



北京郵電大學

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

# 软件工程

# TSEG Software Engineering

*Telecommunications Software Engineering Group*

## 第二章 软件生命周期模型

黄海 [hhuang@bupt.edu.cn](mailto:hhuang@bupt.edu.cn)



- ◆ 软件工程过程
- ◆ 软件生命周期
- ◆ 软件过程模型
- ◆ 传统软件生命周期模型
- ◆ 新型软件生命周期模型



### 工程项目的PDCA循环（戴明环）

美国质量管理专家戴明博士针对工程项目的质量目标，将全面质量管理思想引入工程项目过程，提出了PDCA循环，也称为戴明环。

即Plan（规划）、Do（执行）、Check（检查）、Action（处理）等抽象活动的循环。



### 软件工程过程(Software Engineering Process)

软件工程过程是为获得软件产品，在软件工具支持下由软件工程师完成的一系列软件工程活动。软件工程过程遵循**PDCA**抽象活动，包含四种基本的过程活动：

- P (Plan)** : 软件规格说明。规定软件的功能及其使用的限制；
- D (Do)** : 软件开发。产生满足规格说明的软件；
- C (Check)** : 软件确认。通过有效性验证以保证软件能够满足客户的要求；
- A (Action)** : 软件演进。为满足客户的变更要求，软件必须在使用的过程中不断地改进。

事实上，软件工程过程是一个软件开发机构针对某一类软件产品为自己规定的工作步骤，它应当是科学的、合理的，否则必将影响到软件产品的质量。



- ◆ 软件工程过程
- ◆ 软件生命周期
- ◆ 软件过程模型
- ◆ 传统软件生命周期模型
- ◆ 新型软件生命周期模型



**软件生命周期 (software life cycle )**是指软件产品从考虑其概念开始，到该软件产品不再使用为止的整个时期，一般包括概念阶段、分析与设计阶段、构造阶段、移交阶段等不同时期。

在整个软件生命周期中贯穿了软件工程过程的六个基本活动：

- ① **制定计划**：确定要开发软件系统的总目标，给出它的功能、性能、可靠性以及接口等方面的要求；研究完成该项软件任务的可行性，探讨解决问题的可能方案；制定完成开发任务的实施计划，连同可行性研究报告，提交管理部门审查。



- ② **需求分析和定义**: 对待开发软件提出的需求进行分析并给出详细的定义。编写出软件需求说明书及初步的用户手册，提交管理机构评审。
- ③ **软件设计**: 设计是软件工程的技术核心。把已确定了的各项需求转换成一个相应的体系结构。进而对每个模块要完成的工作进行具体的描述。编写设计说明书，提交评审。
- ④ **程序编写**: 把软件设计转换成计算机可以接受的程序代码。
- ⑤ **软件测试**: 在设计测试用例的基础上检验软件的各个组成部分。
- ⑥ **运行 / 维护**: 已交付的软件投入正式使用，并在运行过程中进行适当的维护。



- ◆ 软件工程过程
- ◆ 软件生命周期
- ◆ 软件过程模型
- ◆ 传统软件生命周期模型
- ◆ 新型软件生命周期模型



**模型**是实际事物、实际系统的抽象。

**软件过程模型**是从一个特定角度提出的对软件过程的简化描述，是对软件开发实际过程的抽象，它包括构成软件过程的各种**活动**、**软件工件**（artifact）以及**参与角色**等。



从软件过程的三个组成成分可以将软件过程模型划分为三种类型：

### (1) 工作流 (work flow) 模型

这类模型描述软件过程中各种活动的序列、输入和输出，以及各种活动之间的相互依赖性。它强调软件过程中活动的组织控制策略。

### (2) 数据流 (data flow) 模型

这类模型描述将软件需求变换成软件产品的整个过程中的活动，这些活动完成将输入工件变换成输出工件的功能。它强调软件过程中的工件的变换关系，对工件变换的具体实现措施没有加以限定。

### (3) 角色/动作模型

这类模型描述了参与软件过程的不同角色及其各自负责完成的动作，即根据参与角色的不同将软件过程应该完成的任务划分成不同的职能(function area)。它强调软件过程中角色的划分、角色之间的协作关系，并对角色的职责进行了具体的规定。



- ◆ 软件工程过程
- ◆ 软件生命周期
- ◆ 软件过程模型
- ◆ 传统软件生命周期模型
- ◆ 新型软件生命周期模型



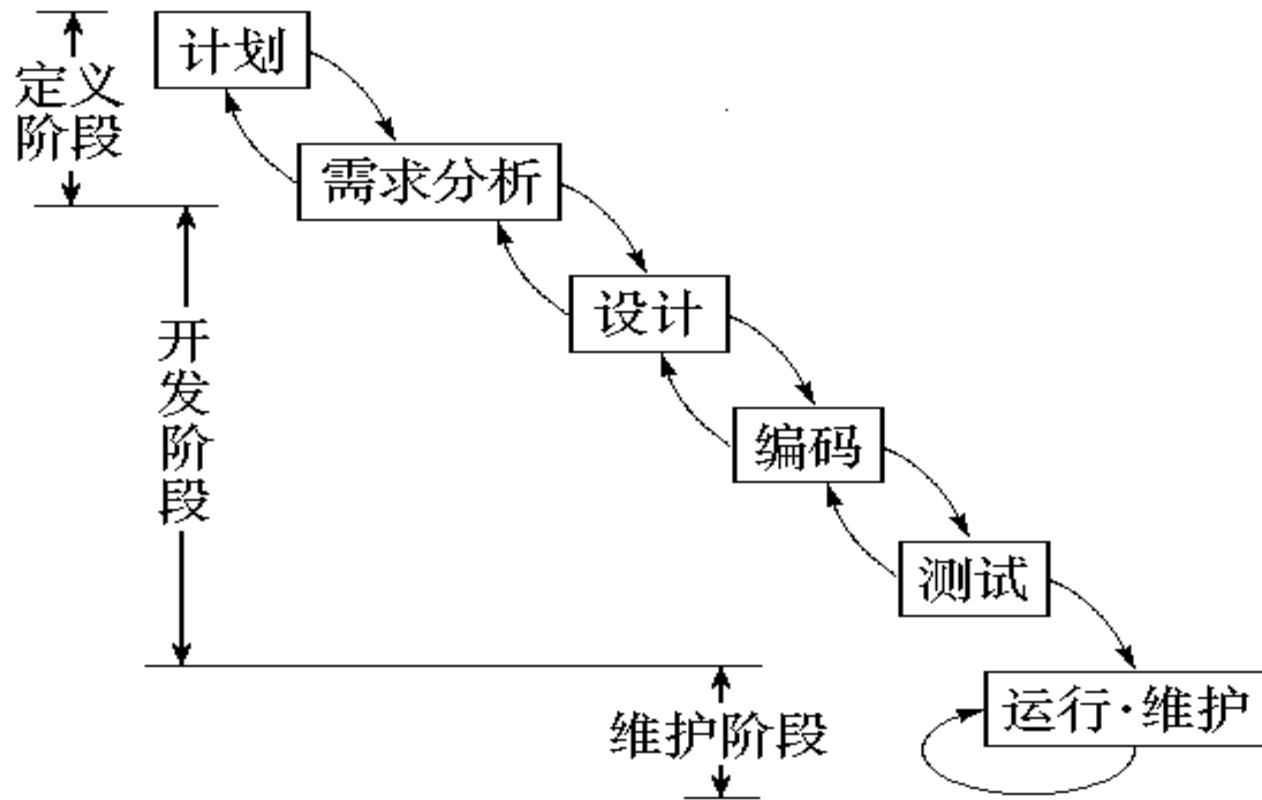
软件过程模型有时也称软件生命周期模型，即描述从软件需求定义直至软件经使用后废弃为止，跨越整个生存期的软件开发、运行和维护所实施的全部过程、活动和任务的结构框架，同时描述生命周期不同阶段产生的软件工件，明确活动的执行角色等。

### 九个传统软件生命周期模型：

- . 瀑布模型
- . V模型和W模型
- . 原型方法
- . 演化模型
- . 增量模型
- . 螺旋模型
- . 喷泉模型
- . 构件组装模型
- . 快速应用开发模型



