软件工程与实践

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com，微信电话: 13679081552

2.1 软件过程的概念

2.2 常见软件过程模型

2.3 软件过程模型选择

第二章：软件过程模型

介绍软件过程的概念，常见模型：瀑布模型、原型和增量模型、螺旋和喷泉模型、基于构建的模型、统一软件模型，以及模型的选择。

- 2.1.1 软件生命期概念
- 2.1.2 软件过程概念
- 2.1.3 软件过程模型概念
- 2.1.4 软件能力成熟度模型

2.1 软件过程概念

介绍软件生命期、软件过程、软件过程模型和能力成熟度模型。

2.1.1 软件生命周期

软件生命周期
Software Life Cycle

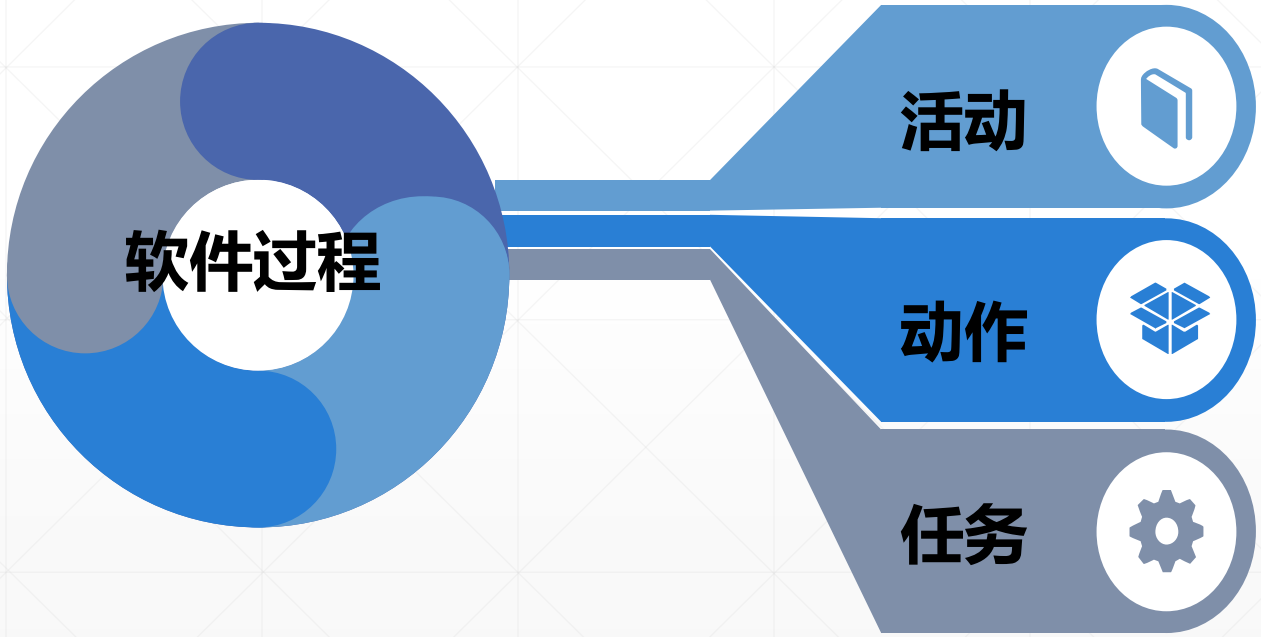
软件产品或软件系统从设计、投入使用到被淘汰的全过程

2.1.1 软件生命周期

项目计划报告	可行性研究报告	需求规格说明书	总体设计说明书	详细设计说明书	源程序	软件测试报告	软件维护说明

2.1.2 软件过程

软件过程是在工作产品构建过程中，所需完成的活动、动作和任务的集合。



活动主要实现宽泛的目标，与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有直接关系。

动作包含了主要工作产品生产过程中的一系列任务。

任务关注小而明确的目标，能够产生实际产品。

2.1.3 软件过程模型

软件过程模型

是软件开发全部过程、活动和任务的结构框架。它能直观表达软件开发全过程，明确规定要完成的主要活动、任务和开发策略。

软件过程模型也常称为

- 软件开发模型
- 软件生存周期模型
- 软件工程范型

2.1.4 软件过程评估

能力成熟度模型

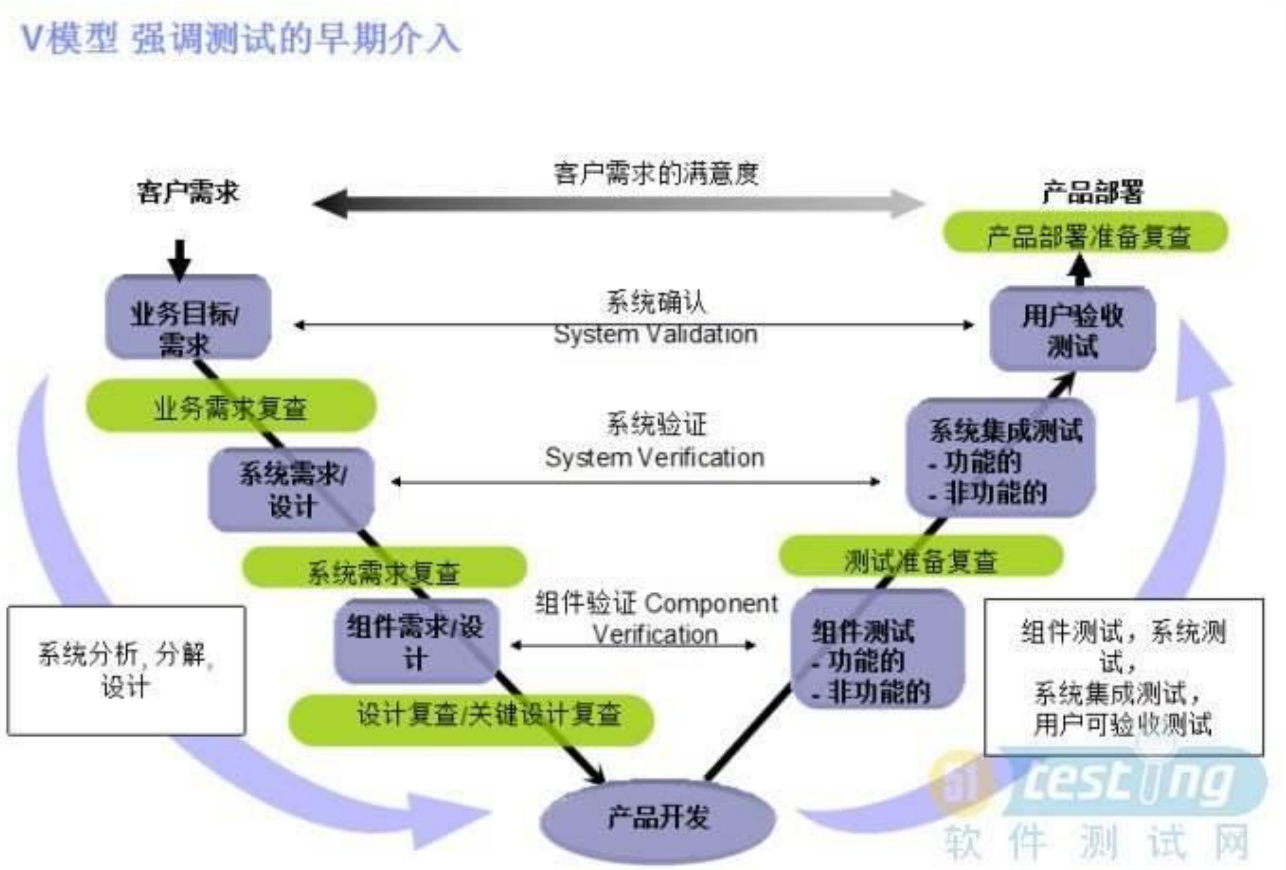
- Capability Maturity Model
- CMU-SEI
- 迄今为止学术界和工业界公认的有关软件工程和管理实践的最好的软件过程评估模型
- 为评估软件组织的生产能力提供了标准
- 为提高软件组织的生产过程指明了方向

2.1.4 能力成熟度模型CMM

- 2.2.1 瀑布模型和V模型
- 2.2.2 原型和增量模型
- 2.2.3 螺旋模型和喷泉模型
- 2.2.4 基于构件的开发模型
- 2.2.5 统一开发模型
- 2.2.6 敏捷开发过程

