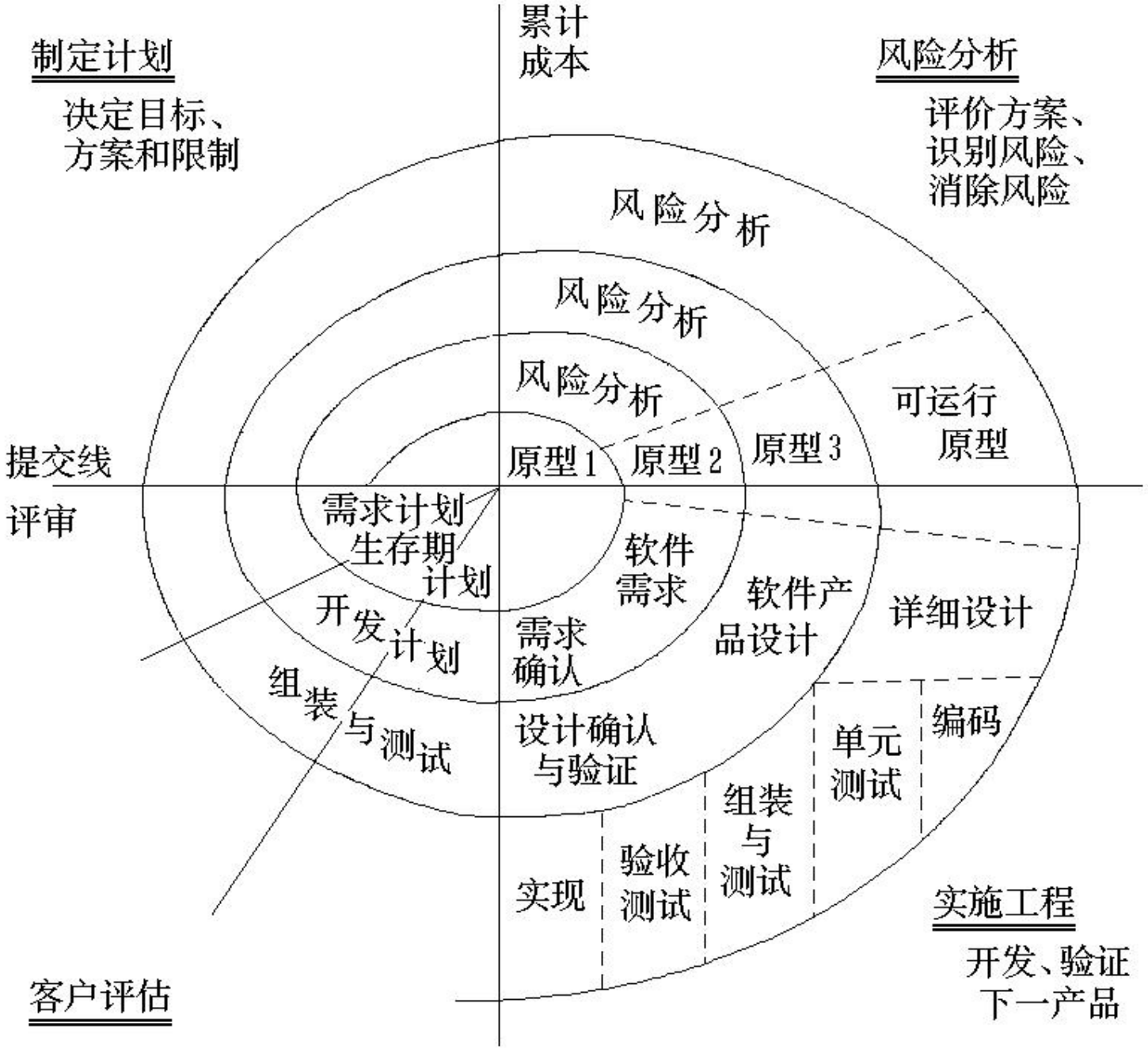
快速原型模型 Rapid Prototyping Model



- 特点
  - 快速开发工具
  - 循环
  - 低成本
- 种类
  - 渐进型
  - 抛弃型

一种迭代次数为2的演化模型

螺旋模型
Spiral Model



一种基于风险的演化模型

第二章 软件过程

大 纲

01-软件过程概述

02-软件生命周期模型

03-统一软件过程 RUP

04-敏捷过程

@第2章.教材

统一软件过程 RUP

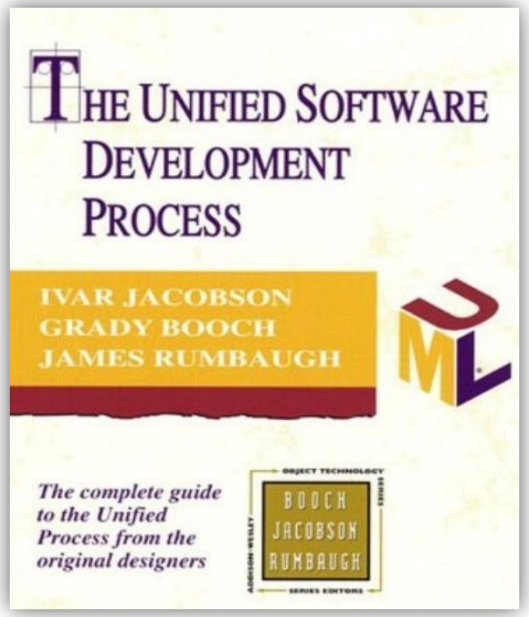
Ivar Jacobson

RUP是迄今为止所能见到的最好的软件过程模型。

它越是看似优秀，就越是值得怀疑。

RUP采用增量迭代开发过程，使用“统一建模语”(UML)作为标准建模语言，并集成了面向对象 (OO) 研究成果。

Ivar Jacobson 是面向对象方法学的奠基人，提出了模块架构、用例、现代业务工程、瑞理统一过程等业界主流方法和技术，先致力于构建软件工程学科理论体系（SEMAT）。
信息网页：http://www.ivarjacobson.com/cn/