### 1. 瀑布模型 (waterfall model)



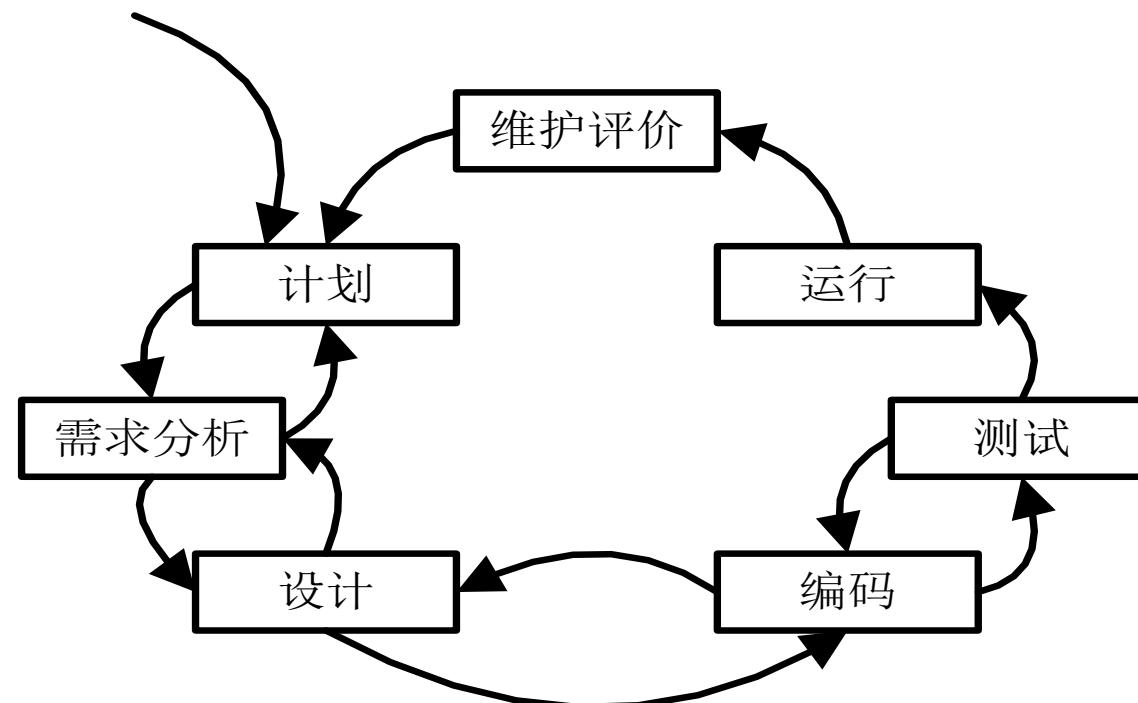


## § 2.4 传统软件生命周期模型

- ◆ 1970年，W.Royce提出瀑布模型。瀑布模型规定了软件生命周期提出的六个基本工程活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水
- ◆ 瀑布模型将软件生命周期划分为**定义阶段**、**开发阶段**和**维护阶段**，在**定义阶段**部署了计划和需求分析活动；在**开发阶段**部署了设计、编码和测试活动，**维护阶段**部署了运行/维护活动
- ◆ **瀑布模型中的每一个开发活动具有下列特征**
  - (1) 本活动的工作对象来自于上一项活动的输出
  - (2) 根据本阶段的活动规程执行相应的任务。
  - (3) 产生本阶段活动相关产出—软件工件，作为下一活动的输入。
  - (4) 对本阶段活动执行情况进行评审。



- ◆ 瀑布模型中的运行/维护活动，是一个具有最长生命周期的循环往复阶段。





### 瀑布模型优点：

- (1) 软件生命周期的阶段划分不仅降低了软件开发的复杂程度，而且提高了软件开发过程的透明性，便于将软件工程过程和软件管理过程有机地融合在一起，从而提高软件开发过程的可管理性。
- (2) 推迟了软件实现，强调在软件实现前必须进行分析和设计工作。
- (3) 瀑布模型以项目的阶段评审和文档控制为手段有效地对整个开发过程进行指导，保证了阶段之间的正确衔接，能够及时发现并纠正开发过程中存在的缺陷，从而使产品达到预期的质量要求。



### 瀑布模型缺点：

- (1) 模型缺乏灵活性，特别是无法解决软件需求不明确或不准确的问题，这是瀑布模型最突出的缺点。因此，瀑布模型只适合于**需求明确**的软件项目。
- (2) 模型的风险控制能力较弱。成品时间长；体系结构的风险和错误只有在测试阶段才能发现，返工导致项目延期。
- (3) 软件活动是文档驱动的，文档过多会增加工作量，文档完成情况会误导管理人员。



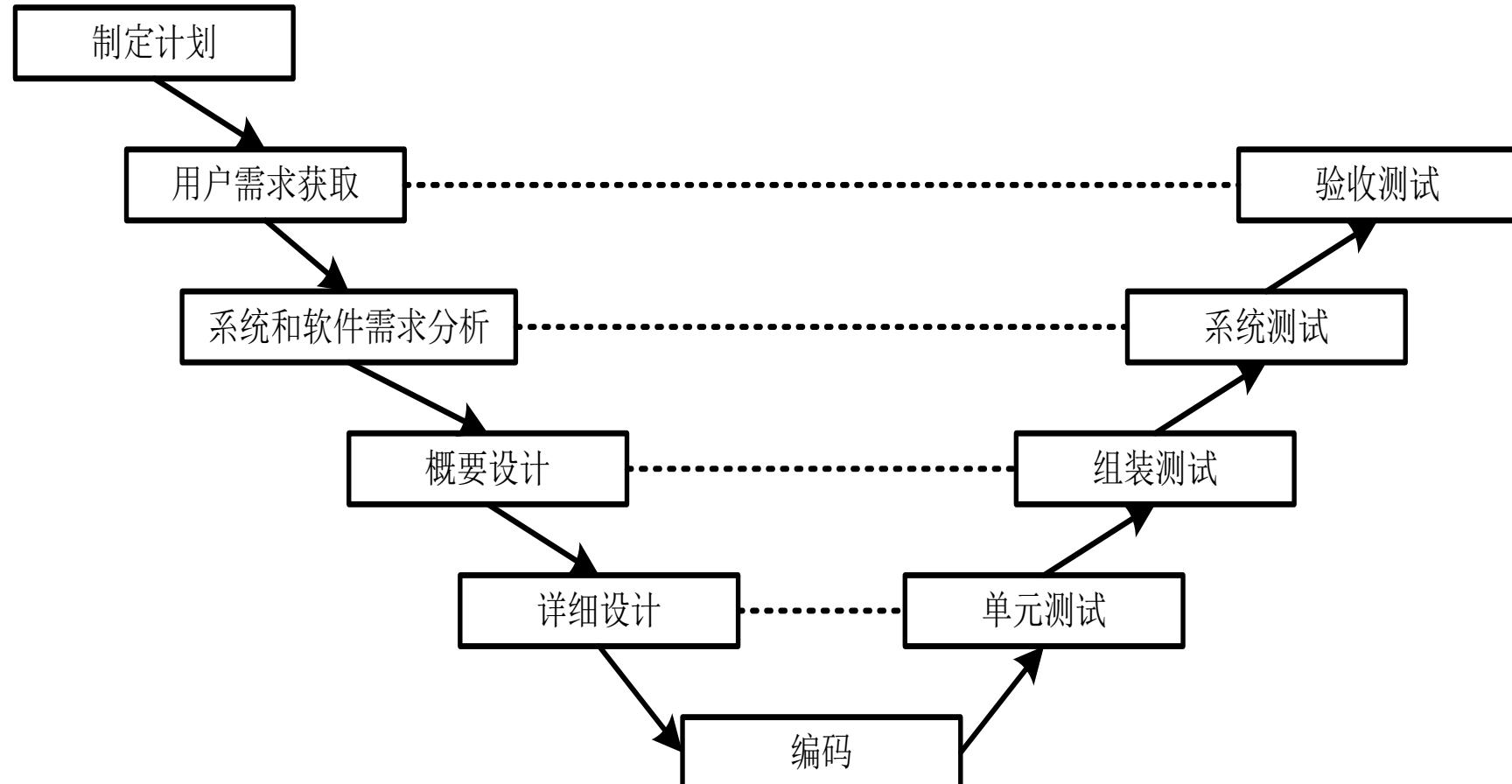
### 2. V模型和W模型

#### (1) V模型——瀑布模型的变种

- 瀑布模型将测试作为软件实现之后的一个独立阶段，没有强调测试的重要性。
- 针对瀑布模型这个缺点，1980年代后期Paul Rook提出了V模型。
- V模型的价值在于纠正了人们不重视测试阶段重要性的错误认识，将测试分等级，并和前面的开发阶段对应起来。