2.2 常见软件过程模型

介绍各种常见的软件过程模型，以及软件过程实践

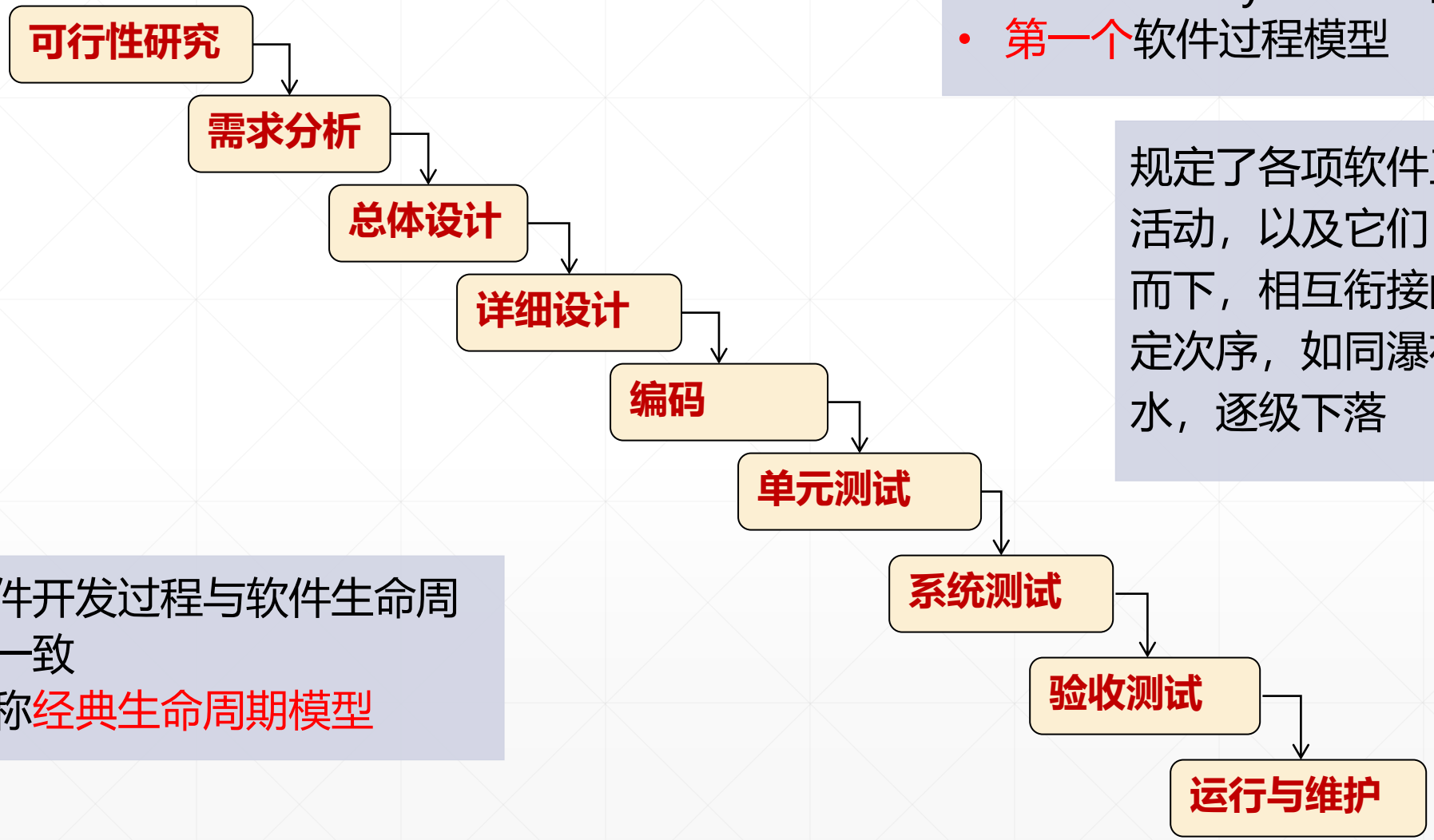


2.2.1 瀑布模型

- 瀑布模型
- V模型

2.2.1-1 瀑布模型（Waterfall model）

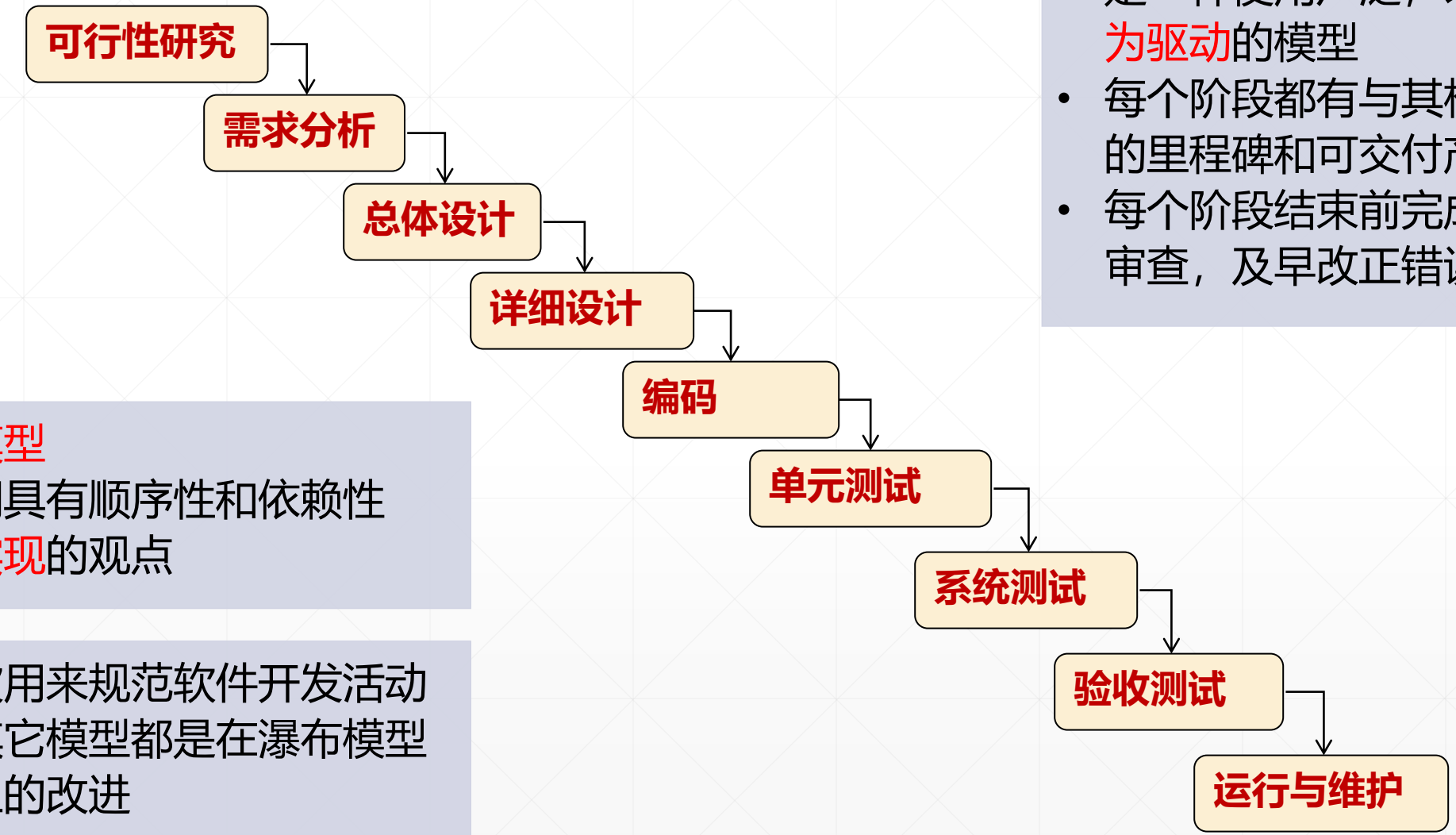
- Winston Royce1970年提出
- 第一个软件过程模型



规定了各项软件工程活动，以及它们自上而下，相互衔接的固定次序，如同瀑布流水，逐级下落

- 软件开发过程与软件生命周期一致
- 也称经典生命周期模型

2.2.1-2 瀑布模型（Waterfall model）

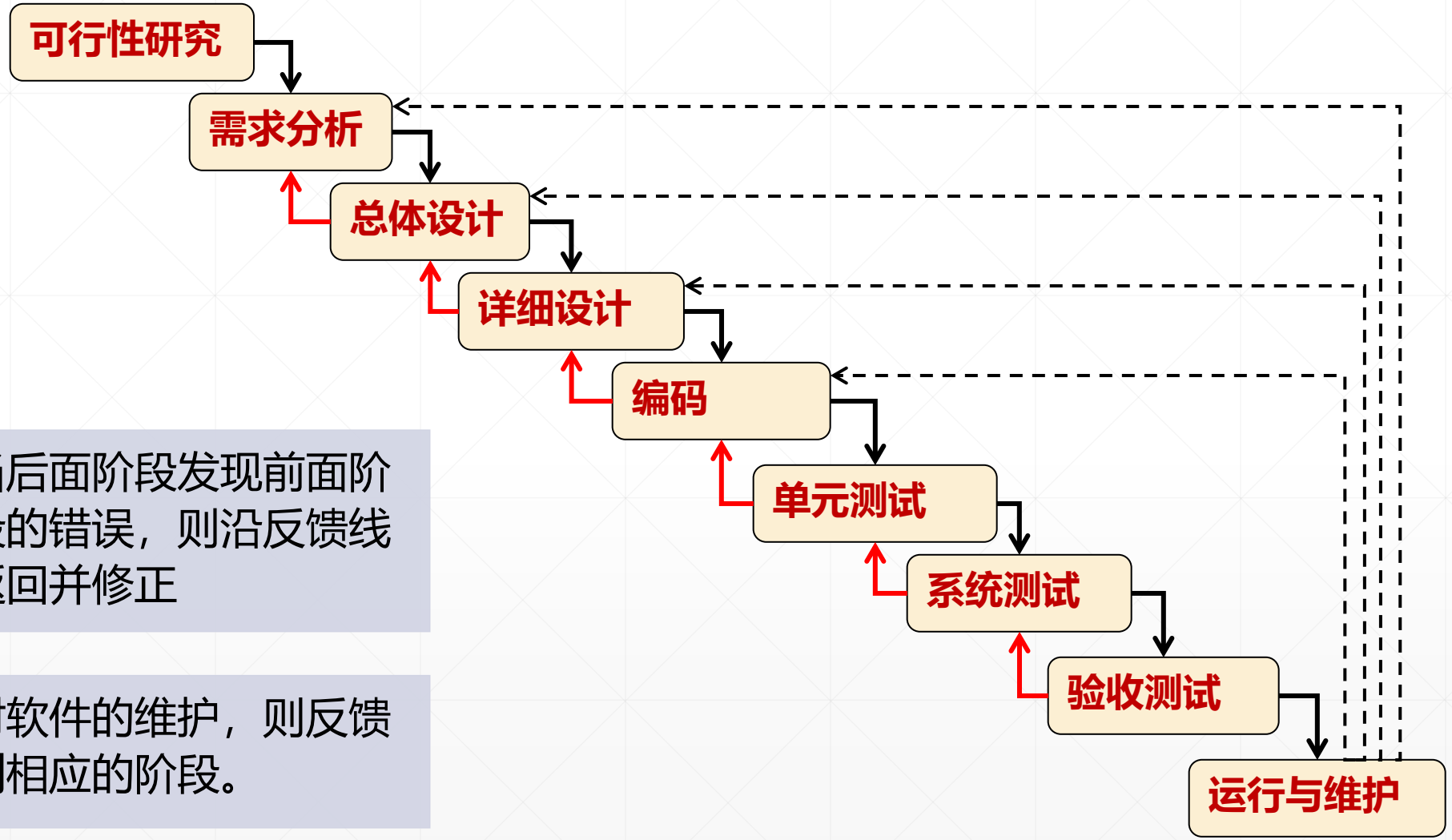


- 是一种使用广泛，以文档为驱动的模型
- 每个阶段都有与其相关联的里程碑和可交付产品
- 每个阶段结束前完成文档审查，及早改正错误

- 线性模型
- 阶段间具有顺序性和依赖性
- 推迟实现的观点

- 一直被用来规范软件开发活动
- 很多其它模型都是在瀑布模型基础上的改进

2.2.1-3 实际（带反馈）的瀑布模型



- 当后面阶段发现前面阶段的错误，则沿反馈线返回并修正

- 对软件的维护，则反馈到相应的阶段。

2.2.1-4 瀑布模型的缺点

增加工作量

各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量

开发风险大

由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发的风险

早期错误发现晚

早期的错误可能要等到开发后期的测试阶段才能发现，进而带来严重的后果

不适应需求变化

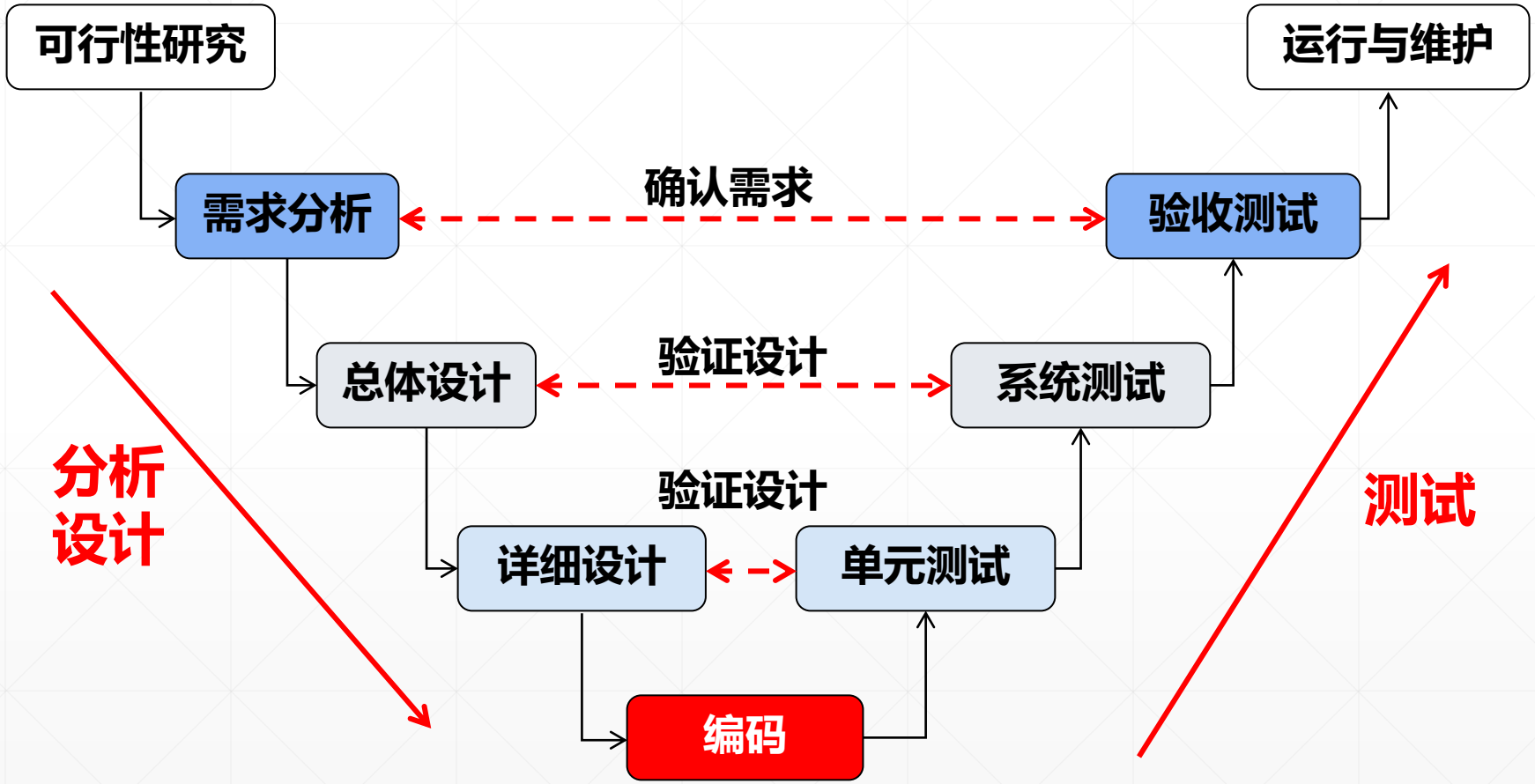
不能反映实际的开发方式，软件开发需要迭代；无法适应需求不明确和需求的变化

2.2.1-5 瀑布模型的适用场合

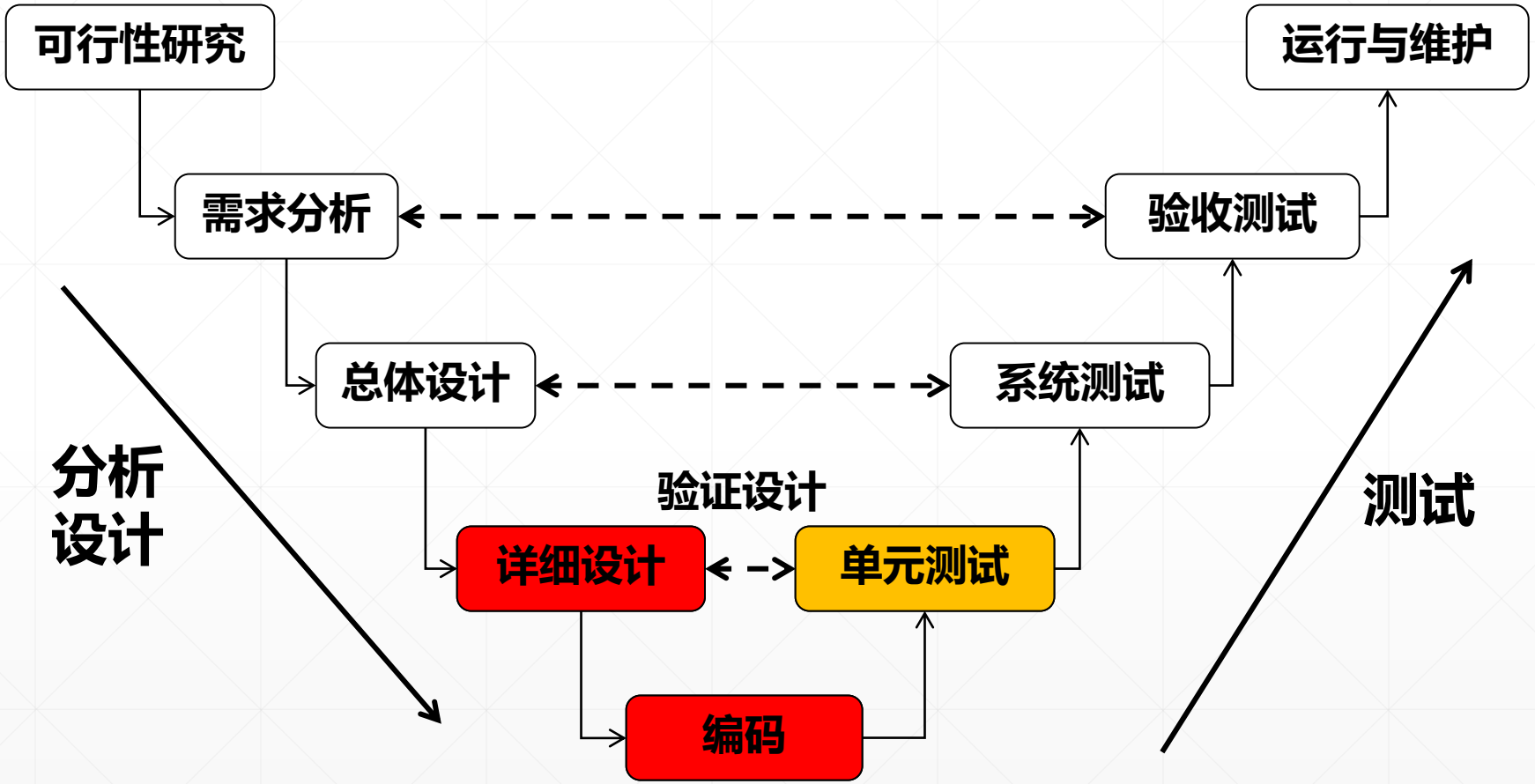
瀑布模型适用于系统需求明确且稳定、技术成熟、工程管理较严格的场合，如军工、航天、医疗。