统一软件过程 RUP

针对不同核心工作流，多次迭代四阶段。

RUP的四大阶段

① **构思/初始(Inception)：**

- 将最初想法形成正式产品描述，包括定义主要需求和设计初始架构。
- 一般采用UML的用例图来描述软件的主要功能。

② **细化(Elaboration)：**

- 细化软件描述，包括添补或细分用例，并设计软件的各类架构视图。

③ **构造(Construction)：**

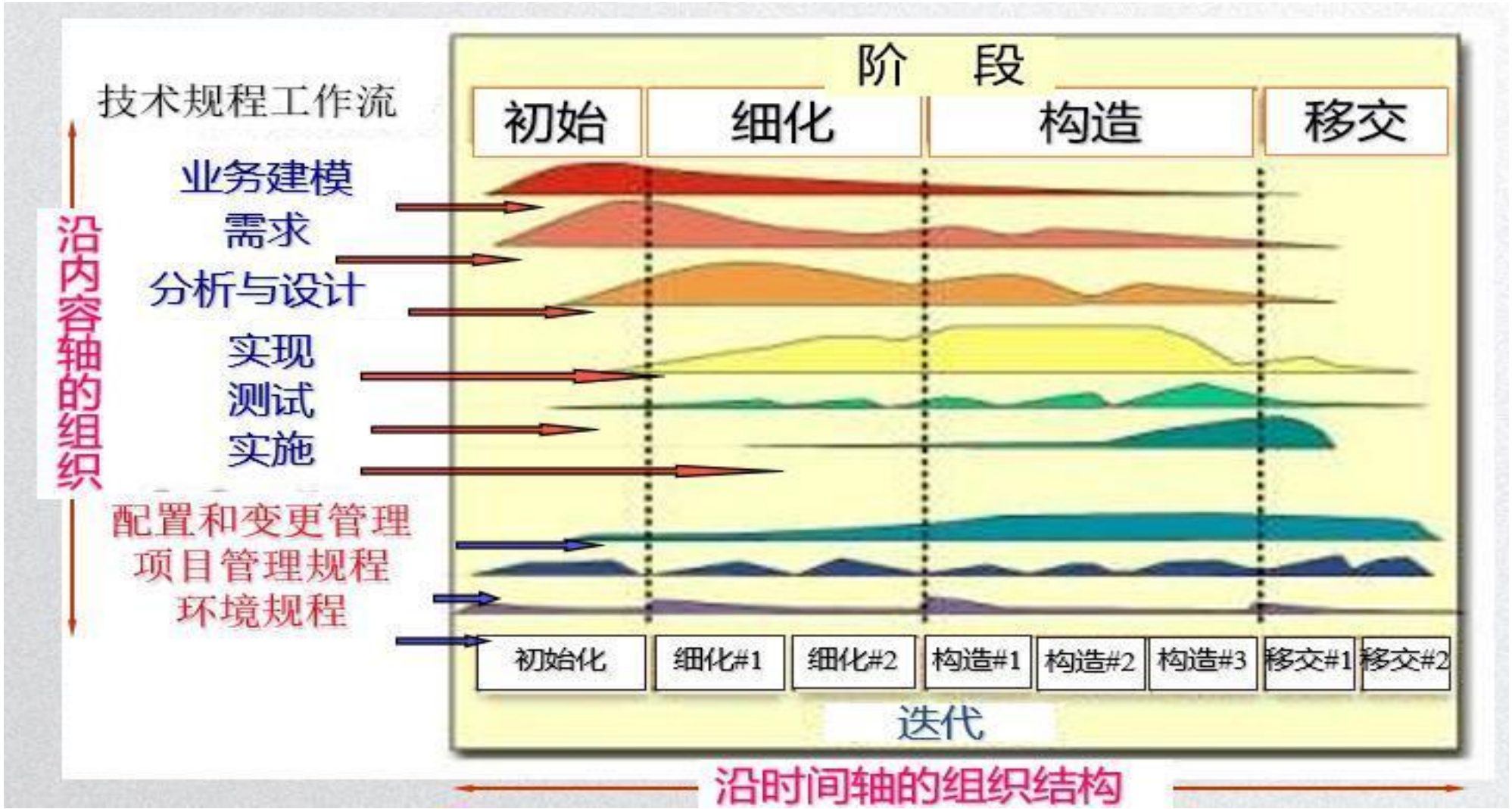
- 依据各种架构视图，实现软件的所有需求。
- 这期间可能还包括对架构的细微修改，同时还包括软件测试。