## § 2.4 传统软件生命周期模型



V模型示意图

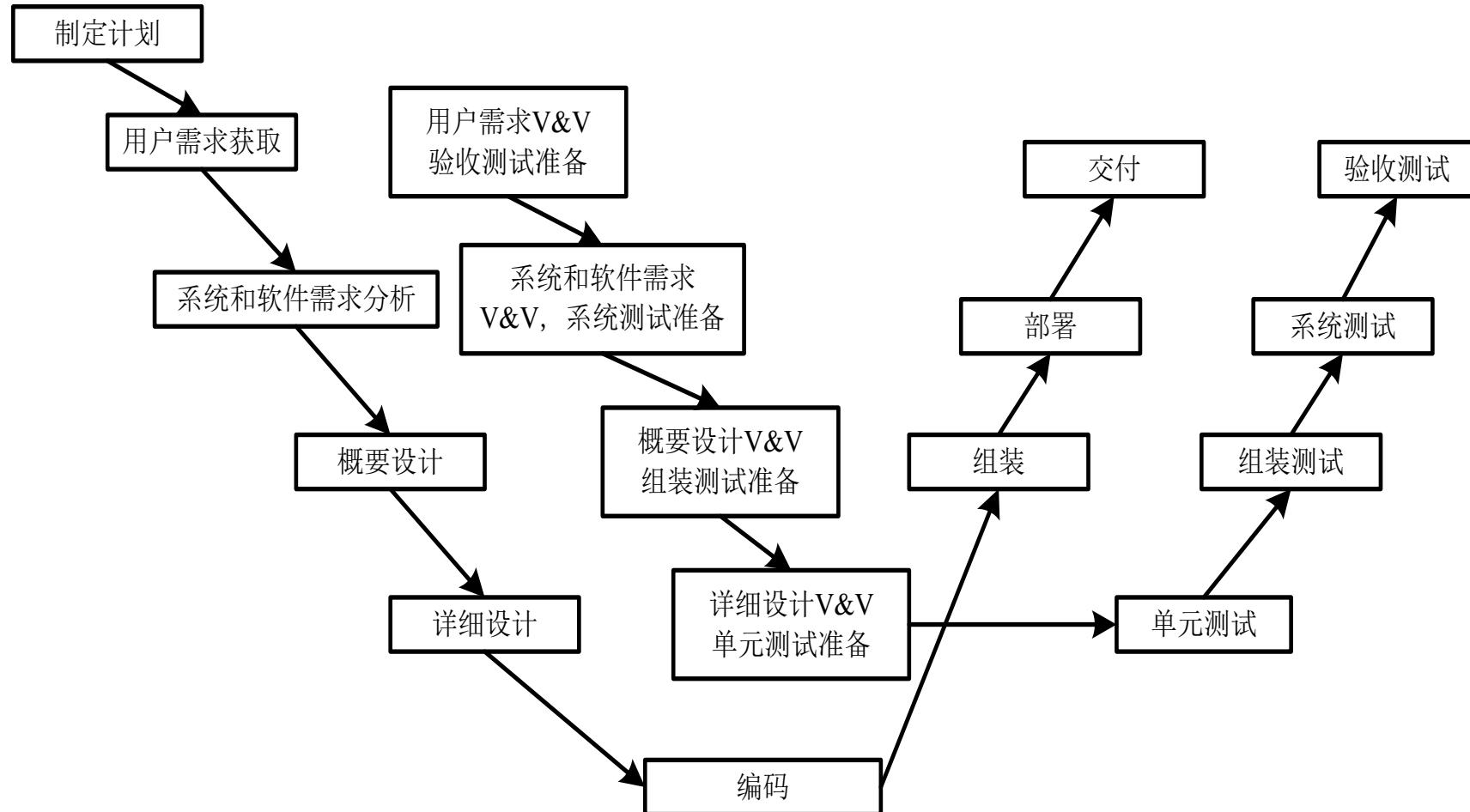


### (2) W模型——瀑布模型的变种

- V模型仍然将测试作为一个独立的阶段，所以并没有提高模型抵抗风险的能力。
- Evolutif公司在V模型的基础上提出了W模型，将测试广义化，增加了**确认**和**验证**内容，并贯穿整个软件生命周期。
- W模型由两个V型模型组成，分别代表测试与开发过程，两个过程是同步进行的。



## § 2.4 传统软件生命周期模型



W模型示意图



### 3. 原型方法(prototyping)

- 完整准确的需求规格说明很难得到
  - 早期用户对系统只有一个模糊的想法；
  - 开发过程中用户可能提出新的需求；
  - 环境的变化也要求开发过程中的系统随之改变；
  - 预料之外的实际困难使得开发人员不得不改变需求来适应。
- 通过加强评审和确认、全面测试也不能从根本上解决需求不稳定带来的问题。
- 为了解决这些问题，逐渐形成了软件系统的原型建设思想



### (1) 原型方法概述

- **原型：**是指模拟某种产品的原始模型。软件原型是一个早期可以运行的版本，它反映最终系统的部分重要特性。
- 原型方法构造软件系统
  - 获得一组基本的需求说明，快速分析构造出一个小型的软件系统，满足用户的基本要求；
  - 用户试用原型系统，对其进行反应和评价；
  - 开发者根据用户意见对原型进行改进，获得新的原型版本；
  - 周而复始，直到产品满足用户的要求。
- 原型化方法是在研究需求分析技术的过程中产生的，但也可以用于软件开发的其他阶段



- 原型的种类（目的）
  - 探索型：弄清对目标系统的要求
  - 实验型：系统实现前考察系统的可行性
  - 进化型：将原型扩展到开发过程，通过原型开发逐步实现所有系统功能。
- 原型的使用策略
  - 废弃策略：探索型和实验型
  - 追加策略：进化型
- 原型不同于最终的系统，需要快速实现和运行，因此，原型可以忽略一切暂时不必关心的部分（抽象）



### ➤ 原型方法的优点

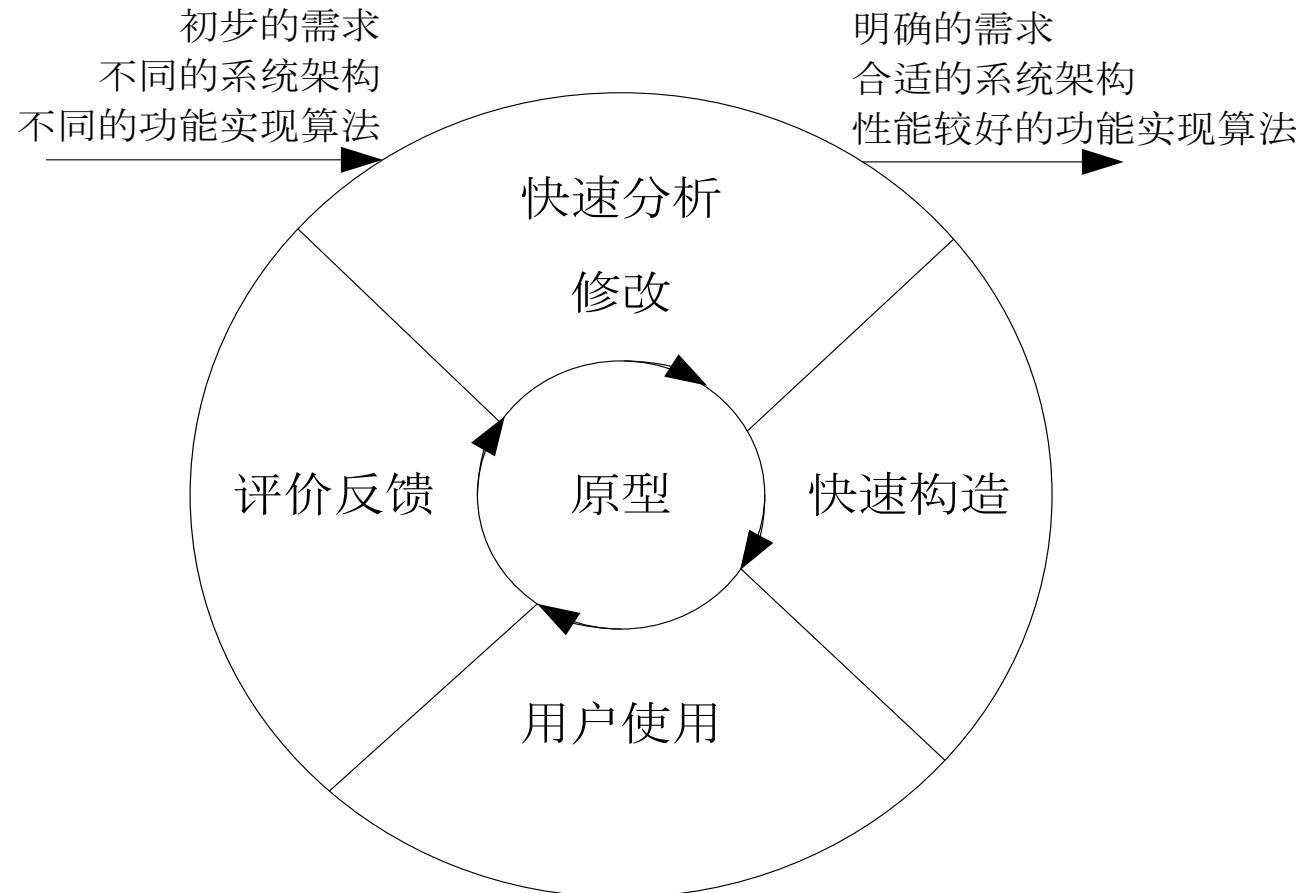
- 有助于增进软件人员和用户对系统服务需求的理解
- 提供了一种有力的学习手段
- 容易确定系统的性能、服务的可应用性、设计的可行性和产品的结果
- 原型的最终版本可作为最终产品或最终系统的一部分