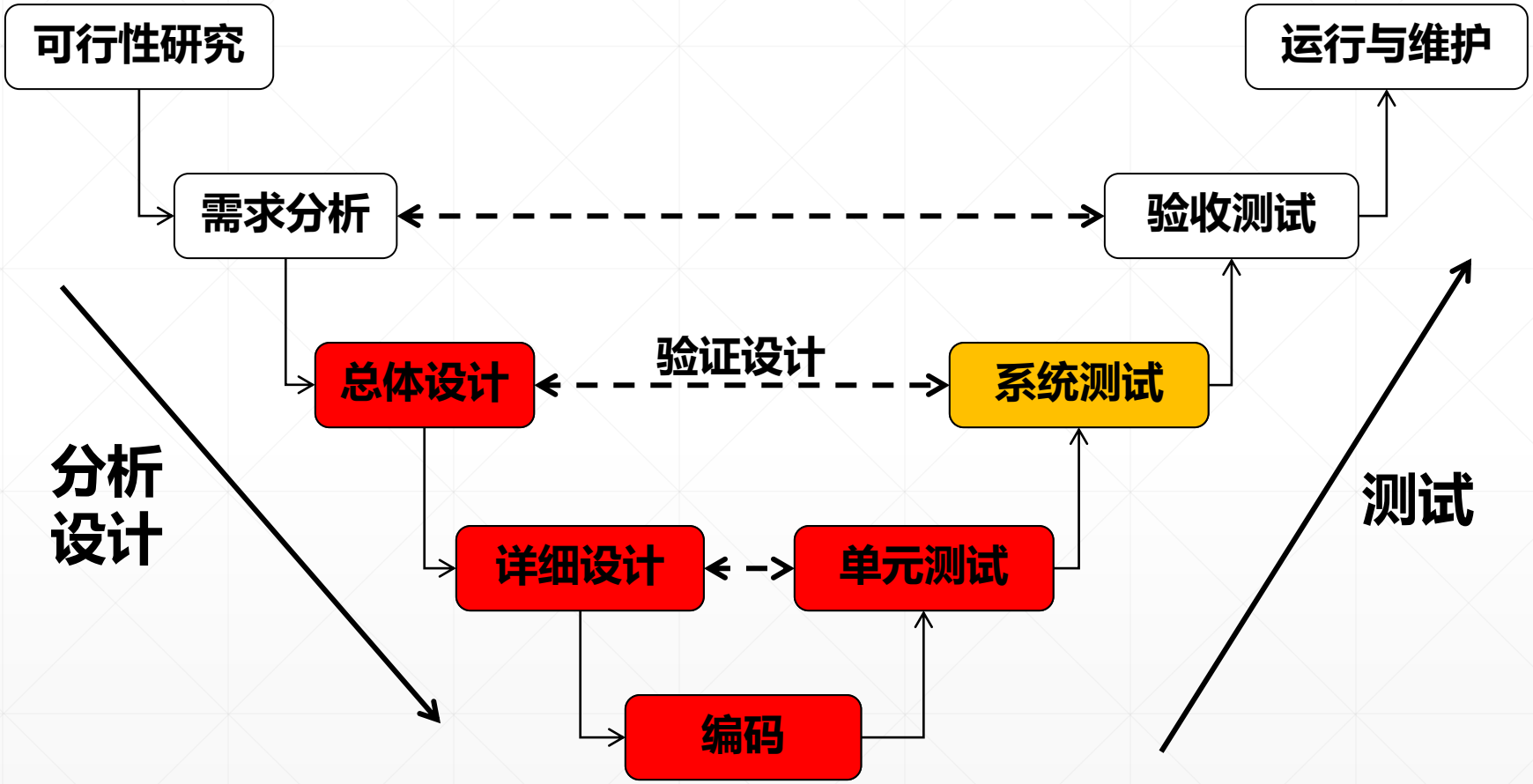

2.2.1-6 V模型（V-model）：瀑布模型的变种



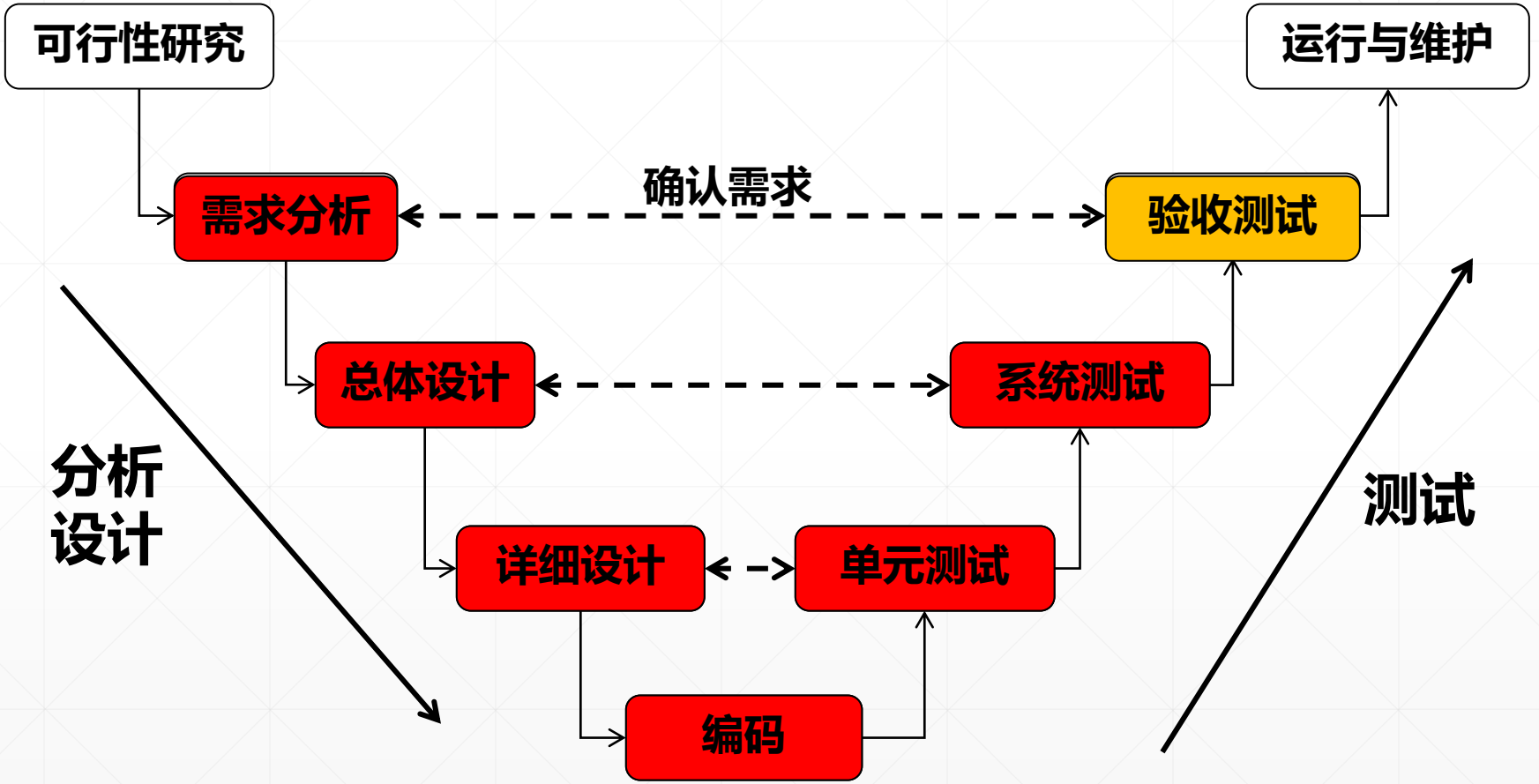
2.2.1-7 V模型：单元测试发现问题

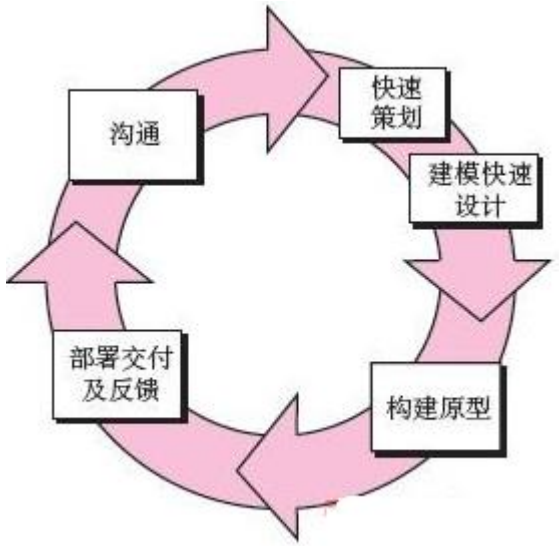


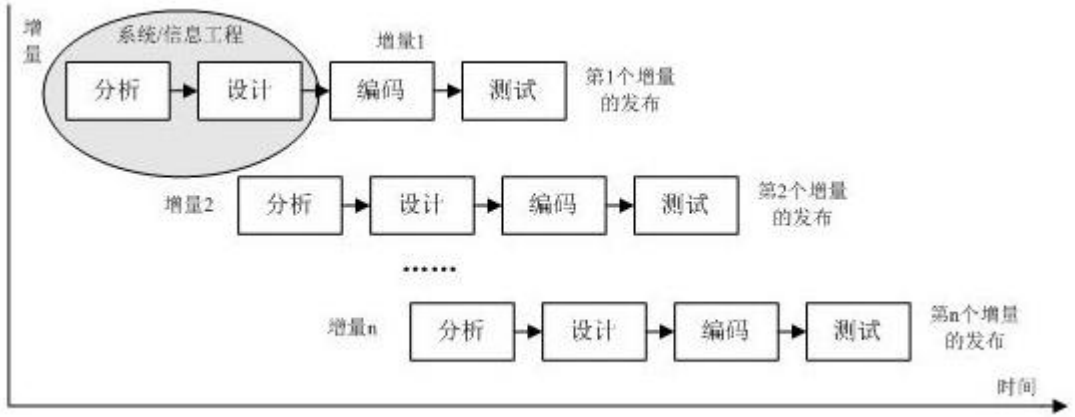
2.2.1-8 V模型：系统测试发现问题



2.2.1-9 V模型：验收测试发现问题







2.2.2 原型模型和增量模型

介绍原型软件过程模型和增量软件过程模型

2.2.2-1 原型模型（Prototype model）

也称为

- 原型化模型、快速原型模型

原型（prototype）

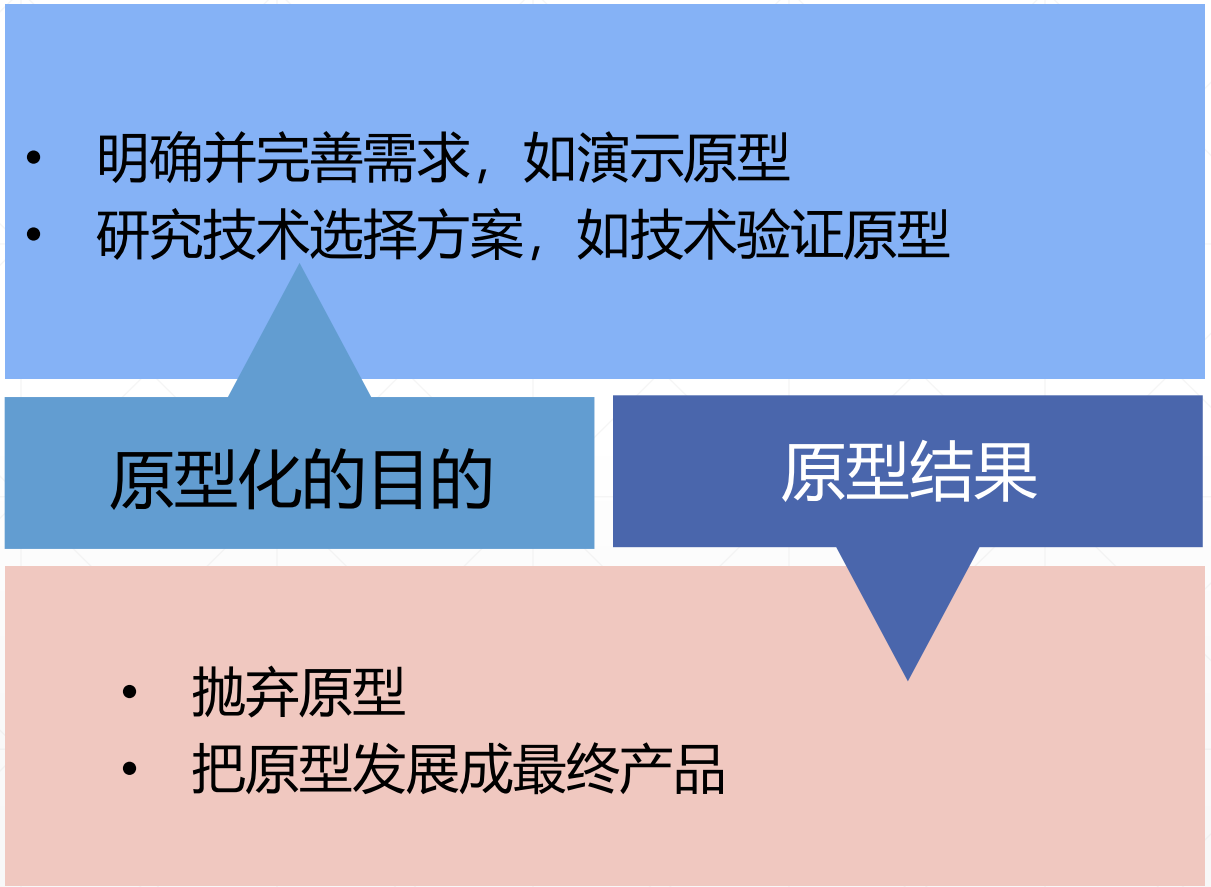
- 一个部分开发的产品，使客户和开发人员能够对计划开发的系统的相关方面进行检查。