④ **移交(Transition)：**

- 完成各核心工作流的收尾任务，并准备用于软件交付的各类文档。
- 同时邀请用户试用产品以发现缺陷，另外还涉及用户培训等。

统一软件过程 RUP

※ 驼峰图：RUP的核心工作流及工作量



RUP是一个风险驱动的、基于UML和构件式架构的演化开发过程

RUP软件开发实践

① **用例(Use Case)驱动**
- 以用例和用例图的形式描述软件的主要功能

② **以架构为中心**
- 定义架构的不同视角，并据此固化(即映射)软件需求

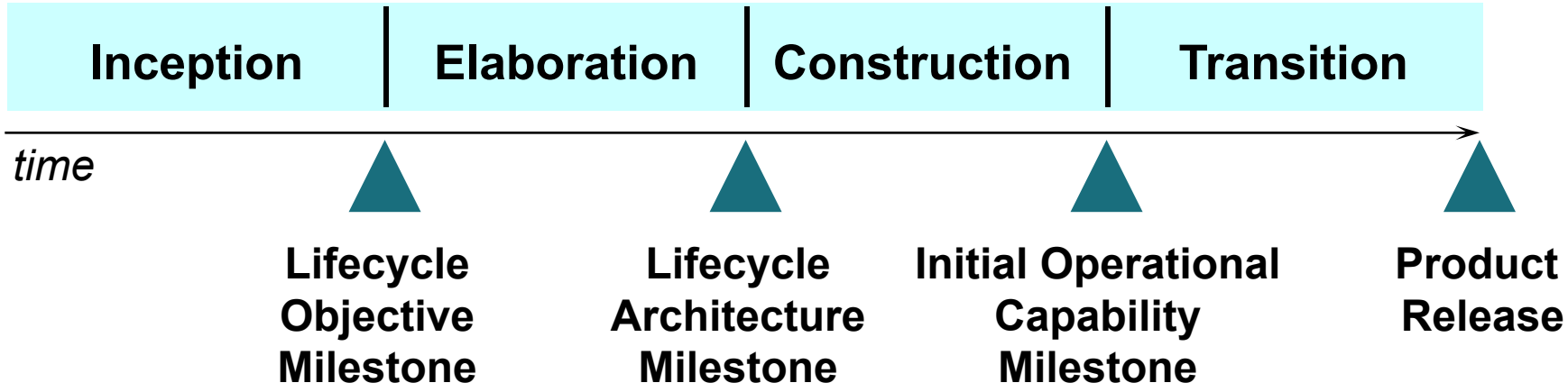
③ **管理需求**
- 包括需求获取、组织、编档和追踪，以及商业业务捕获

④ **可视化软件建模**
- 使用UML图从各种建模角度，描述软件的结构、功能和行为

⑤ **验证软件质量**
- 依据需求，审查和评估软件的质量

⑥ **控制软件变更**
- 针对所有制品，依据正常的变更控制过程管理每一次变更

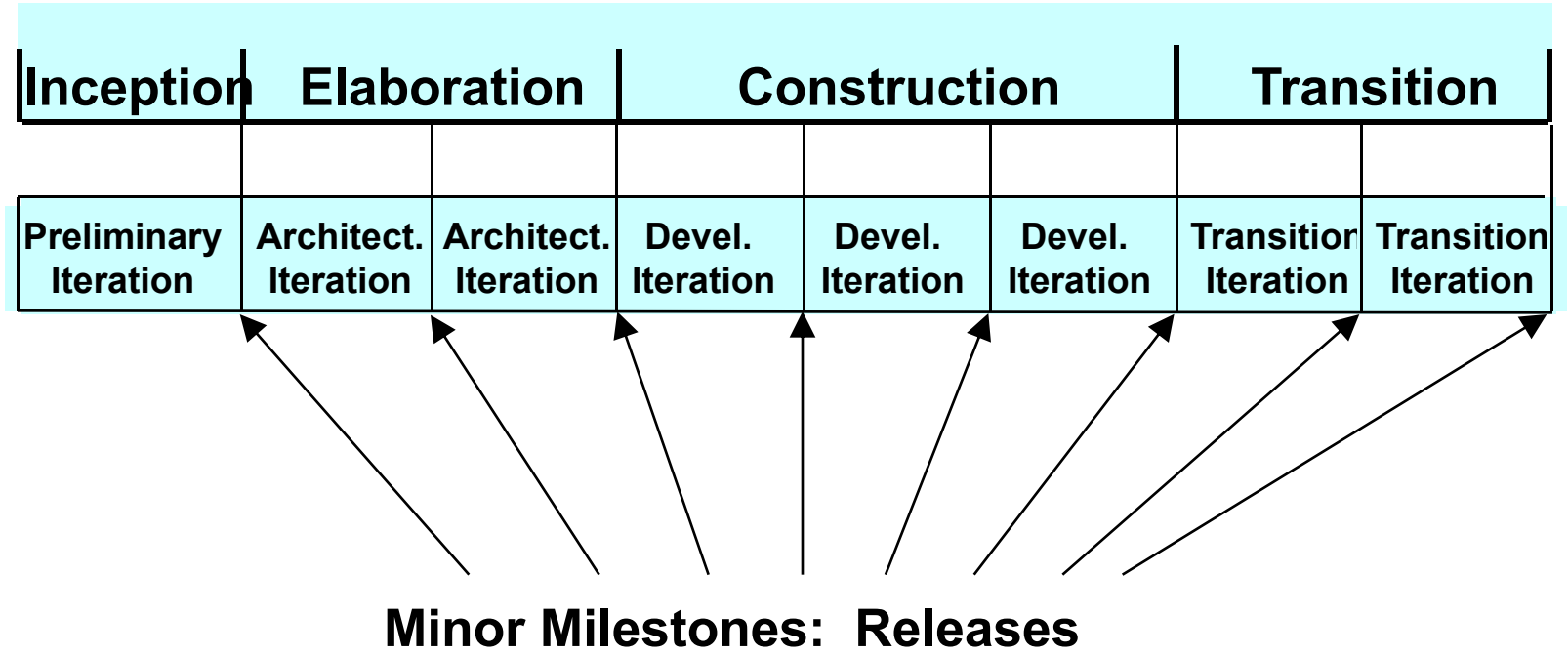
RUP的四个阶段



- Inception - Define the scope of project
- Elaboration - Plan project, specify features, baseline architecture
- Construction - Build the product
- Transition - Transition the product into end user community