### ➤ 原型方法的缺点

- 文档容易被忽略
- 建立原型的许多工作会被浪费掉
- 项目难以规划和管理



### (2) 原型方法应用过程





- 快速分析
- 快速构造

尽快实现一个可运行的系统，可忽略目标系统在某些细节（如安全性、健壮性、异常处理等）上的要求。

- 用户使用
- 评价反馈

是否满足规格说明的要求；纠正分析过程中的一些误解和错误；增补新的要求

- 修改
  - 反复迭代，直到形成最终产品



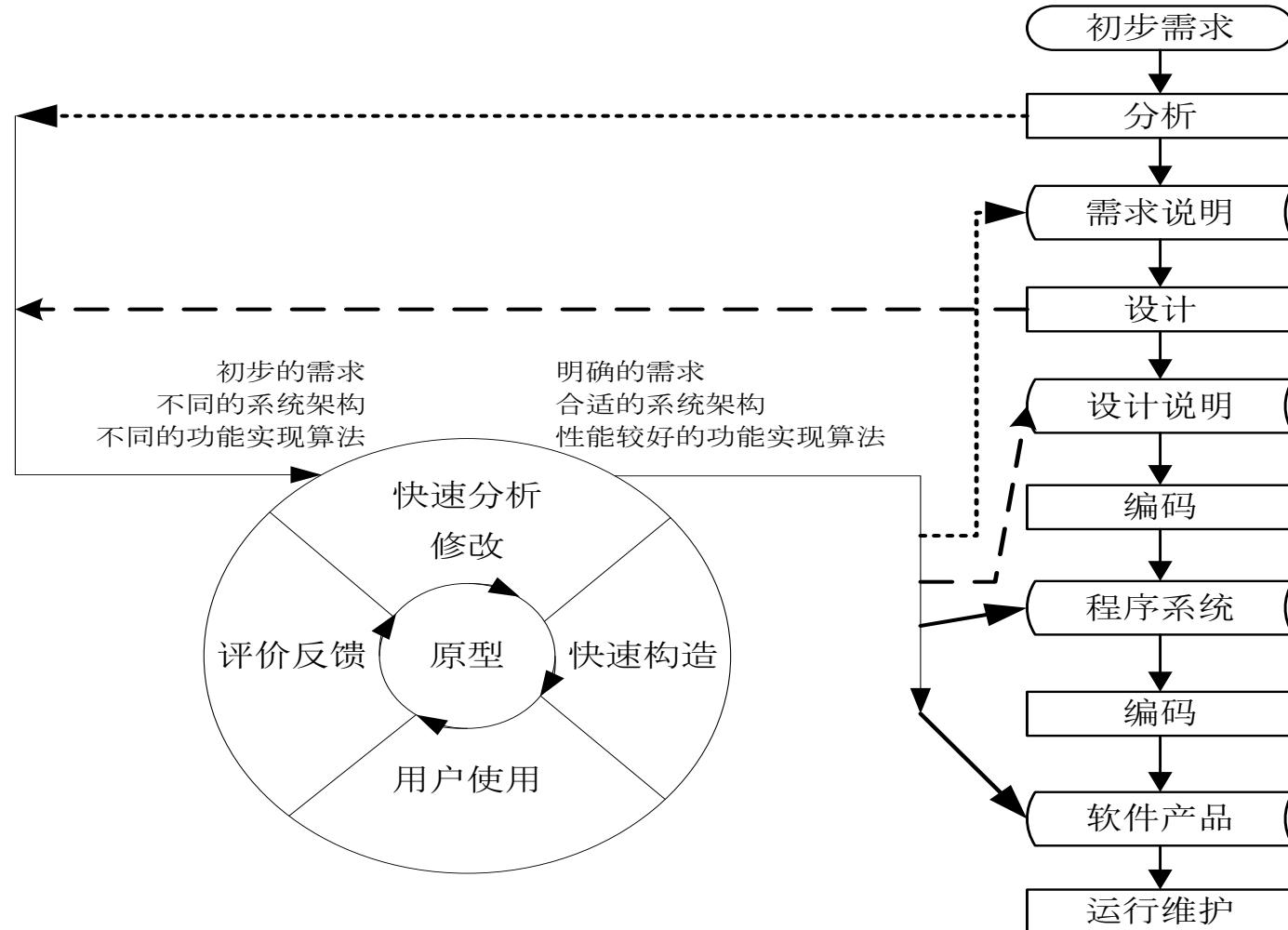
### (3) 原型方法支持的软件生命周期

原型方法可以支持软件生命周期的不同阶段

- 辅助或代替分析阶段（确定需求）
- 辅助设计阶段（确定设计方案的合理性）
- 代替分析与设计阶段
- 代替分析、设计和实现阶段
- 代替全部开发阶段（典型的演化模型）



## § 2.4 传统软件生命周期模型



### 原型方法支持下的软件生命周期



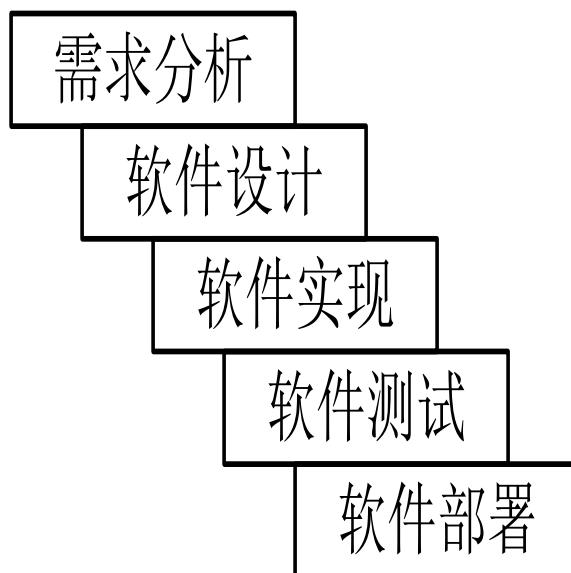
### 4. 演化模型

- 项目开发初始阶段对需求的认识不够清晰，使得开发工作出现再开发在所难免。经验告诉我们：开发“**两次**”后的软件能较好地满足用户的要求。
- **第一次：**试验开发，目的是探索可行性，弄清楚项目的需求。第一次得到的试验性产品称为“**原型**”。
- **第二次：**在第一次的原型基础上进行开发，从而获得较为满意的软件产品。

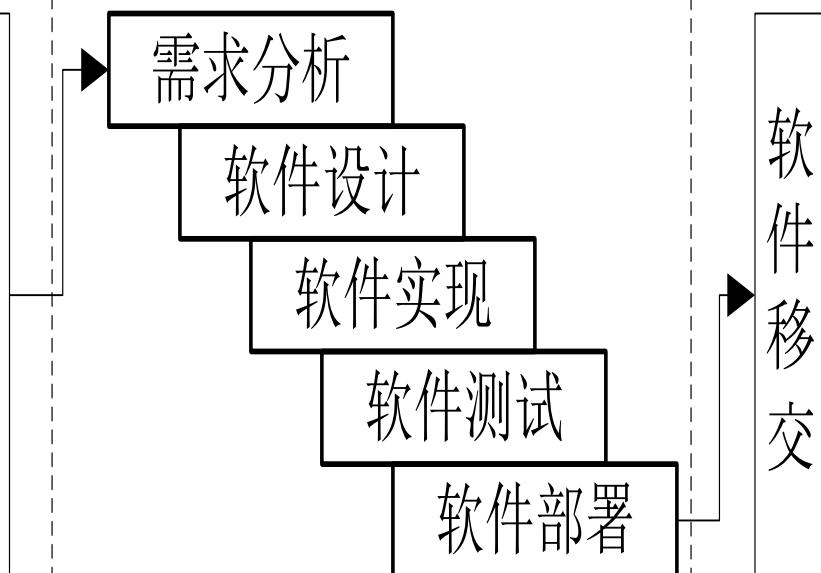


## § 2.4 传统软件生命周期模型

第一次试验开发



第二次产品开发



演化模型示意图



- 演化模型主要针对需求不是很明确的软件项目
- 演化模型缺点
  - 可能会抛弃瀑布模型的文档控制优点，开发过程不透明
  - 探索式演化模型可能会导致最后的软件系统的系统结构较差
  - 可能会用到一些不符合主流、不符合要求或者不成熟的工具和技术