举例1

- 图书借阅系统：主要界面

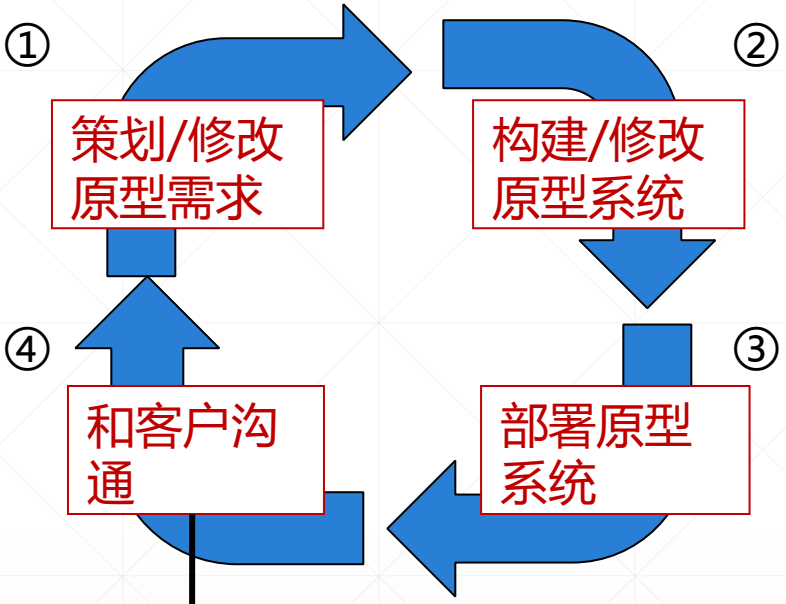
举例2

- 智能家居系统：少量的室内信息监视和电器控制

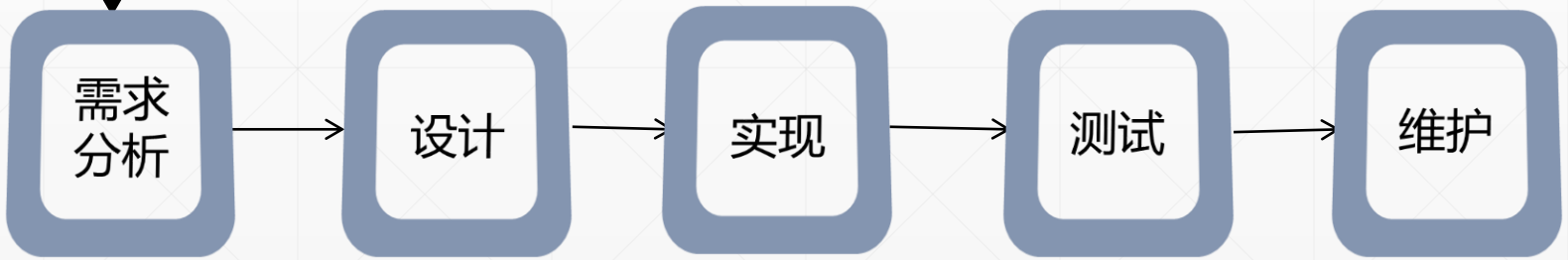


2.2.2-2 原型模型（Prototype model）

原型构建



系统开发



2.2.2-3 原型模型的优缺点

优点

减少需求不明确带来的风险

缺点

- 构造原型采用的技术和工具不一定主流
- 快速建立起来的系统加上连续的修改可能导致原型质量低下
- 设计者在质量和原型中进行折中
- 客户意识不到一些质量问题

2.2.2-4 原型模型的适用场合

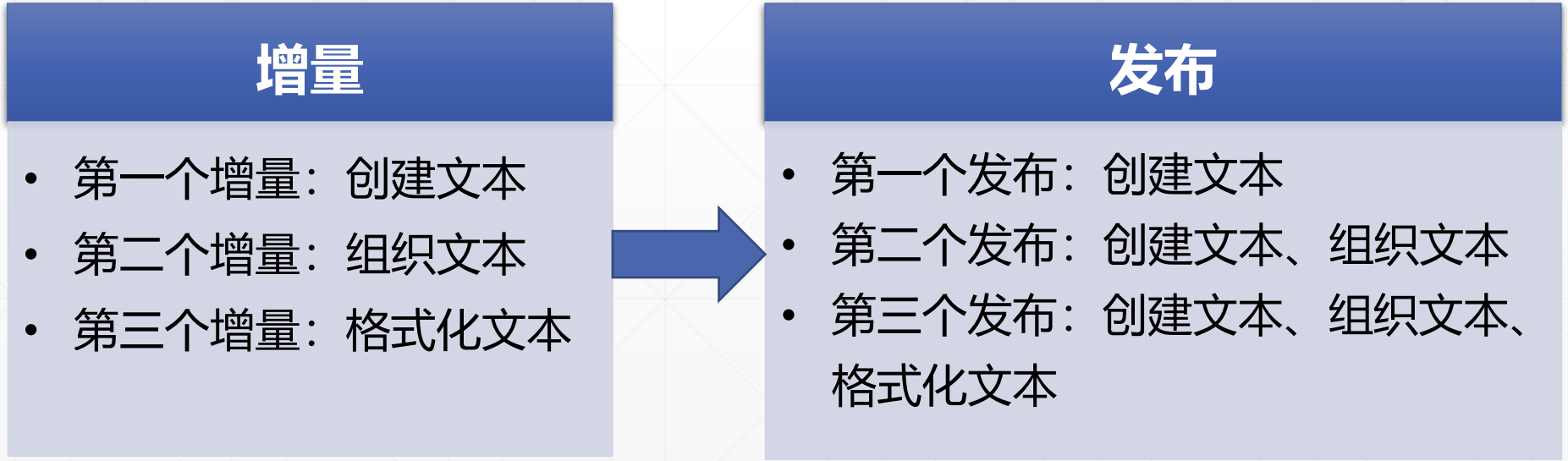
- 客户定义一个总体目标集，但是他们并不清楚系统的具体输入输出；或开发者不确定算法的效率、软件与操作系统是否兼容以及客户与计算机交互的方式。
- 此时，原型模型是很好的选择。