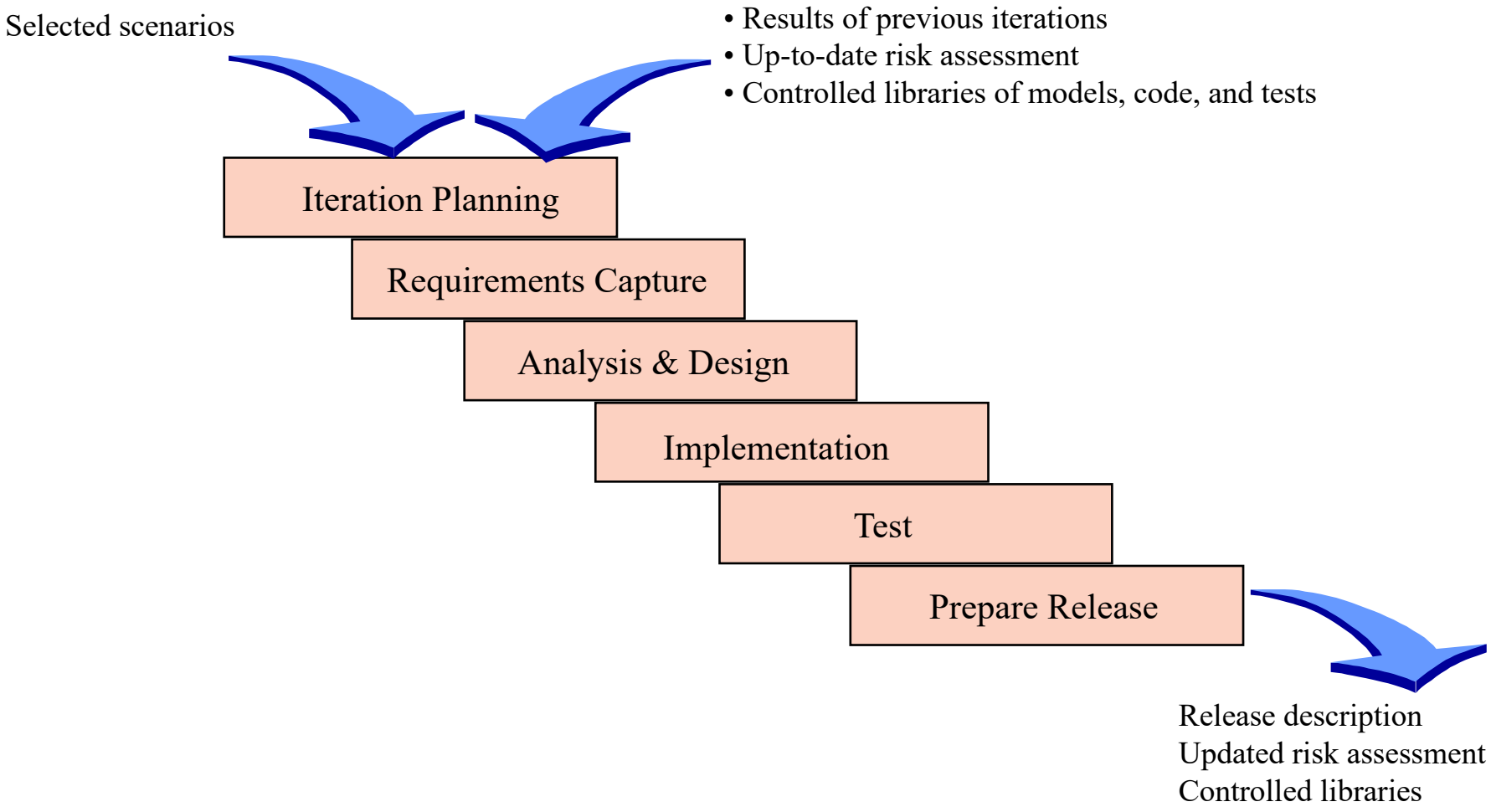
每个阶段结束是一个大的里程碑

阶段和迭代



An iteration is a distinct sequence of activities with an established plan and evaluation criteria, resulting in an executable release (internal or external)

一个迭代周期：一个小的瀑布模型



第二章 软件过程

大 纲

01-软件过程概述

02-软件生命周期模型

03-统一软件过程 RUP

04-敏捷过程

@第2章.教材

敏捷(Agile)过程

※ “敏捷” (Agile)，即快速响应变更的能力

※ 敏捷开发，即在开发软件的过程中积极响应随时可能出现的变更请求。

※ 敏捷方法，即敏捷开发模型，是一类能够敏捷地应对快速需求变更的软件开发模型，属于增量—迭代模型家族。

敏捷方法被视为
当前最有效的
软件开发方法。

它越是看似优秀，就越是值得怀疑。

敏捷宣言

※ 2001年2月11日，在美国犹他州雪鸟滑雪圣地的一间小屋中，由Kent Beck等领头的17位软件工程专家成立了“敏捷联盟”(Agile Alliance)。

※ 与会专家经过为期两天的讨论之后共同发布了“敏捷软件开发宣言”。

※ 这标志着敏捷方法的正式诞生。

敏捷联盟的官网：http://www.agilealliance.org/

敏捷宣言—全文

从亲自和协助他人进行软体开发的实践中，我们发现了更好的软体开发方法[即敏捷方法]。 基于已有努力，我们已经为这个方法树立了如下四项价值观：

个体和交互 **胜过** **流程和工具**

可工作的软件 **胜过** **面面俱到的文档**

客户协作 **胜过** **合同谈判**

响应变更 **胜过** **遵循计划**

虽然上述右边项也具有实践价值，但我们认为左边项具有更大的价值。

反之，就是瀑布宣言

敏捷过程的适用范围

- Martin Fowler认为：**新方法不是到处可适用的**

- 适合采用敏捷过程的情况：
  - 需求不确定、易挥发
  - 有责任感和积极向上的开发人员
  - 用户容易沟通并能参与
  - 小于10个人的项目团队

敏捷十二法则

- ※ 致力于通过不断地按时交付有价值的软件，以保持客户的满意度。
- ※ 不论何时都要欣然面对需求变更。
- ※ 每隔几周或一两月就须交付可工作的软件。
  - 周期宜短不宜长。
- ※ 业务和开发人员必须坚持每天协同工作。
- ※ 激励个体的斗志并以他们为项目核心，给予充分支持、信任和授权。
- ※ 倡导可持续开发
- ※ 可工作的软件是项目进度的首要度量标准。

敏捷十二法则

- ※ 坚持不懈地追求卓越技术和良好设计。
- ※ 以简单为本，最大限度地减少无用功。
- ※ 组建自组织团队。
  - 高质量的架构、需求和设计都源自此类团队。
- ※ 定期反思如何提升整体性能。
- ※ 首选面对面的交谈。
  - 这是团队内外最有效、
    也最高效的信息传递方式。

常用敏捷方法

- Scrum方法
- 极限编程(XP)

- 精益(Lean)开发
- 特征驱动型开发(FDD)
- 自适应软件过程(ASP)
- Crystal轻量级方法簇
- 动态系统开发方法(DSDM)
- …

Scrum - 敏捷的软件项目管理

- 1994年由Ken Schwaber和 Jeff Sutherland 提出
- Scrum一词来源于橄榄球运动，意为两队并列争球
- Scrum过程的核心：
  - 一个体育队加小队长,全体团队负责拿球向前冲
  - 团队成员能够独立地、集中地在创造性的环境下工作