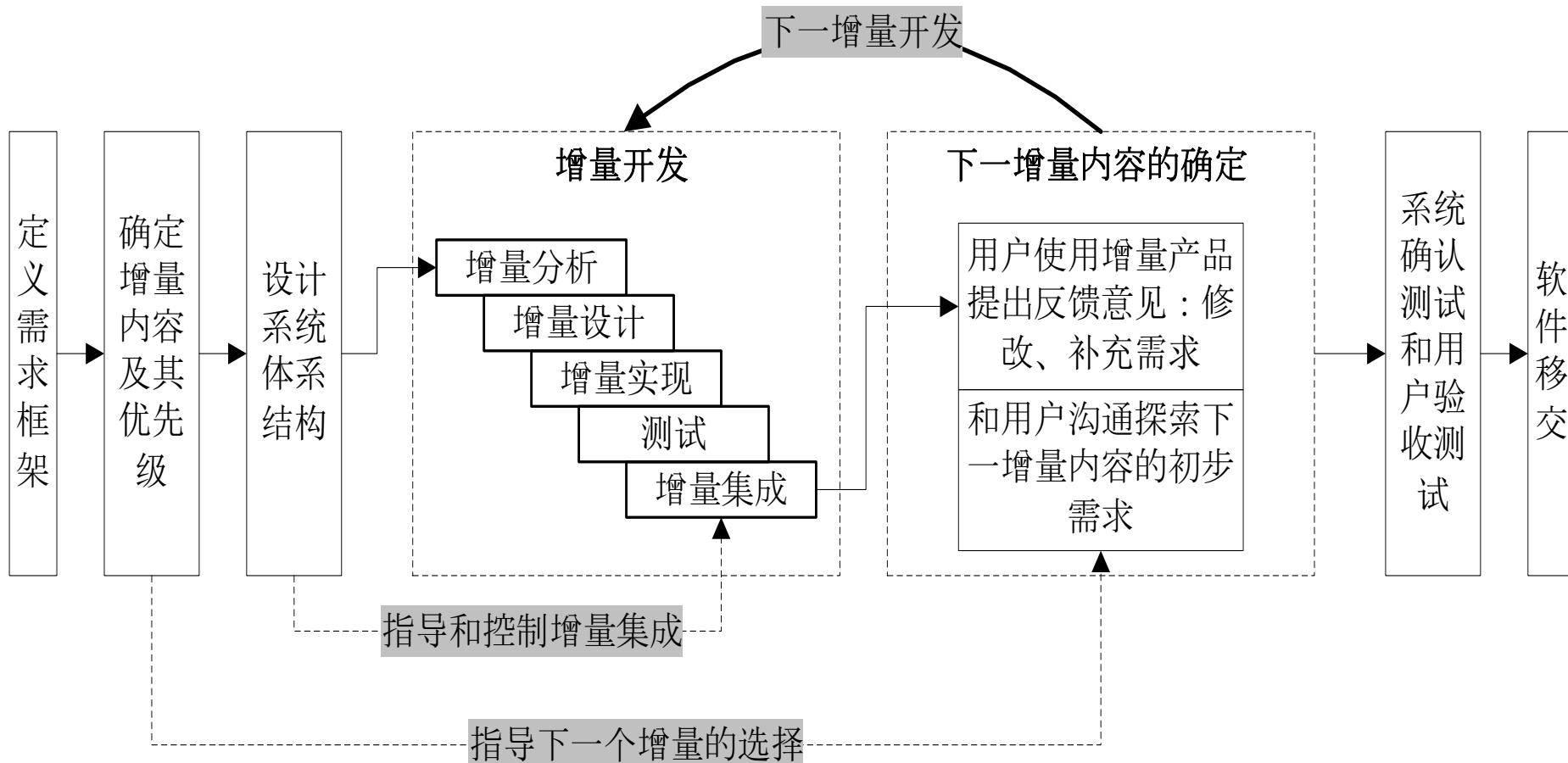


### 5. 增量模型

- 增量模型首先由Mills等人于1980年提出，结合了瀑布模型和演化模型的优点。
- 允许客户的需求可以逐步提出来；每一次“增量”需求的划分与“增量”实现的集成是以不影响系统体系结构为前提的。
- 在增量模型中，客户定义需求框架，确定系统需求实现的优先级；此后针对核心需求以及系统的性能要求确定系统的体系结构，并以此体系结构指导增量的集成，保证在整个开发过程中体系结构的稳定性。



## § 2.4 传统软件生命周期模型



增量模型示意图



### ➤ 增量模型优点

- 增强了客户使用系统的信心，逐步提出对后续增量的需求
- 项目总体失败的风险较低
- 增量从高到低的优先级确定保障了系统重要功能部分的可靠性
- 同一个体系结构提高了系统的稳定性和可维护性

### ➤ 增量模型缺点

- 增量的粒度选择问题
- 确定所有的基本业务服务比较困难



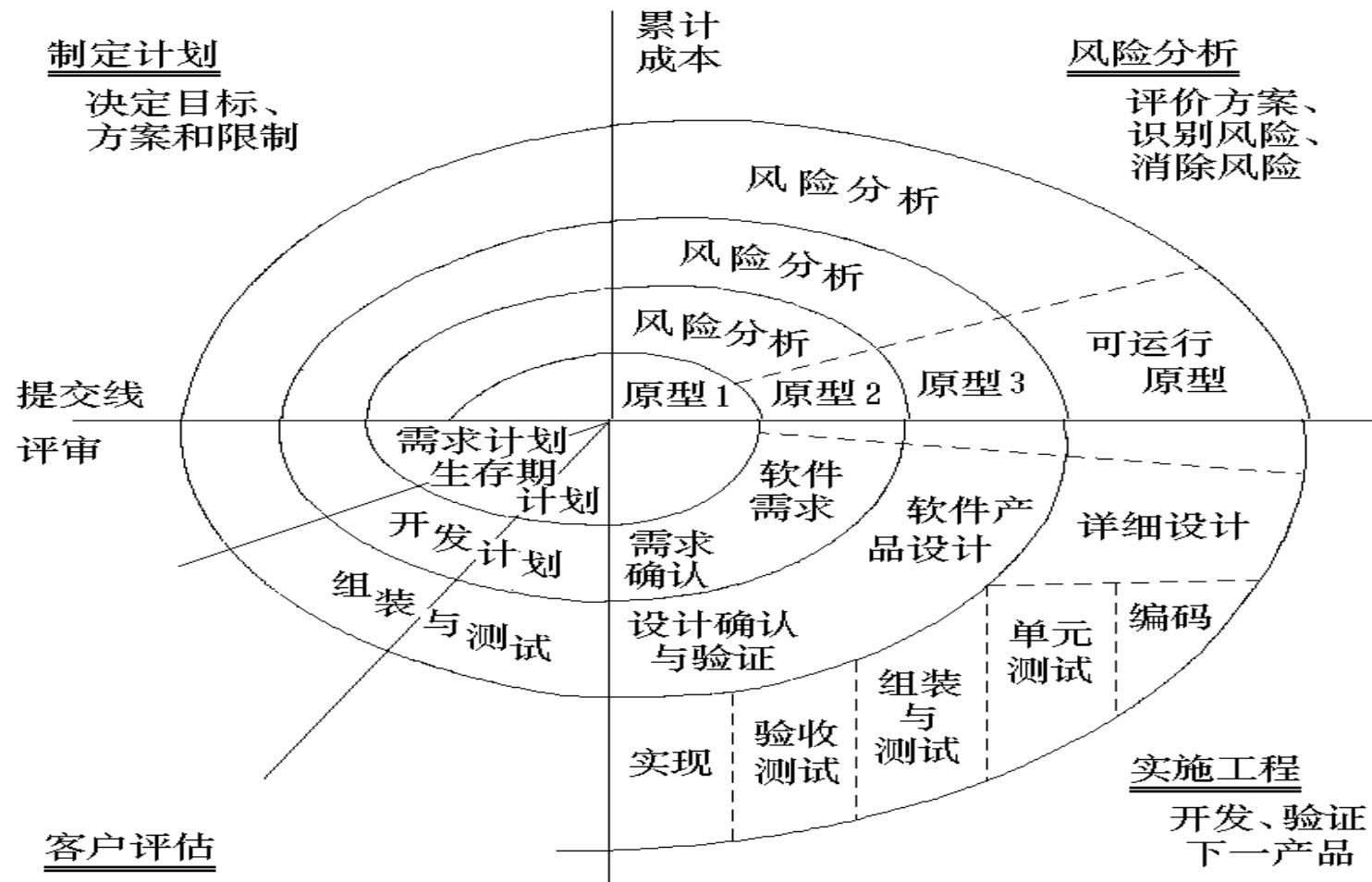
### 6. 螺旋模型

螺旋模型是Boehm于1988年针对**大型软件项目**的特点提出来的

- 对于复杂的大型软件而言，事先不能完整清晰地定义需求是常事，而且设计方案、技术实现方案不允许出现问题，也需要经过多次试验才能明确下来，开发一个只明确需求的原型是远远不能解决问题的，需要开发内容逐步丰富的多个原型。
- 大型软件项目往往存在着诸多风险因素，螺旋模型将瀑布模型与演化模型结合起来，并加入了两种模型均忽略了的风险分析。因为大型项目的规模和复杂性增加，软件开发过程中必然存在着许多风险问题，风险分析是保证项目成功的必要手段。



## § 2.4 传统软件生命周期模型



### 螺旋模型示意图



螺旋模型沿着螺线旋转，在四个象限上分别表达了四个方面的活动，即：

- 制定计划——确定软件目标，选定实施方案，弄清项目开发的限制条件
- 风险分析——分析所选方案，考虑如何识别和消除风险
- 实施工程——实施软件开发
- 客户评估——评价开发工作，提出修正建议