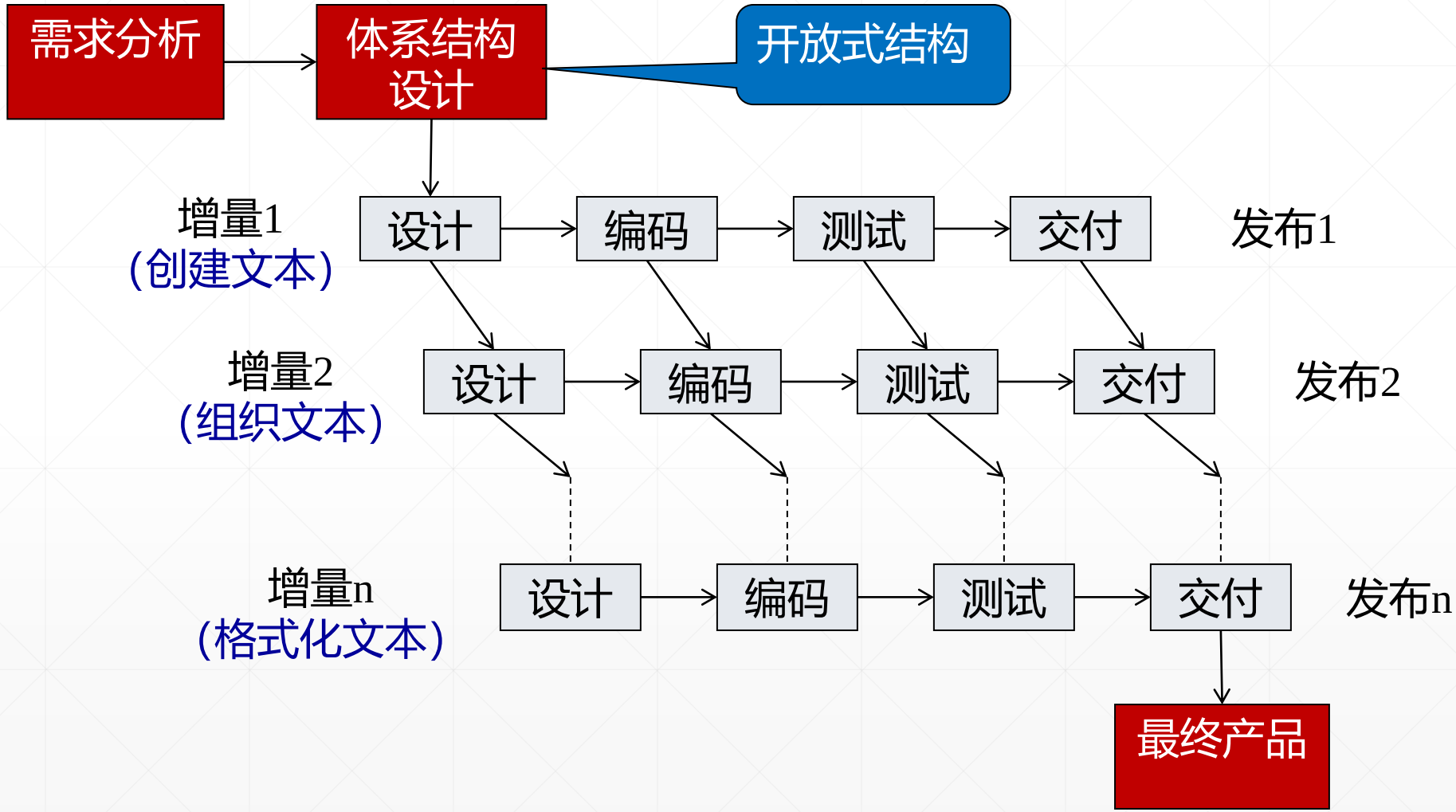
2.2.2-5 增量模型（Incremental model）

- **增量**：满足用户需求的一个子集，能够完成一定功能、小而可用的软件

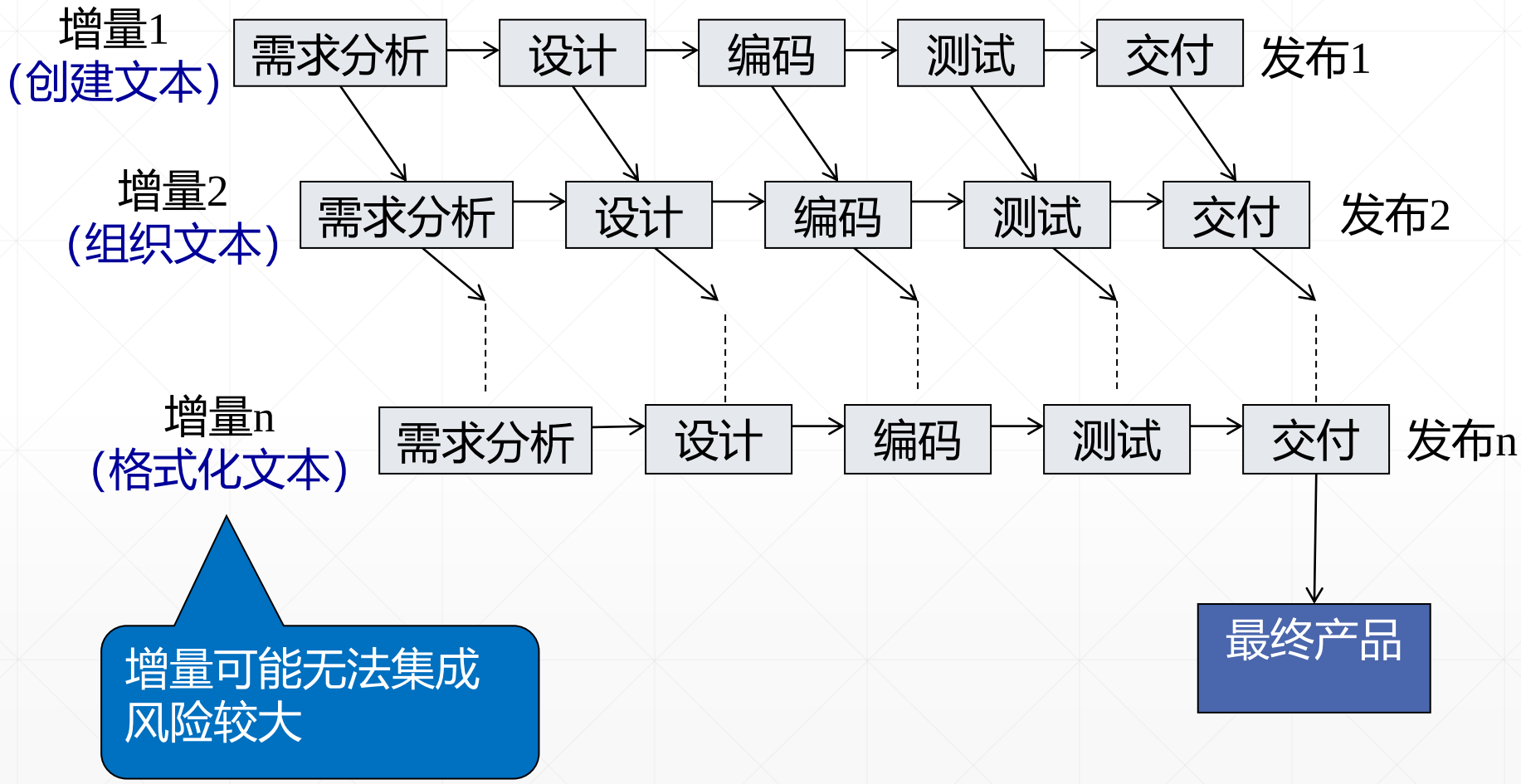
- 举例：
  - 文字处理软件：创建文本、组织文本、格式化文本



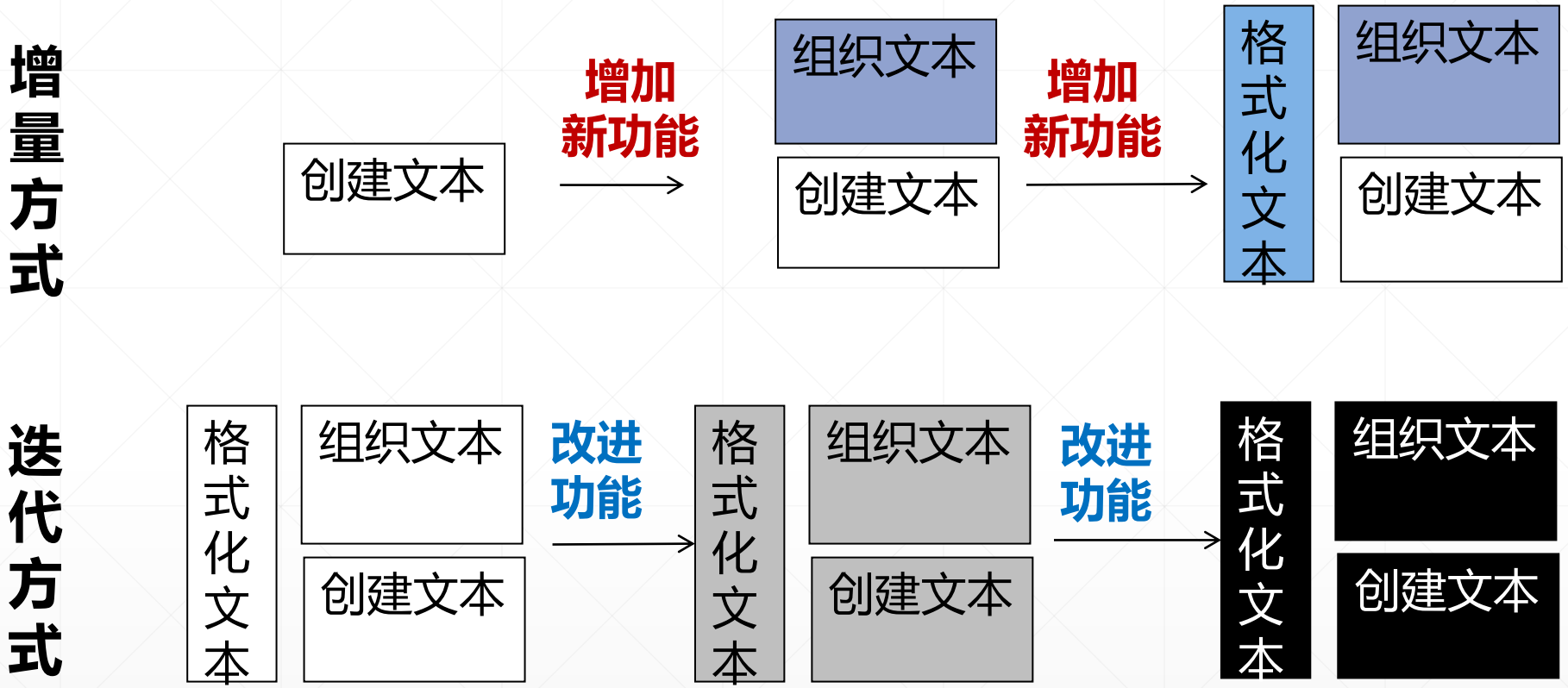
2.2.2-6 增量模型（Incremental model）



2.2.2-7 增量模型（Incremental model）



2.2.2-8 增量的方式



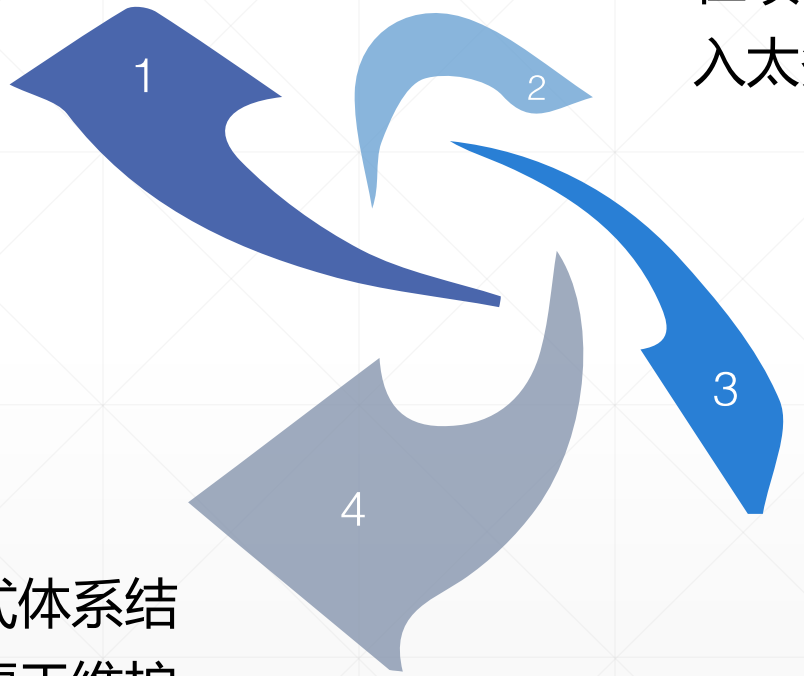
实际使用中，常常是两种方式的结合

2.2.2-9 增量模型的特点

- 增量模型是一种非整体开发的模型，是一种进化式的开发过程
- 增量模型从部分需求出发，先建立一个不完整的系统，通过测试运行这个系统取得经验和反馈，进一步使系统扩充和完善
- 如此反复进行，直至软件人员和用户对所设计的软件系统满意为止
- 增量模型结合了原型模型的基本要素和迭代的特征，采用了基于时间的线性序列，每个线性序列都会输出该软件的一个“增量”
- 每个增量的开发可用瀑布或快速原型模型

2.2.2-10 增量模型的优点

- 增量概念的引入，不需要提供完整的需求，只要有一个增量出现，开发就可以进行；
- 软件能够更早投入市场；



- 在项目的初始阶段不需要投入太多的人力资源；
- 产品逐步交付，软件开发能够较好地适应需求的变化；
- 能够看到软件中间产品，提出改进意见，减少返工，降低开发风险；
- 开放式体系结构，便于维护。