Scrum的核心准则

Iterative Development 迭代开发

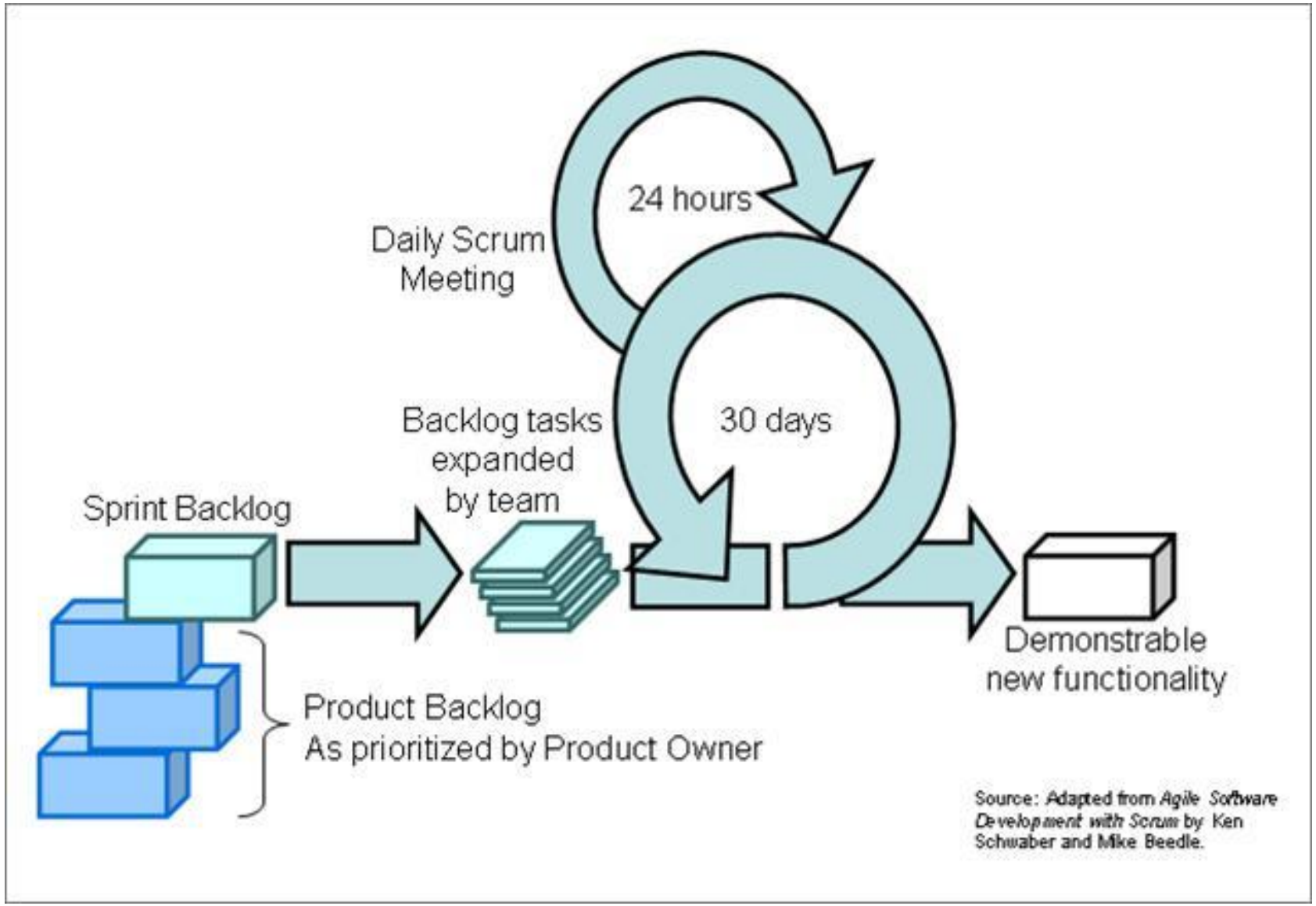
- The project deliverables are built over several iterative development cycles, each adding additional features, and each resulting in demonstratable results: working code, written documentation, viewable designs, etc.

可持续的、小批量、高质量、快速地发布

Team Empowerment 自我管理

- The project team is divided into self-managing multi-function units called Sprint Teams consisting of up to six or seven people. The team is empowered to use whatever development methods or tools they think best to prepare their deliverables

Scrum过程框架



Sprint

周期为30天的迭代

Backlog

待办事项表（功能和非功能需求清单）

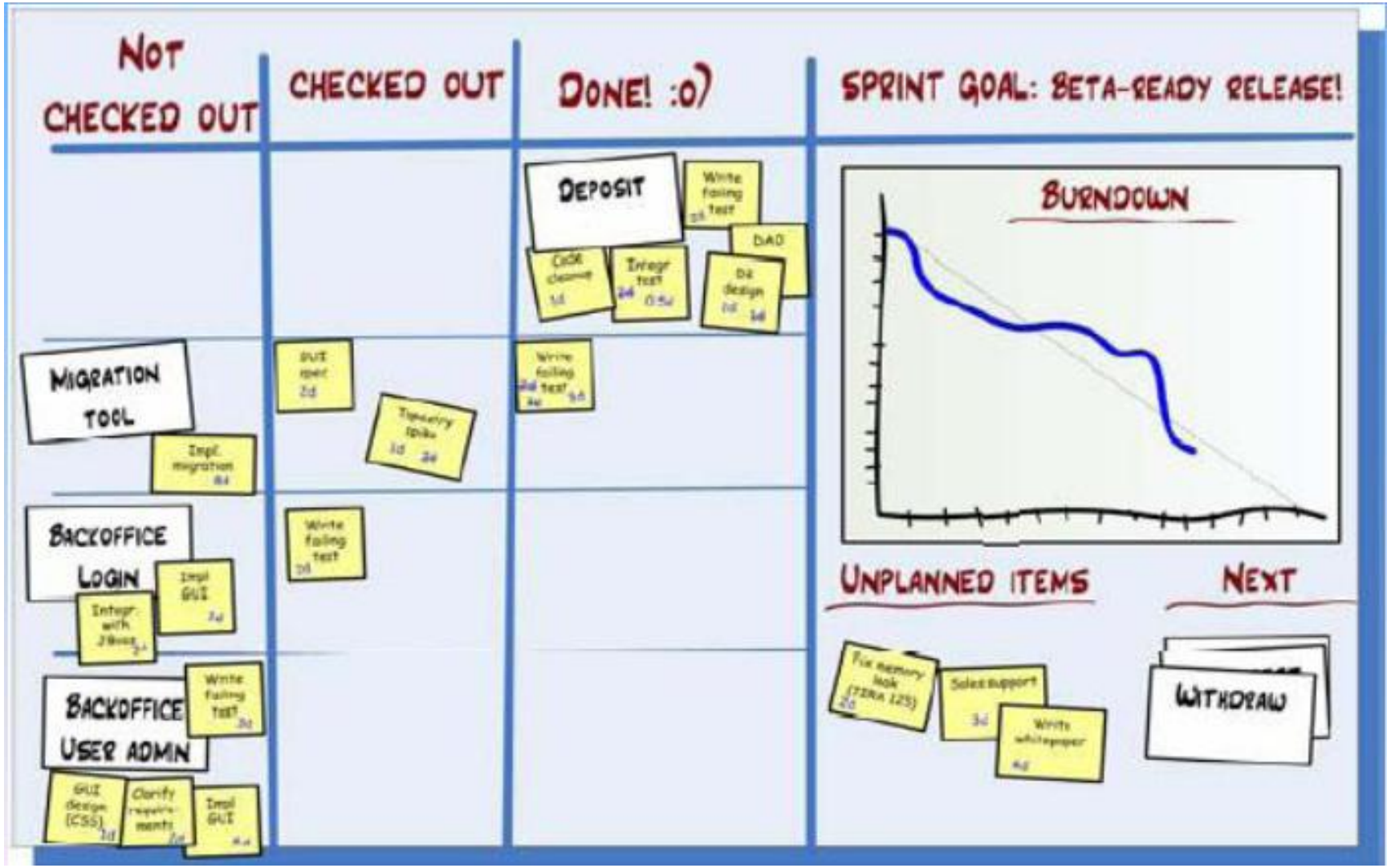
Daily Scrum

每日15分钟简会

Scrum Players

- **Scrum 项目组**
  - Product Owner
    - Adds items to product backlog list
    - Set priorities
  - Team
    - Determines sprint list（determine what can be done ）
    - Develops software
  - ScrumMaster （相当于传统的项目经理角色）
    - Responsible for Scrum process
    - Removes impediments
- Other interested parties
  - Stakeholder
    - Funded project, will use it, affected by it
    - Requests enhancements
  - IT Management
    - Manpower allocation
    - Budgets & Billing
  - Senior Business Management
    - Best use of corporate resources