螺旋模型适合于大型软件的开发；然而风险分析需要相当丰富的评估经验，风险的规避又需要深厚的专业知识，这给螺旋模型的应用增加了难度。



### 7. 喷泉模型(迭代模型)

喷泉模型认为软件开发过程具有两个固有的本质特征：

➤ 迭代

多次重复、演进。

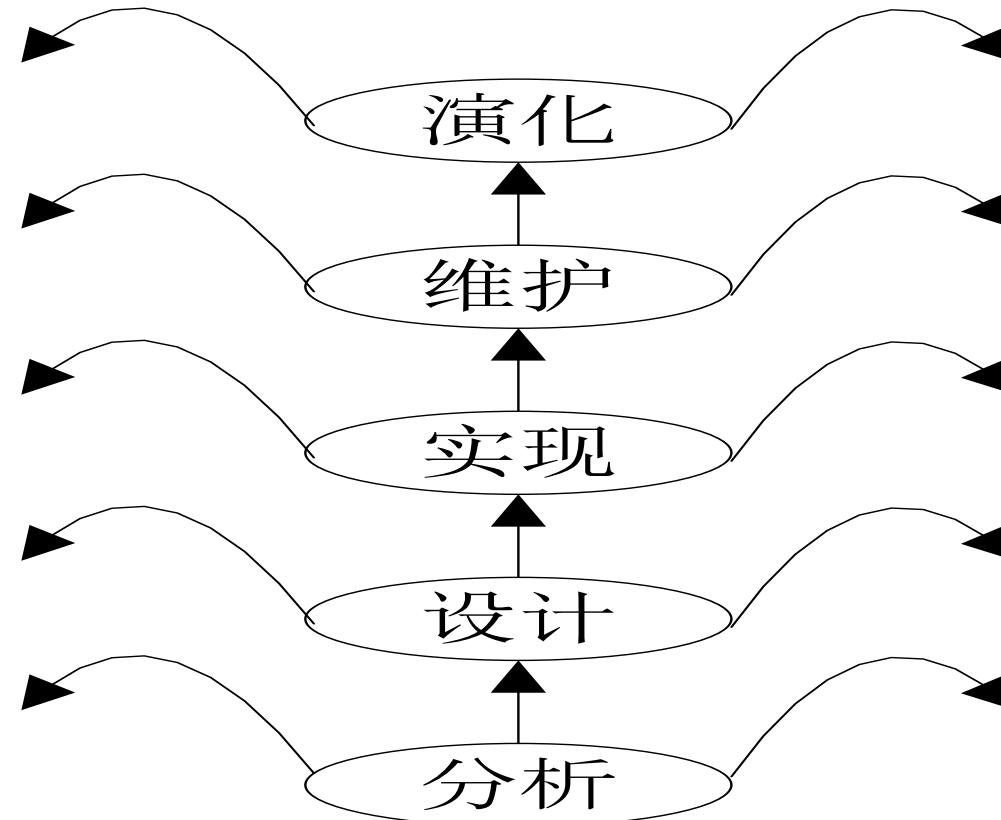
➤ 无间隙

各阶段间无明显的界限。支持分析和设计结果的自然复用。

**适用：**面向对象的软件开发过程。对象概念的引入，对象及对象关系在分析、设计和实现阶段的表达方式的统一，使得开发活动之间的迭代和无间隙性能够容易地实现。



## § 2.4 传统软件生命周期模型



喷泉模型示意图

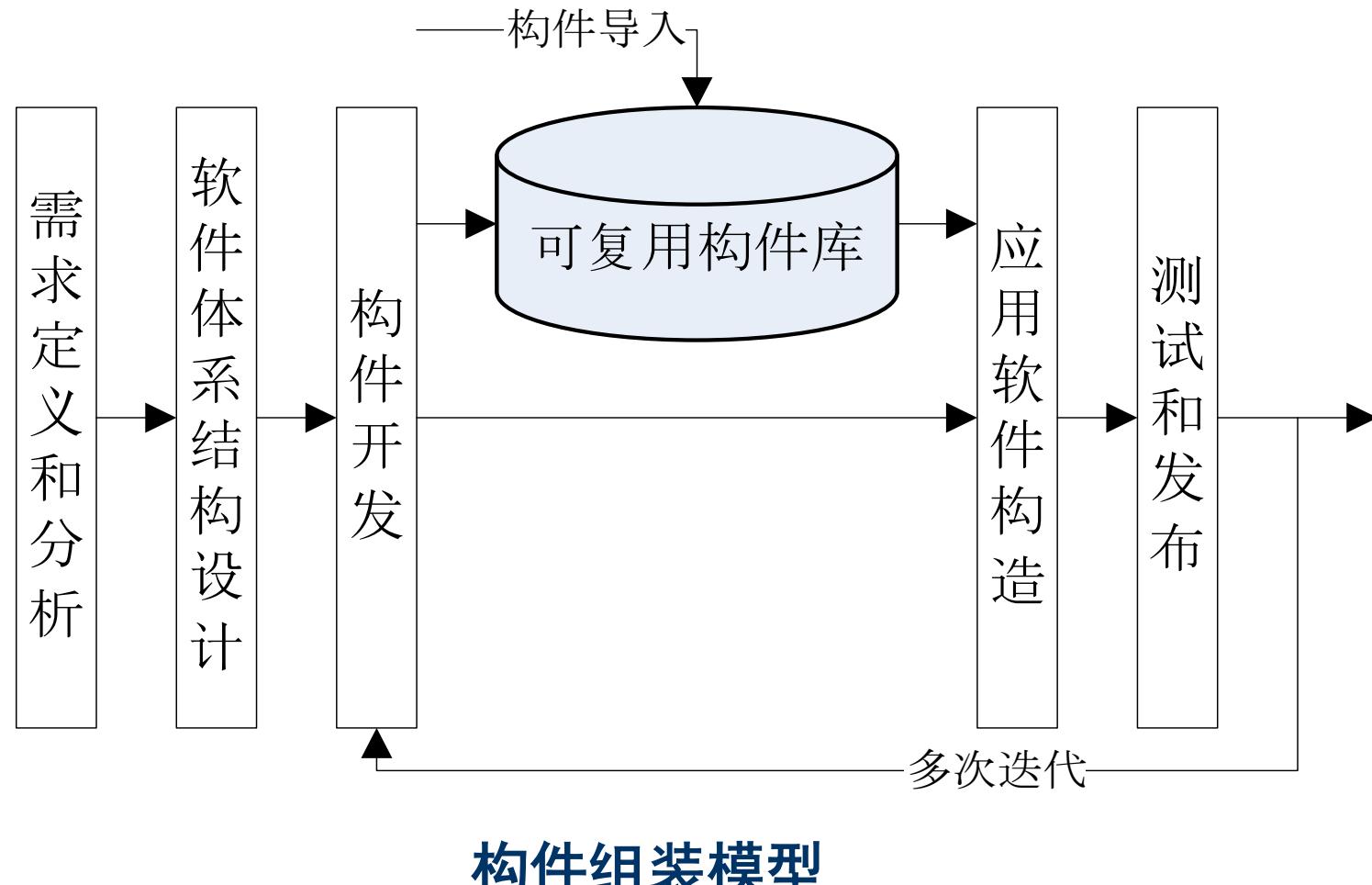


### 8. 构件组装模型

- 构件组装模型本质上是演化的，开发过程是迭代的。
- 构建组装模型由五个阶段组成：
  - 需求定义和分析
  - 软件体系结构设计
  - 构件开发
  - 应用软件构造
  - 测试和发布



## § 2.4 传统软件生命周期模型





- 软件的开发过程步骤如下：
  - (1) 定义和分析需求；
  - (2) 标识本项目需要什么构件；
  - (3) 从库中查找构件或相似的构件；
  - (4) 如果可用转(5)，否则自行开发或修改，确认后入库；
  - (5) 构造为新系统作第m次迭代；
  - (6) 测试、确认。