2.2.2-11 增量模型的缺点

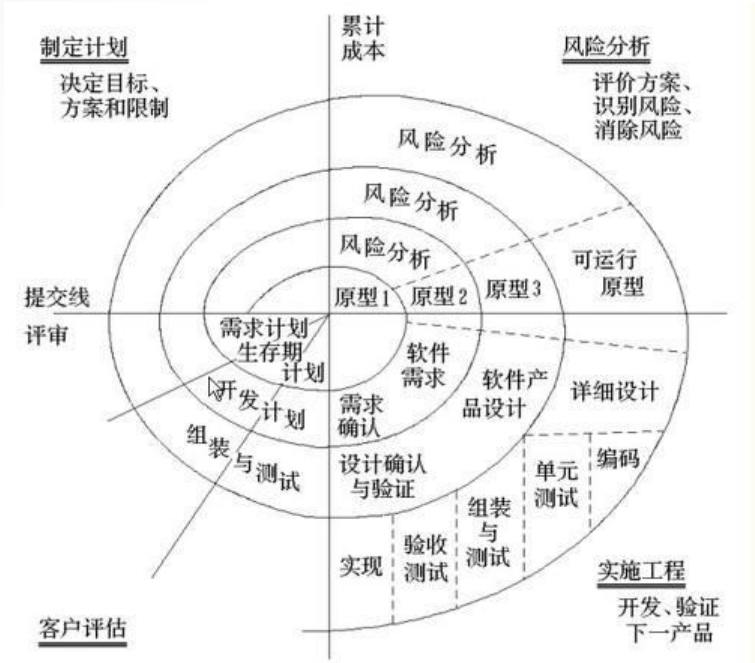
每个增量必须提供一些系统功能，这使得开发者很难根据客户需求给出大小适合的增量

软件必须具备开放式体系结构（困难）

易退化成边做边改的方式，使软件过程控制失去整体性

2.2.2-12 增量模型的适用场合

适用于软件开发中需求可能发生变化、具有较大风险、或者希望尽早进入市场的项目。



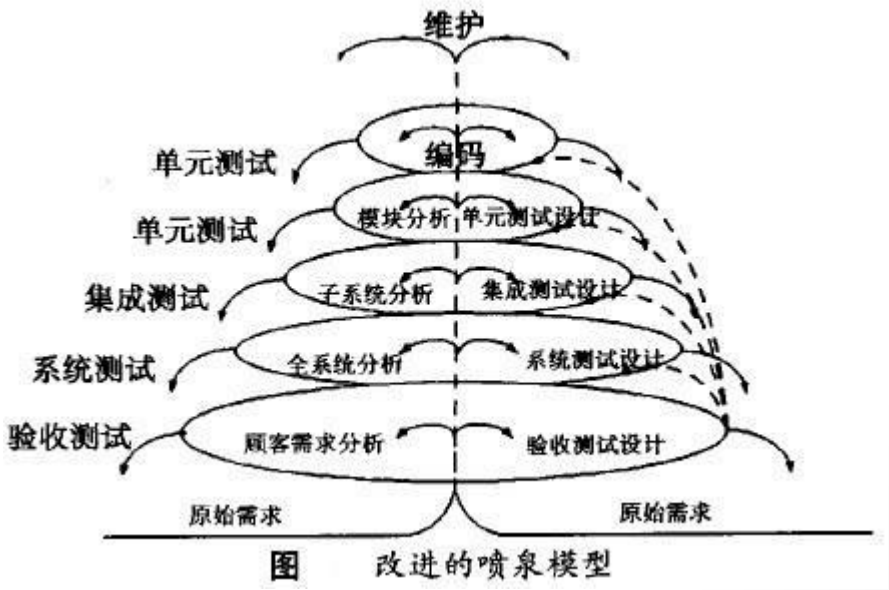


图　改进的喷泉模型

2.2.3 螺旋模型和喷泉模型

介绍两种针对项目风险的软件过程模型

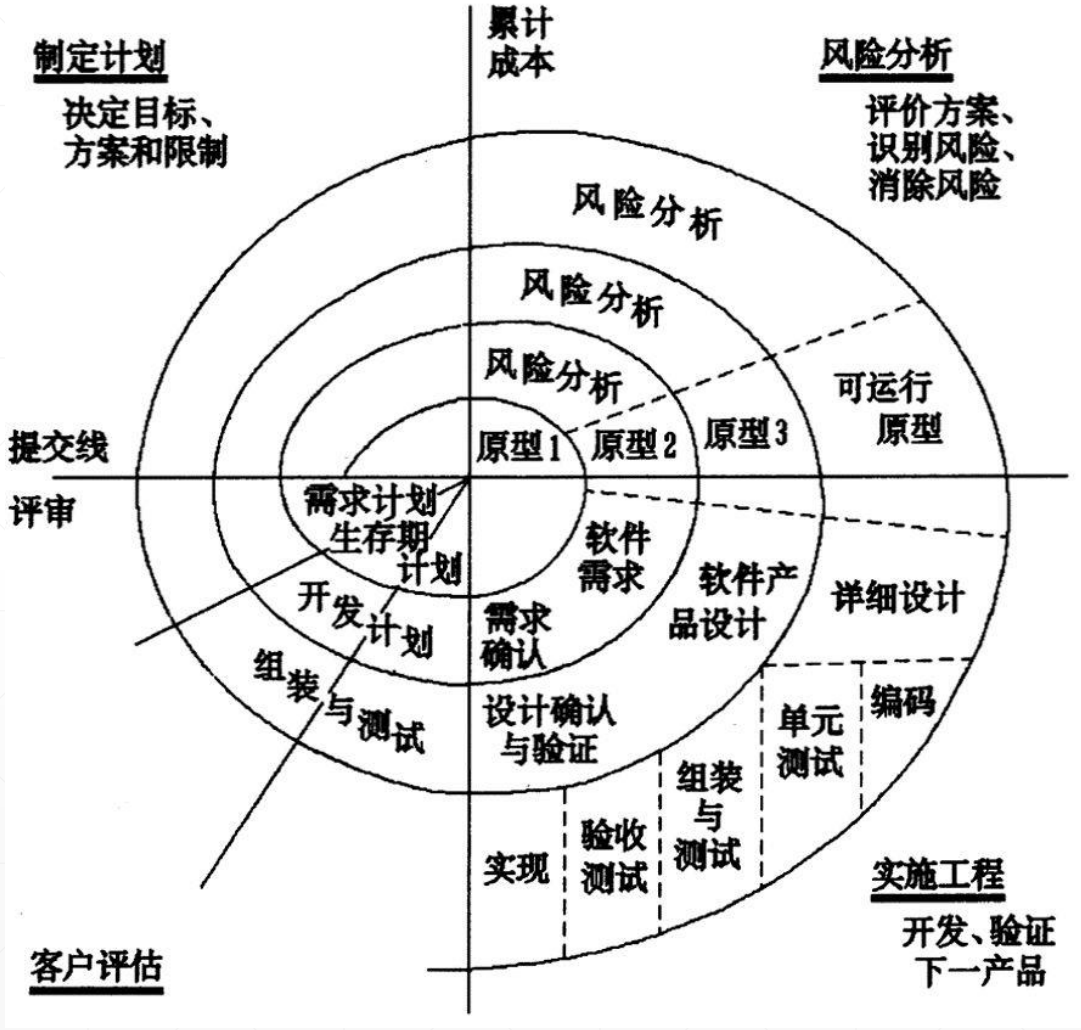
2.2.3-1 螺旋模型（Spiral model）

- 软件开发普遍存在风险
  - 交付的产品用户不满意
  - 产品不能按时交付
  - 开发成本超过预算
  - 产品开发期间关键开发人员离职
  - 产品投入市场前竞争对手发布功能相近价格更低产品

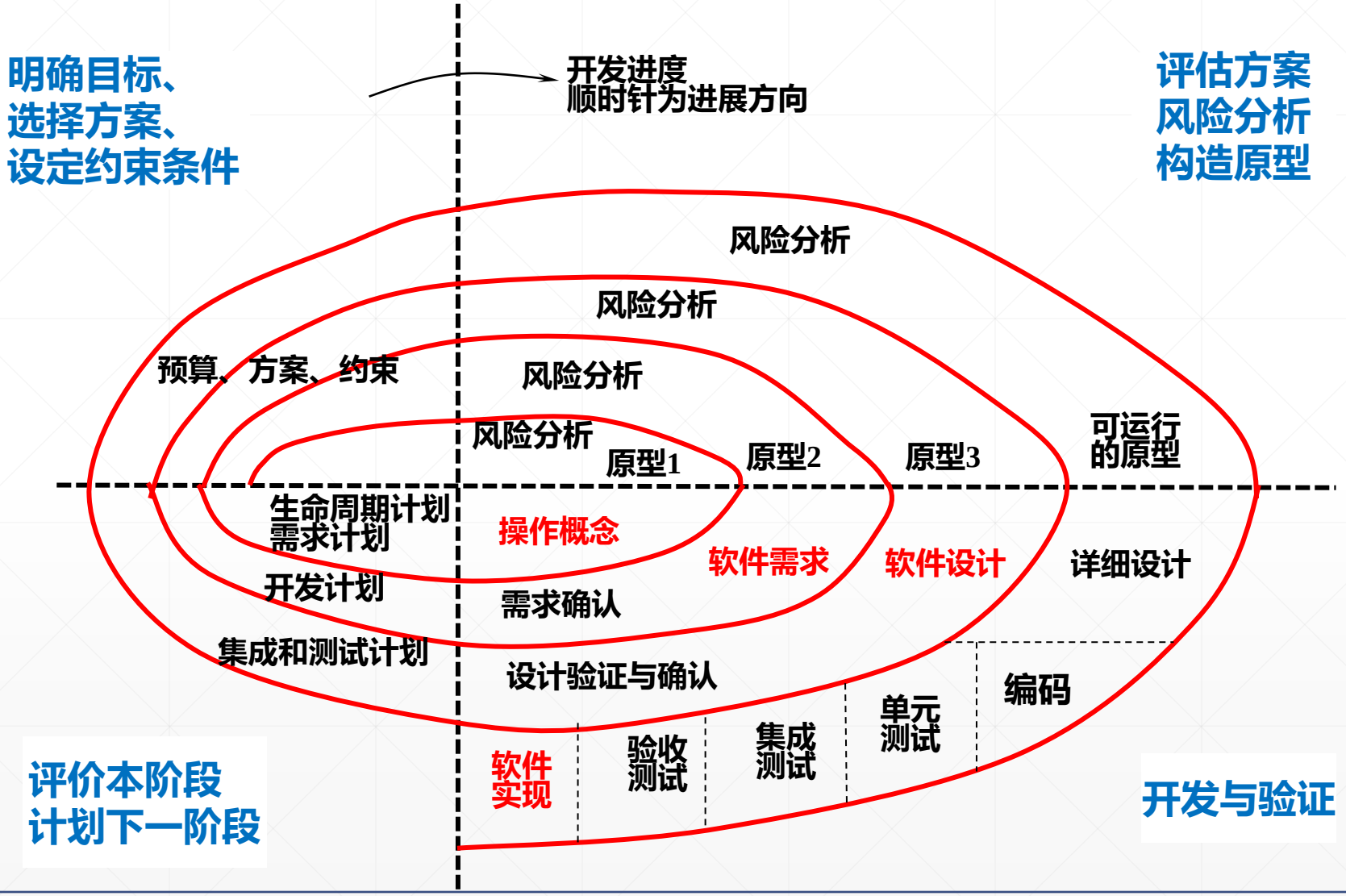
  …
- 把开发活动和风险管理结合起来控制风险

2.2.3-2 螺旋模型（Spiral model）

- 开发过程分成若干次迭代，每次迭代代表开发的一个阶段，对应模型中一条环线
- 每次迭代分成四个方面的活动，对应笛卡尔坐标的四个象限：
  - ① 确定本阶段目标，选定实施方案，弄清项目开发的限制条件；
  - ② 评估所选方案，通过构造原型和风险分析识别和消除风险；
  - ③ 实施软件开发和验证；
  - ④ 评价本阶段的工作成果，提出修正建议，并计划下一阶段工作。
- 模型结合了瀑布模型和原型模型的特点

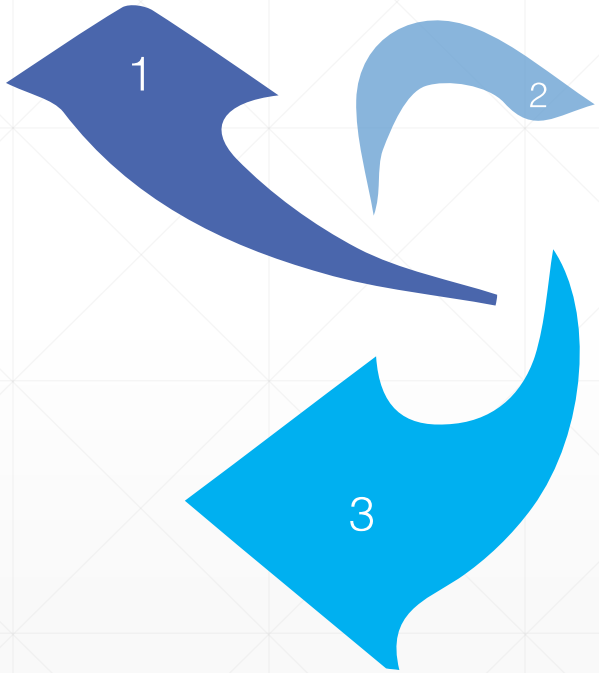


2.2.3-3 螺旋模型（Spiral model）



2.2.3-4 螺旋模型的优点

螺旋模型强调原型的可扩充性和可修改性，原型的进化贯穿整个软件生存周期，这将有助于目标软件的适应能力，支持用户需求的动态变化；



原型可看作可执行的需求规格说明，易于为用户和开发人员共同理解，还可作为继续开发的基础，并为用户参与所有关键决策提供了方便；

螺旋模型为项目管理人员及时调整管理决策提供了方便，进而可降低开发风险。

2.2.3-5 螺旋模型的缺点

如果每次迭代的效率不高，致使迭代次数过多，将会增加成本并推迟交付时间；

使用该模型需要有相当丰富的风险评估经验和专门知识，要求开发队伍水平较高，否则会带来更大风险。

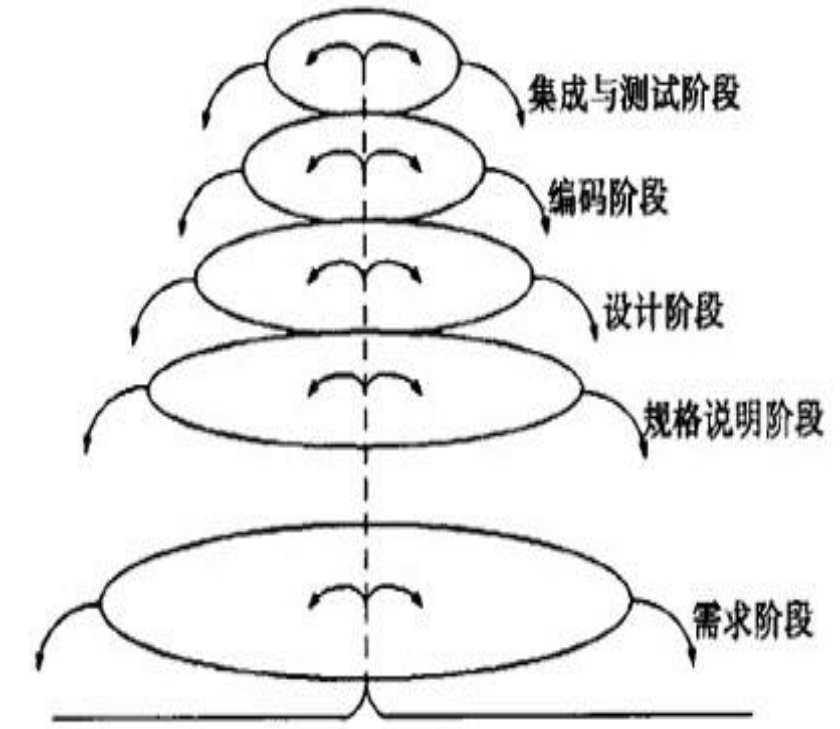
2.2.3-6 螺旋模型的适用场合

适用于需求不明确或者需求可能发生变化的大型复杂的软件系统。

支持面向过程、面向对象等多种软件开发方法，是一种具有广阔前景的模型。

2.2.3-7 喷泉模型（Fountain model）

- 喷泉模型是一种以用户需求为动力，以对象为驱动的模型，主要用于描述面向对象的软件开发过程
- 软件开发早期定义对象，整个开发过程充实和扩充对象
- 各个阶段使用统一的概念和表示方法，生命周期各阶段无缝连接
- 各个开发步骤多次反复迭代