Sprint 任务板

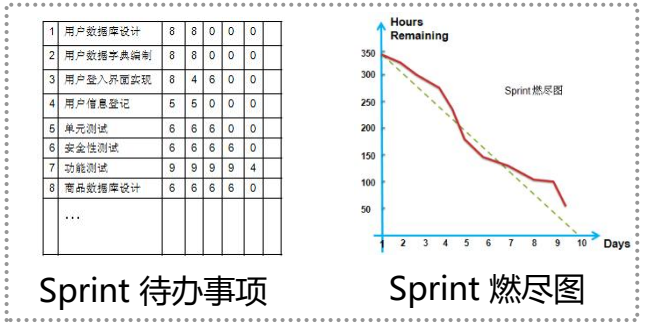

NOT CHECKED OUT
CHECKED OUT
DONE! :o)
SPRINT GOAL: BETA-READY RELEASE!
DEPOSIT
MIGRATION TOOL
BACKOFFICE LOGIN
BACKOFFICE USER ADMIN
BURNDOWN
UNPLANNED ITEMS
NEXT
WITHDRAW

Scrum过程示例

优先级	内容	大小
1	用户注册	5
2	登入商品	13
3	展示商品	5
4	搜索商品	3
5	在线交易	8
6	交易状态确认	2
7	特色商品推广	5
8	第三方支付	13
9	高级搜索功能	3
10	信用评价	5
11	返点活动支持	3
12	虚拟商品支持	8
13	物流跟踪	8
14	投诉处理	5

产品代办事项

发布燃尽图

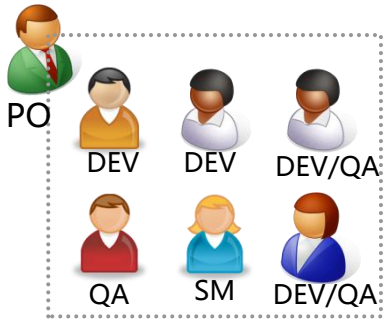


Sprint 待办事项　　Sprint 燃尽图

1. 产品开发过程
   1.1 高度透明
   1.2 不断反馈调整

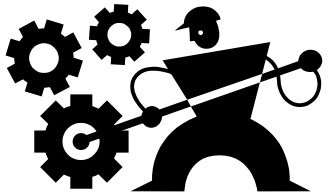
- Sprint 计划会议
- Sprint 评审
- Sprint 回顾
- 每日scrum 会议