### 9. 快速应用开发(RAD)模型

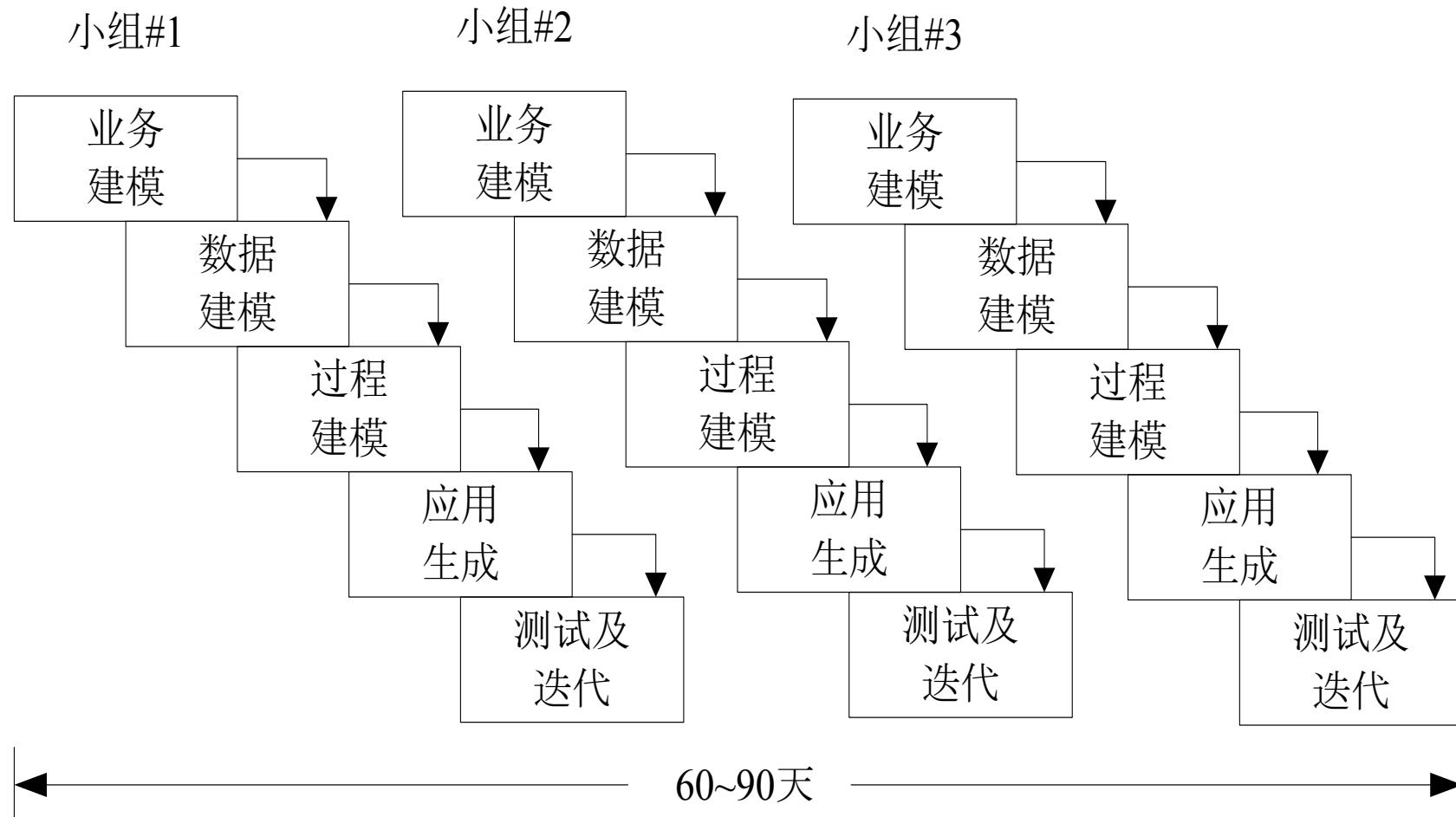
- 快速应用开发(Rapid Application Development, RAD)是一个增量型的软件开发过程模型，采用构件组装方法进行快速开发。
- RAD模型包含如下阶段：
  - (1) 业务建模：通过捕获业务过程中信息流的流动及处理情况描述业务处理系统应该完成的功能。回答以什么信息驱动业务过程运作？要生成什么信息？谁生成它？信息流的去向？由谁处理？可以辅之以数据流图。
  - (2) 数据建模：对于支持业务过程的数据流，建立数据对象集合，定义数据对象属性，与其它数据对象的关系构成数据模型，可辅之以E-R图。



- (3) **过程建模：**定义如何使数据对象在信息流中完成各业务功能。描述数据对象的增加、修改、删除、查找。即细化数据流图中的处理框。
- (4) **应用生成：**利用第四代语言(4GL)写出处理程序，重用已有构件或创建新的可重用构件，利用环境提供的工具，自动生成，构造出整个的应用系统。
- (5) **测试及迭代：**由于大量重用，一般只作总体测试，但新创建的构件还是要测试的。当一轮需求完成快速开发后，可以迭代进入下一轮需求的开发。



## § 2.4 传统软件生命周期模型



### RAD模型示意图



- ◆ 软件工程过程
- ◆ 软件生命周期
- ◆ 软件过程模型
- ◆ 传统软件生命周期模型
- ◆ 新型软件生命周期模型



### 1. 统一软件开发过程 (RUP)

- RUP (Rational Unified Process) 是由 Rational公司开发的一种软件工程过程框架，是一个面向对象的基于web的程序开发方法论。
- RUP既是一种软件生命周期模型，又是一种支持面向对象软件开发的工具，它将软件开发过程要素和软件工件要素整合在统一的框架中。

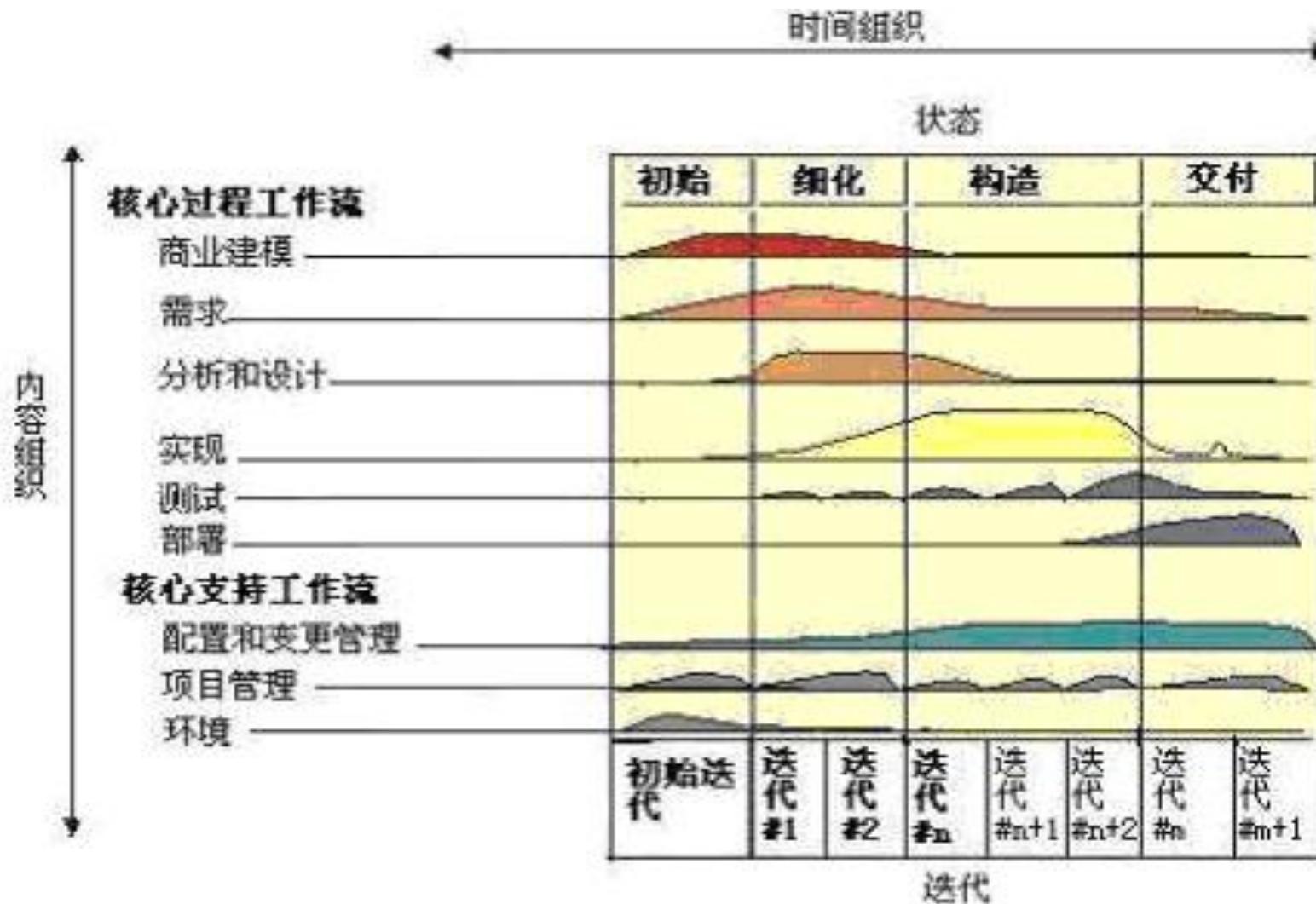


### (1) RUP的基本结构

- RUP是一个二维的软件开发模型。
- 横轴在时间上将生命周期过程展开成四个阶段（Phase），每个阶段特有的里程碑（Milestone）是该阶段结束的标志，每个阶段里又划分为不同的迭代（Iteration），体现了软件开发过程的动态结构。
- 纵轴按照活动的内容进行组织，包括活动（activity）、活动产出的工件（artifact）、活动的执行角色（worker）以及活动执行的工作流（workflow），体现软件开发过程的静态结构。