2.2.3-8 喷泉模型的优缺点及适用场合

优点

喷泉模型的各个阶段没有明显的界限，开发人员可以同步进行开发，可以提高软件项目开发效率，节省开发时间，适应于面向对象的软件开发过程。

缺点

由于喷泉模型在各个开发阶段是重叠的，在开发过程中需要大量的开发人员，因此不利于项目的管理。

喷泉模型要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入的各种信息、需求与资料的情况。

适用场合

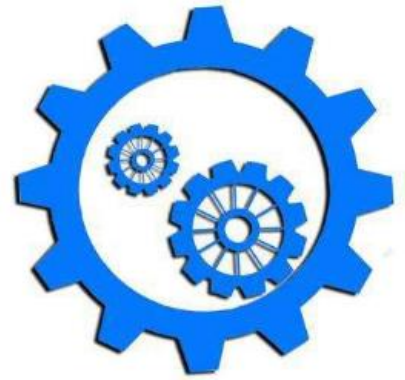
适用于面向对象开发

2.2.4 基于构件的开发模型

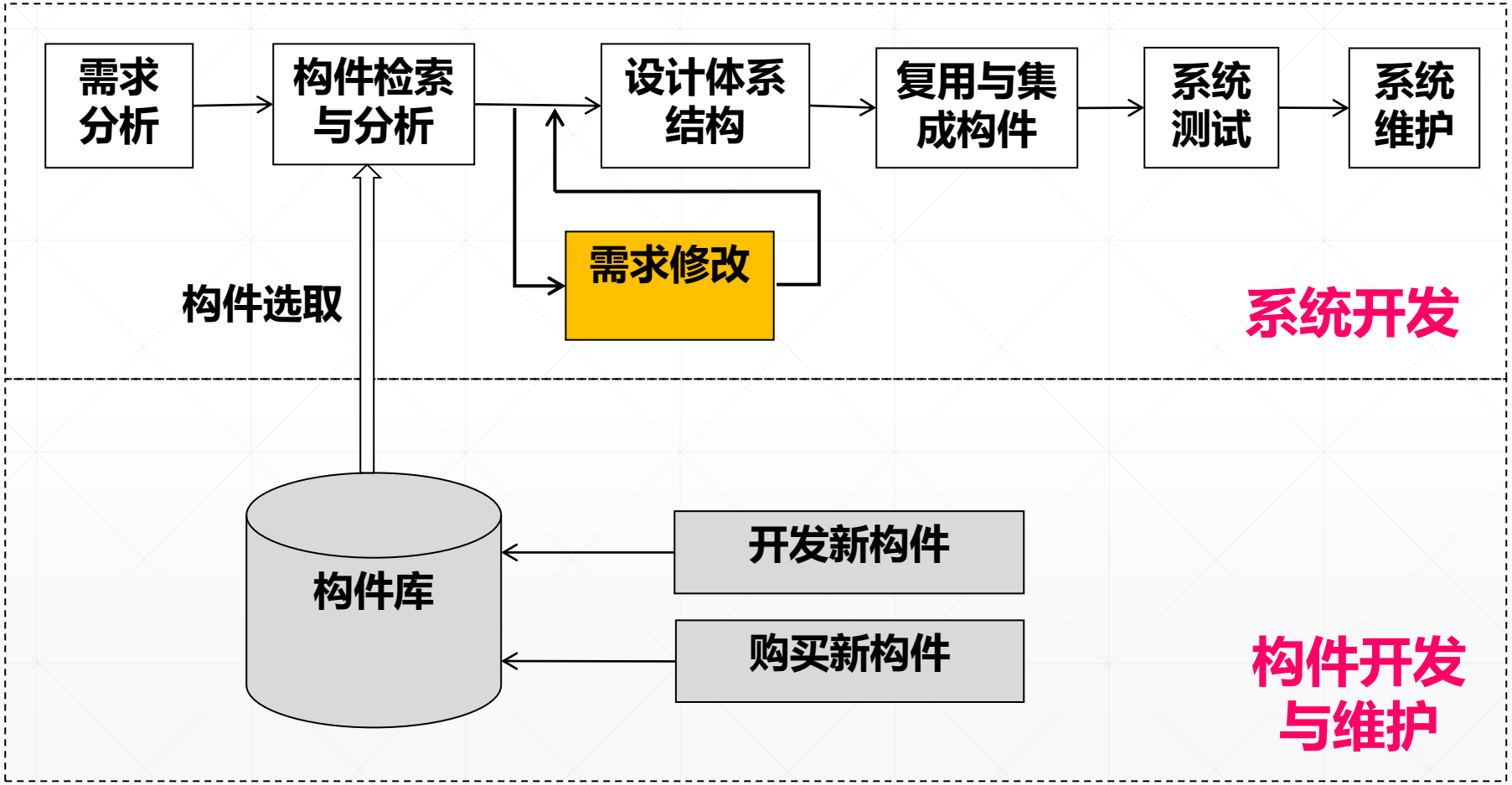
- 这是一种专用的软件过程，需要使用构件开发方法

2.2.4-1 基于构件的开发模型

- Component-based development model
- 近年来得到广泛应用，改变大型软件开发方式
- 考虑的焦点是集成，而非实现
- 构件/组件（Component）
  - 系统中模块化的、可更换的部分
  - 实现特定的功能
  - 对实现进行封装，暴露一组接口
  - 例如：动态链接库（.dll），浏览器插件



2.2.4-2 基于构件的开发模型



2.2.4-3 基于构件的开发模型

01. 需求分析

- 与其它过程模型相同

02. 构件分析

- 根据需求搜索构件
- 如果没有完全匹配的构件，则需要修改构件或者修改需求

03. 系统设计

- 与其它过程模型不同
- 考虑重用和集成
- 如果没有可重用的构件，则设计新软件

04. 开发集成

- 将构件集成到系统中
- 开发新软件

2.2.4-4 基于构件的开发模型的优缺点

优点

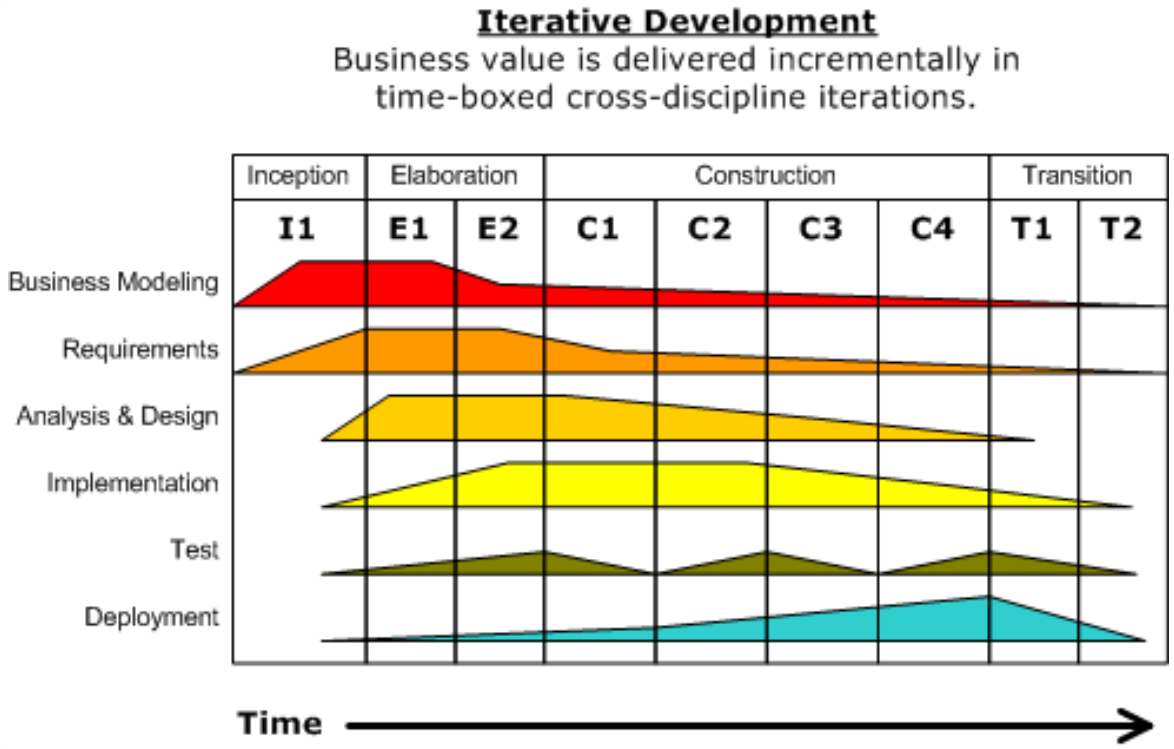
- 软件复用思想
- 降低开发成本和风险，加快开发进度，提高软件质量

缺点

- 模型复杂
- 商业构件不能修改，会导致修改需求，进而导致系统不能完全符合客户需求
- 无法完全控制所开发系统的演化
- 项目划分的好坏直接影响项目结果的好坏

适用场合

适用于系统之间有共性的情况。



2.4.5 统一过程模型

- 基于面向对象分析和设计方法的软件过程

2.4.5-1 Rational统一过程模型

- Rational Unified Process - RUP
- 由Rational公司（现已被IBM收购）推出的完整且完美的软件工程方法
- 获得广泛使用
- 基于面向对象方法学
- 使用统一建模语言UML（Unified Modeling Language）

2.4.5-2 Rational统一过程模型

- 从3个视角描述软件开发过程
  - **动态视角**：随时间变化的各个阶段
  - **静态视角**：所进行的活动
  - **实践视角**：可采用的良好实践建议