项目启动 → Sprint 1 → Sprint 2 → Sprint .... → Sprint n



2. 团队
   2.1 多功能
   2.2 自组织

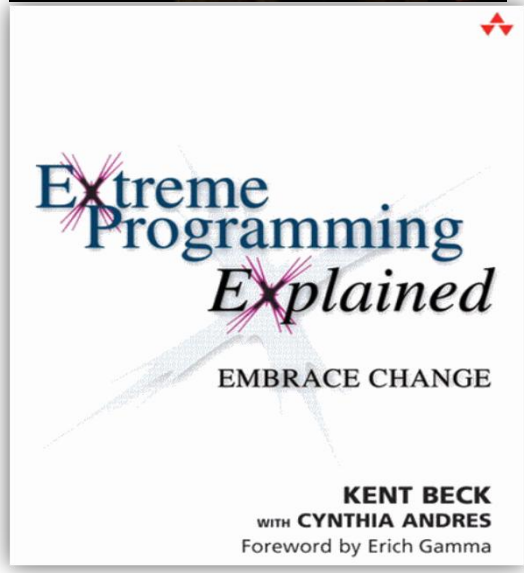
改善



3. 持续改进
   3.1 将改进嵌入开发过程
   3.2 不断暴露和解决问题

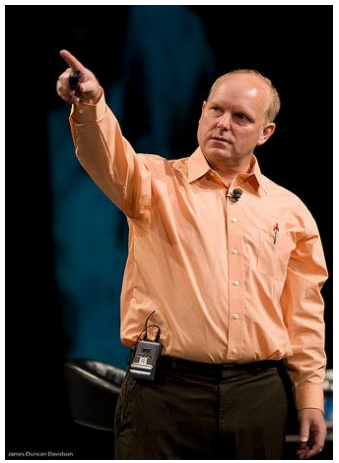
极限编程（XP）

- XP方法由业界专家Kent Beck于1998年创造
- 理念：将有价值的实践发挥到极致
- 是轻型增量迭代方法的典型代表

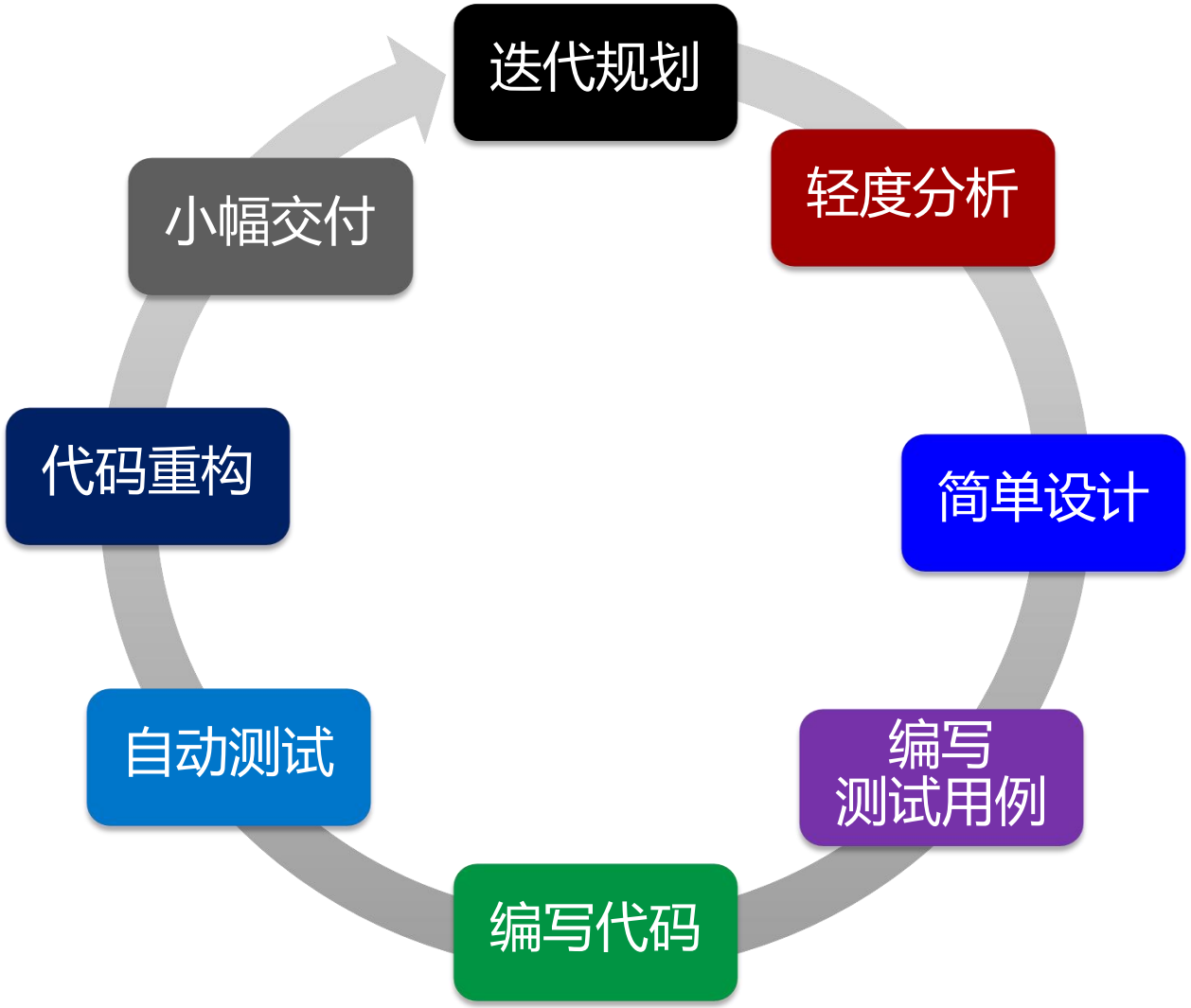


XP的五项价值观

推行XP的最大阻力来自于工程师内心的极度恐惧。



XP通用流程



XP十二实践

- **规划游戏**
  - 为下一次迭代制定计划，由一线工程师负责成本、时间和进度的估算
- **频繁交付小增量版本，及时获取客户/用户反馈**
- **隐喻(Metaphor)**
  - 隐喻如用不好，就会造成误读误解或过度简化软件逻辑
- **简单设计(即KISS法则)，不添加任何不必要的设计元素**
  - KISS法则不仅适用于软件设计，还适用于软件需求和编码，是一项普适的技术写作法则
- **测试驱动型开发(TDD)**
  - 依据需求编写测例，据此编写代码，并让测例顺利通过

XP十二实践

- **持续重构(Refactoring)**
  - 持续地重整代码的结构(但不改变代码的原有功能)
- **结对编程(Pair Programming)**
  - 由两个程序员共用一台电脑和一个键盘编写程序。
  - 其中一人写，一人分析和审查；两人可互换角色。
- **集体所有制(Collective Ownership)**
  - 所有成员为所有软件制品负责，同时有权修改任何代码。
- **持续集成(Continuous Integration)**
  - 新代码在经过验证之后就即刻“入库”，通过集成之后成为软件系统的一部分

极限编程（XP）

※ 40小时周工作制

- 每周工作通常不宜超过40小时；坚决杜绝连续两周加班
- 现实情形是：一方面软件团队总是频繁地加班，而另一方面软件项目总是延期完成。
- 无奈的是，尽管所有人都知道非自愿加班不能提升效率，但还是不得不加班。
- 特别是，接近项目收尾阶段的团队成员通常需要每周工作超80小时。

※ 在线(On-site)客户

- 将客户视为团队固定成员，负责实时解答领域和业务问题。

※ 编码标准:

- 当前几乎所有大企业和大项目都会定制编码标准。

敏捷方法的实际表现

相比于Royce(瀑布)模型、普通增量和迭代模型，敏捷方法更适合于当前复杂、频变且更苛刻的软件开发大环境。

相关实证：

- ※ Michael Mah的2008年报告
- ※ Shine公司的2003年调查报告
- ※ David Rico的2008年调研论文
- ※ Scott Ambler系列调查报告(2007-2011)
- ※ VersionOne公司的系列调查报告(2006-2011)