## § 2.5 新型软件生命周期模型



RUP二维软件开发模型



### RUP中的软件生命周期的四个顺序阶段：

#### (1) 初始阶段

生命周期目标 (Lifecycle Objective) 里程碑

#### (2) 细化阶段

生命周期体系结构 (Lifecycle Architecture) 里程碑

#### (3) 构造阶段

初始运行能力 (Initial Operational Capability) 里程碑

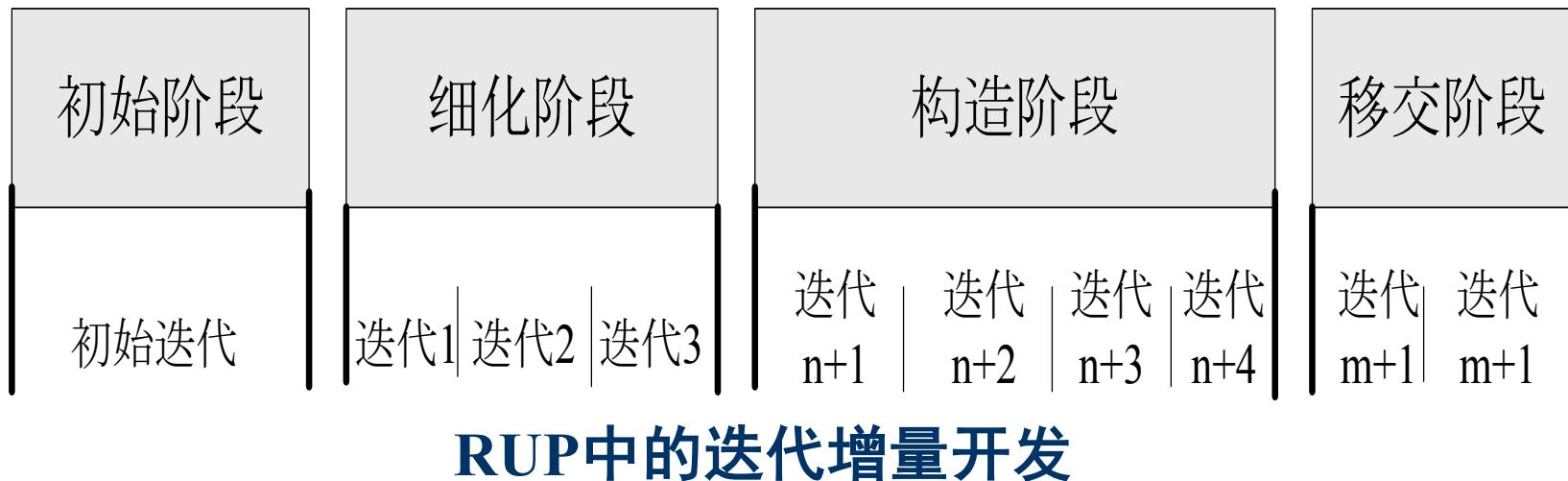
#### (4) 移交阶段

产品发布 (Product Release) 里程碑



### (2) RUP的迭代增量开发思想

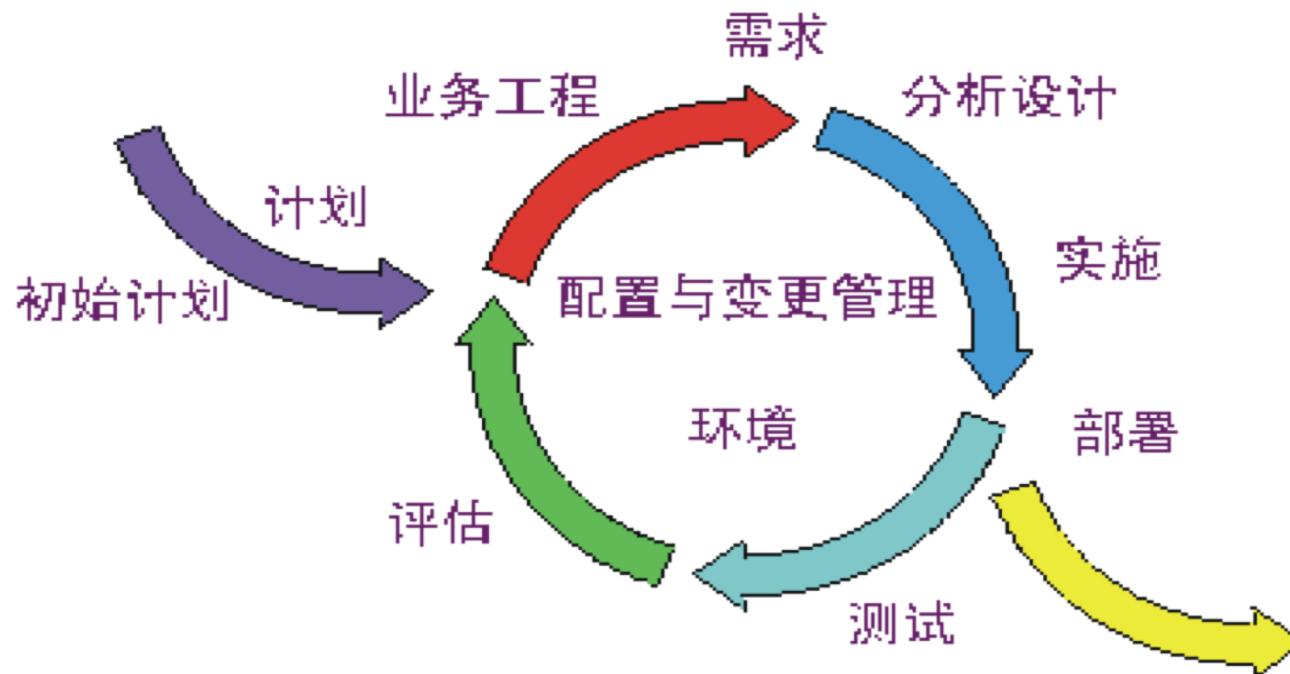
- RUP的每一个阶段可以进一步划分为一个或多个迭代过程，从一个迭代过程到另一个迭代过程增量形成最终的系统。
- RUP是融合了喷泉模型和增量模型的一种综合生命周期模型。





## § 2.5 新型软件生命周期模型

- RUP将整个项目的开发目标划分成一些更易于完成和达到的阶段性小目标。每一次迭代就是为了完成一定阶段性小目标而从事的一系列开发活动，包含需求、设计、实施（编码）、部署、测试等。



RUP中的迭代过程



### (3) RUP的核心工作流

- 6个核心过程工作流 (Core Process Workflows)
  - 商业建模 (Business Modeling)
  - 需求 (Requirements)
  - 分析和设计 (Analysis & Design)
  - 实现 (Implementation)
  - 测试 (Test)
  - 部署 (Deployment)



### ➤ 3个核心支持工作流 (Core Supporting Workflows)

- 配置和变更管理 (Configuration & Change Management)
- 项目管理 (Project Management)
- 环境 (Environment)



### (4) RUP的最佳实践

- 短时间分区式的迭代
- 适应性开发
- 在早期迭代中解决高技术风险和高业务价值的问题
- 不断地让用户参与迭代结果的评估
- 在早期迭代中建立内聚的核心架构
- 不断地验证质量；尽早、经常和实际地测试
- 使用用例驱动软件建模
- 可视化软件建模：使用UML进行软件建模
- 仔细地管理需求
- 实行变更请求和配置管理



### 2. 敏捷模型(Agile Modeling, AM)

- 敏捷建模是由Scott W. Ambler从许多的软件开发过程实践中归纳总结出来的一些敏捷建模价值观、原则和实践等组成的
- AM只是一种**态度**，不是一个说明性过程，它描述了一种**建模风格**
- AM是对已有生命周期模型的补充，它本身不是一个完整的方法论，在应用传统的生命周期模型时可以借鉴AM的过程指导思想，将主要焦点置于建模过程，然后才是文档，而不要过度建模和过度编制文档。



### (1) 敏捷建模价值观

- 沟通
- 简单
- 反馈
- 勇气
- 谦逊



### (2) 敏捷建模原则

#### ➤ 核心原则

- 1) 主张简单                          2) 拥抱变化
- 3) 软件开发的第二个目标应是可持续性
- 4) 递增的变化                          5) 令项目干系人投资最大化
- 6) 有目的地建模                          7) 多种模型
- 8) 高质量的工作                          9) 快速反馈
- 10) 软件产品是主要目标    11) 轻装前进



### ➤ 补充原则

- 1) 内容比表示更重要
- 2) 三人行必有我师
- 3) 了解软件建模方法
- 4) 了解软件开发工具
- 5) 局部调整
- 6) 开放诚实的沟通
- 7) 利用好直觉



### (3) 敏捷建模实践

#### ➤ 核心实践

- 1) 项目干系人的积极参与      2) 正确使用工件
- 3) 集体所有制                          4) 测试性思维
- 5) 并行创建模型                          6) 创建简单的内容
- 7) 简单地建模                            8) 公开展示模型
- 9) 切换到另外的工件                    10) 小增量建模
- 11) 和他人一起建模                    12) 用代码验证
- 13) 使用最简单的工具



### ➤ 补充实践

- 1) 使用建模标准
- 2) 逐渐应用模式 (pattern)
- 3) 丢弃临时模型
- 4) 合同模型要正式
- 5) 为外部交流建模
- 6) 为帮助理解建模
- 7) 重用现有的资源
- 8) 不到万不得已不更新模型



### 3. 敏捷开发

- › 敏捷开发是一种以人为核心、迭代、循序渐进的开发方法。在敏捷开发中，软件项目的构建被切分成多个**子项目**，各个子项目的成果都经过测试，具备集成和可运行的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。
- › 敏捷方法很多，包括 Scrum、极限编程(XP)、功能驱动开发(FDD)、CrystalClear 等多种法，这些方法本质实际上是一样的，都遵循“敏捷宣言”



### 3.1 敏捷宣言

2001年2月由17位当时称之为“轻量级方法学家”所编写签署的“敏捷宣言”（Agile Manifesto）：

- 个体和交互 胜过 过程和工具
- 可以工作的软件 胜过 面面俱到的文档
- 客户合作 胜过 合同谈判
- 响应变化 胜过 遵循计划



### 3.2 敏捷开发实践

Iteration 迭代开发

Iteration Planning Meeting 迭代计划会议

Story Card/Story Wall/Feature List 用户故事

Standup Meeting 站立会议

Pair Programming 结对编程

CI/Daily Build 持续集成

Retrospect 总结和反思

ShowCase 演示

Refactoring 重构

TDD 测试驱动开发