适合大团队
大项目

2.4.5-3 Rational统一过程模型

实践视角：

6条最佳实践

1. 迭代式开发

- 需求变更不可避免
- 每次迭代产生一个可交付版本，用户反馈，减少风险
- 根据客户的轻重缓急来规划增量，先开发和交付优先级最高的增量

2. 管理需求

- 采用用例分析来捕获需求，由用例驱动设计和实现
- 对需求及其变更进行管理

2.4.5-4 Rational统一过程模型

实践视角：

6条最佳实践

3. 基于构件体系结构

- 采用基于构件的体系结构
- 提高软件复用率

4. 可视化建模

- 使用统一建模语言（UML）对系统进行可视化建模

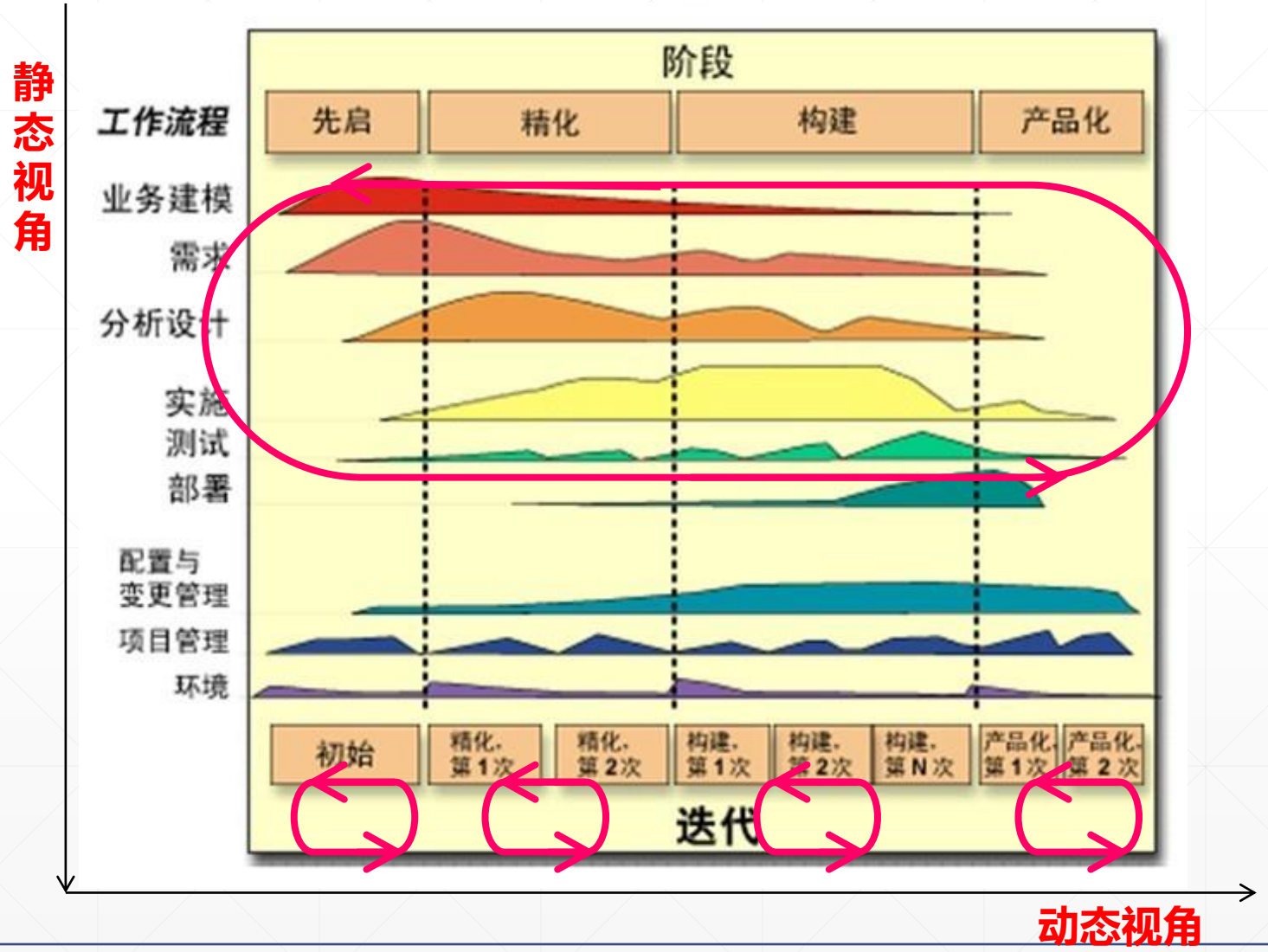
5. 验证软件质量

- 软件质量评估贯穿整个开发过程的所有活动
- 全体成员参与

6. 控制软件变更

- 描述如何控制和跟踪软件的变更

2.4.5-5 Rational统一过程模型



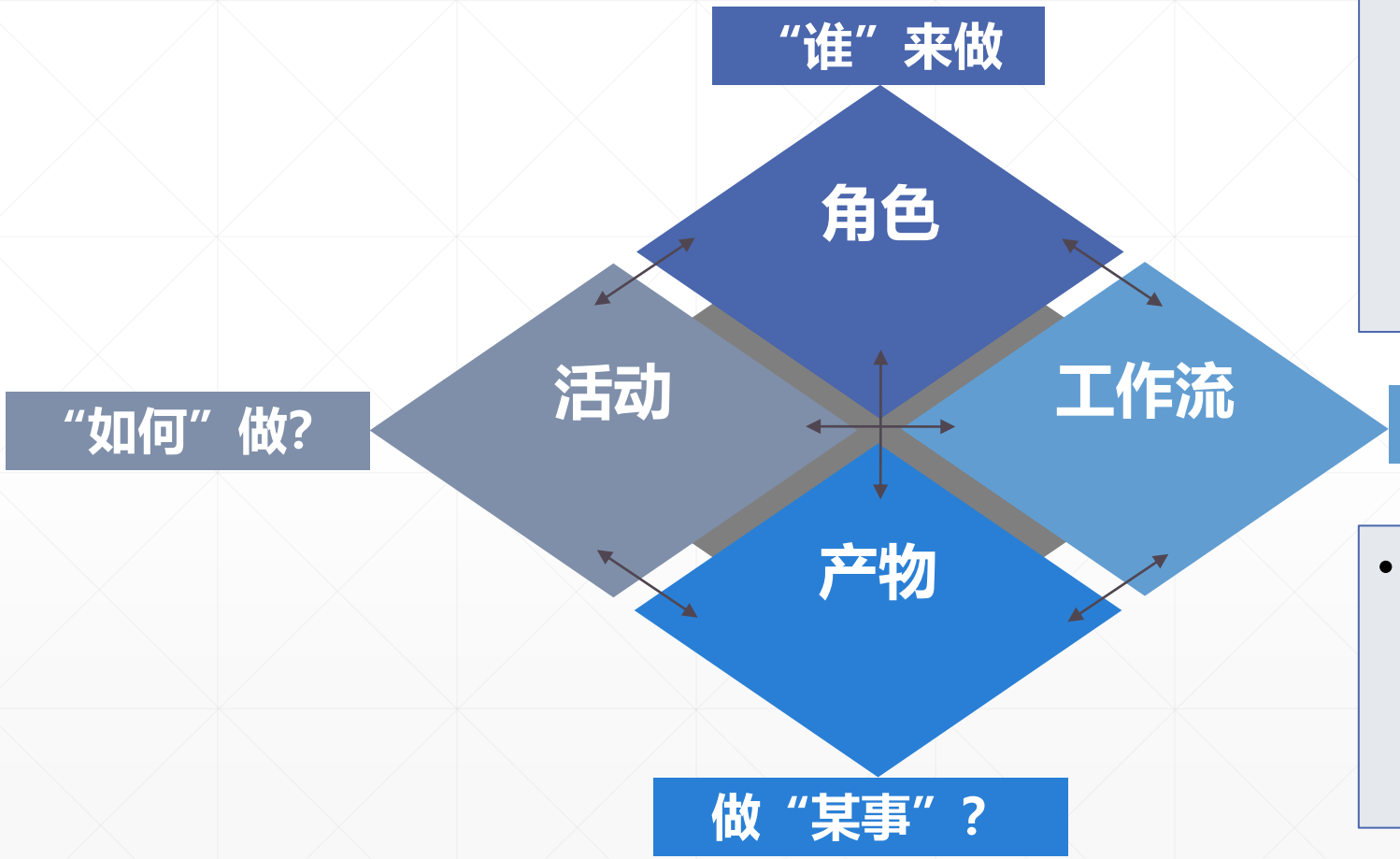
初始：项目计划、评估风险；

精化：设计系统的体系结构、制定项目计划、确定资源需求；

构建：开发出所有组件和应用程序，集成并进行详尽测试；

产品化：将产品移交给用户。

2.4.5-6 统一过程的静态结构



- 6个核心工程工作流:
  - 业务建模工作流
  - 需求工作流
  - 分析设计工作流
  - 实现工作流
  - 测试工作流
  - 部署工作流

- 3个核心支持工作流:
  - 项目管理工作流
  - 配置与变更管理工作流
  - 环境工作流。

2.4.5-7 统一过程的动态结构

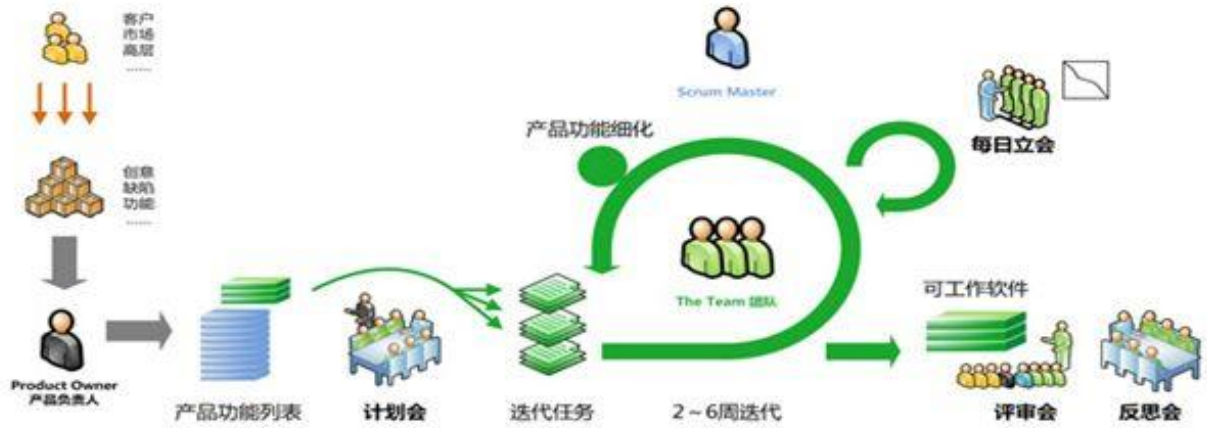
精化阶段

构建阶段

初始阶段

产品化阶段

2.4.6 敏捷开发过程


客户
市场
高层
创意
缺陷
功能
Product Owner
产品负责人
产品功能列表
计划会
Scrum Master
产品功能细化
The Team 团队
迭代任务
2～6周迭代
每日立会
可工作软件
评审会
反思会

2.4.6-1 敏捷软件开发

- Agile software development
- 2001年2月，17位编程大师发表敏捷软件开发宣言

01. 个体交互	02. 可工作软件	03. 客户合作	04. 响应变化
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

- 虽然右边的项有价值，但我们更重视左边的项
- 高效工作、快速响应变化

2.4.6-2 敏捷开发方法

- 极限编程：eXtreme Programming/XP
- 自适应软件开发
  Adaptive Software Development/ASD
- 并列争球法：Scrum
- 动态系统开发方法
  Dynamic System Development Method/DSDM
- 水晶法：Crystal
- 特征驱动开发：Feature-Driven Development/FDD
- 精益软件开发：Lean Software Development/LSD
- …

2.4.6-3 敏捷软件开发

- 敏捷软件过程是**基本原理**和**开发准则**的结合

基本原理强调

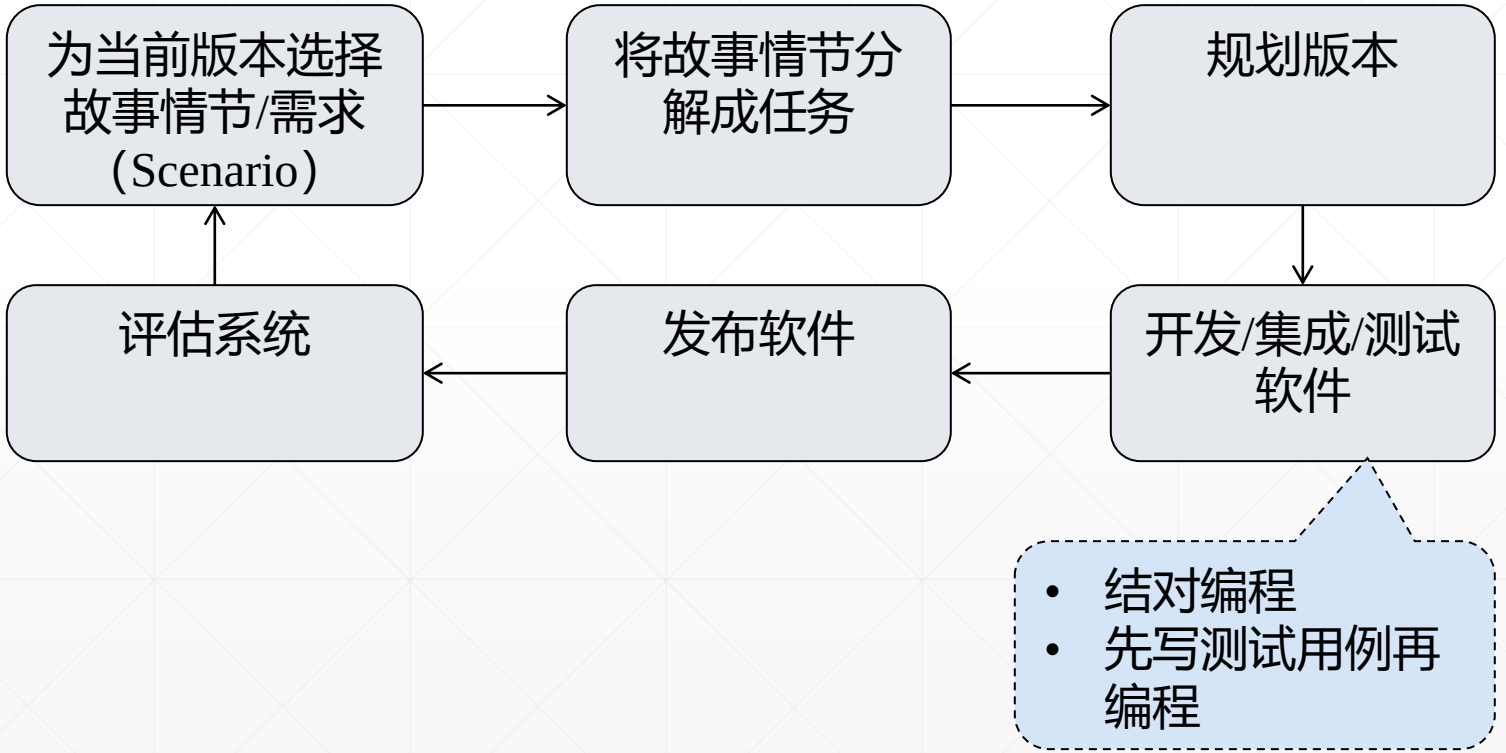
- 客户满意度和较早的软件增量交付
- 小但有激情的团队
- 非正式的方法
- 最小的软件工程产品
- 简化整体开发

开发准则强调

- 超越分析和设计的交付
- 开发者和客户之间积极持续的交流

2.4.6-4 极限编程

- eXtreme Programming - XP
- 把好的开发实践运用到极致



2.4.6-5 极限编程的有效实践

- 增量式开发
- 小版本短周期交付
- 结对编程
- 代码集体所有
- 开放的工作空间
- 可持续的开发速度：<40小时/周，连续加班不超过两周
- 简单的设计
- 测试驱动开发
- 持续集成
- 重构
- 及时调整计划
- 客户作为开发团队成员

2.4.6-6 敏捷开发的优缺点

优点

- 快速响应变化和不确定性
- 可持续开发速度
- 适应商业竞争环境下的有限资源和有限时间

缺点

- 测试驱动开发可能导致通过测试但非用户期望
- 重构而不降低质量困难

适用场合

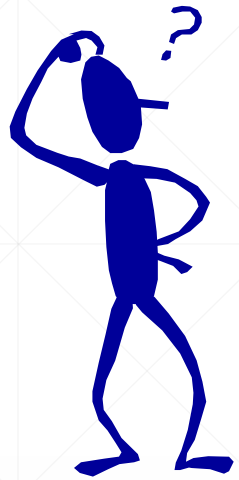
适用于需求模糊且经常改变的场合，适合商业竞争环境下的项目。

2.3 选择过程模型

- 如何根据项目的实践情况，选择软件过程和裁剪软件过程

2.3.1 如何选择软件过程模型

- 软件过程模型是不断发展的
- 各种软件过程模型各有优缺点和适用场合
- 不同软件往往需要不同软件过程模型
- 选用时不必拘泥于某种模型
- 可组合多种模型
- 可根据实际创造新的模型



2.3.1 如何选择软件过程模型

1. 前期需求明确的情况下，尽量采用瀑布模型
2. 用户无系统使用经验，需求分析人员技能不足的情况下，尽量借助原型模型
3. 不确定因素很多，很多东西无法提前计划的情况下，尽量采用增量模型或螺旋模型
4. 需求不稳定的情况下，尽量采用增量模型
5. 资金和成本无法一次到位的情况下，可采用增量模型
6. 对于完成多个独立功能开发的情况，可在需求分析阶段就进行功能并行，每个功能内部都尽量遵循瀑布模型
7. 全新系统的开发必须在总体设计完成后再开始增量或并行
8. 编码人员经验较少的情况下，尽量不要采用敏捷或迭代模型
9. 增量、迭代和原型可以综合使用，但每一次增量或迭代都必须有明确的交付和出口原则

2.3.2 案例1：医疗设备控制软件

- 案例分析：
  - 需求明确且稳定
  - 可靠性和安全性要求极高
  - 对软件错误和故障的控制和跟踪能力强
  - 需要对软件开发过程严格控制
  - 需要大量严格的文档

- 模型选择：瀑布模型

2.3.3 案例2：校园一卡通系统

- 案例分析：
  - 包括若干相对独立的功能
  - 系统具体需求不明确且会发生变化
  - 系统需要具有可扩充性
  - 用户需要熟悉和适应新的系统
  - 项目复杂程度中等、有一定风险
  - 产品和文档的再使用率较高

- 模型选择：增量模型

2.3.4 案例3：智能化小区

- 智能家庭
  - 家居信息的实时和远程监视
  - 家用电器的远程和自动控制
  - 家庭安防报警和远程通知
- 智能小区
  - 安防门禁、可视对讲等
  - 物业管理
  - 一卡通系统
  - 缴费、包裹、公告、便民信息等发布到户
  - 家政相关服务，如送水、送餐等

2.3.4 案例3：智能化小区

- 案例分析：
  - 包括若干相对独立的业务管理功能
  - 系统具体需求不明确且会发生变化
  - 部分技术方案可行性不确定
  - 系统需要具有可扩充性
  - 用户需要熟悉和适应新的系统
  - 项目复杂程度较大、风险较大
  - 希望尽早投入市场

- 模型选择：原型化模型+增量模型

感谢观看！

授课教师：邱元杰　　电子邮箱：yuanjiq@126.com