敏捷方法是主流开发模型

※ 超50%的软件开发项目采用敏捷方法。

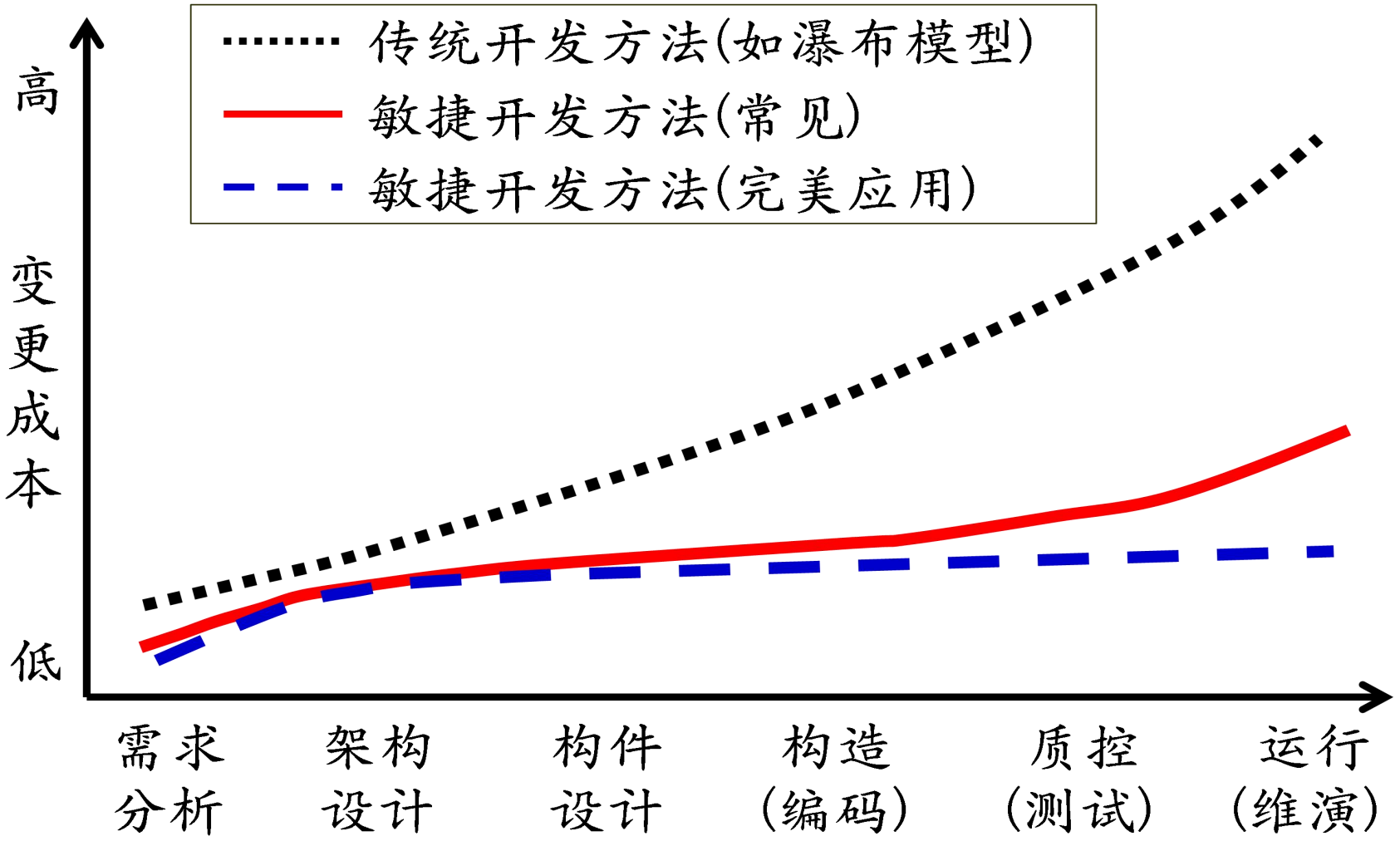
※ 注意

- 敏捷方法广受小项目和小团队的亲睐
- 并不受大项目和大团队的欢迎

Eric Olafson

培养敏捷性更多的是需要文化转型，而不是技术转型。

敏捷型项目的变更成本增长趋势



微软过程

1.微软过程准则

2.微软软件生命周期

（1）规划阶段

（2）设计阶段

（3）开发阶段

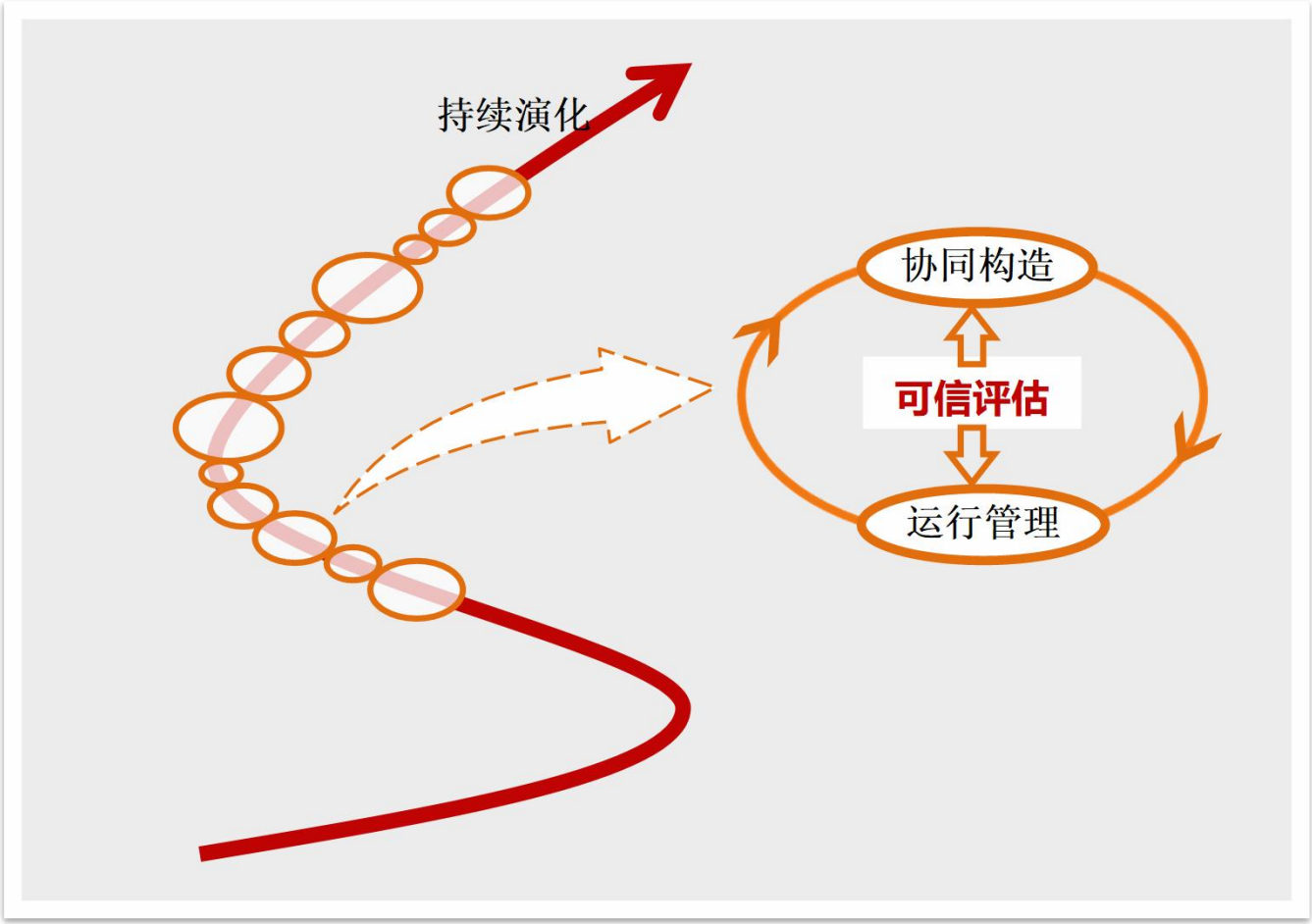
（4）稳定阶段

（5）发布阶段

3.微软过程模型

网构化软件过程

网构化软件是在网络环境下进行构造、运行在网络上、通过网络提供服务的大型复杂软件系统。



网构化软件过程

（1）网构化软件的新特性突出表现为：开放、动态、复杂、演化。

（2）网构化软件的开发过程具有开放性和协同性，要求其运行过程具有可维性和适应性，而且必须通过综合评估降低复杂性导致的软件演化的不可控性、提升其可信性。

（3）网构化软件的生命周期本质上是一种不断适应变化的持续演化过程，而每次局部演化都是经过可信评估的协同构造和运行管理两个过程的深度融合。