第二章 软件过程模型

周 瑞

QQ群：235458708

信息与软件工程学院

本章学习目标

1. 了解软件过程和软件过程模型的概念
2. 理解不同模型的特征和优缺点
3. 能依据实际项目选择使用不同的模型

本章内容

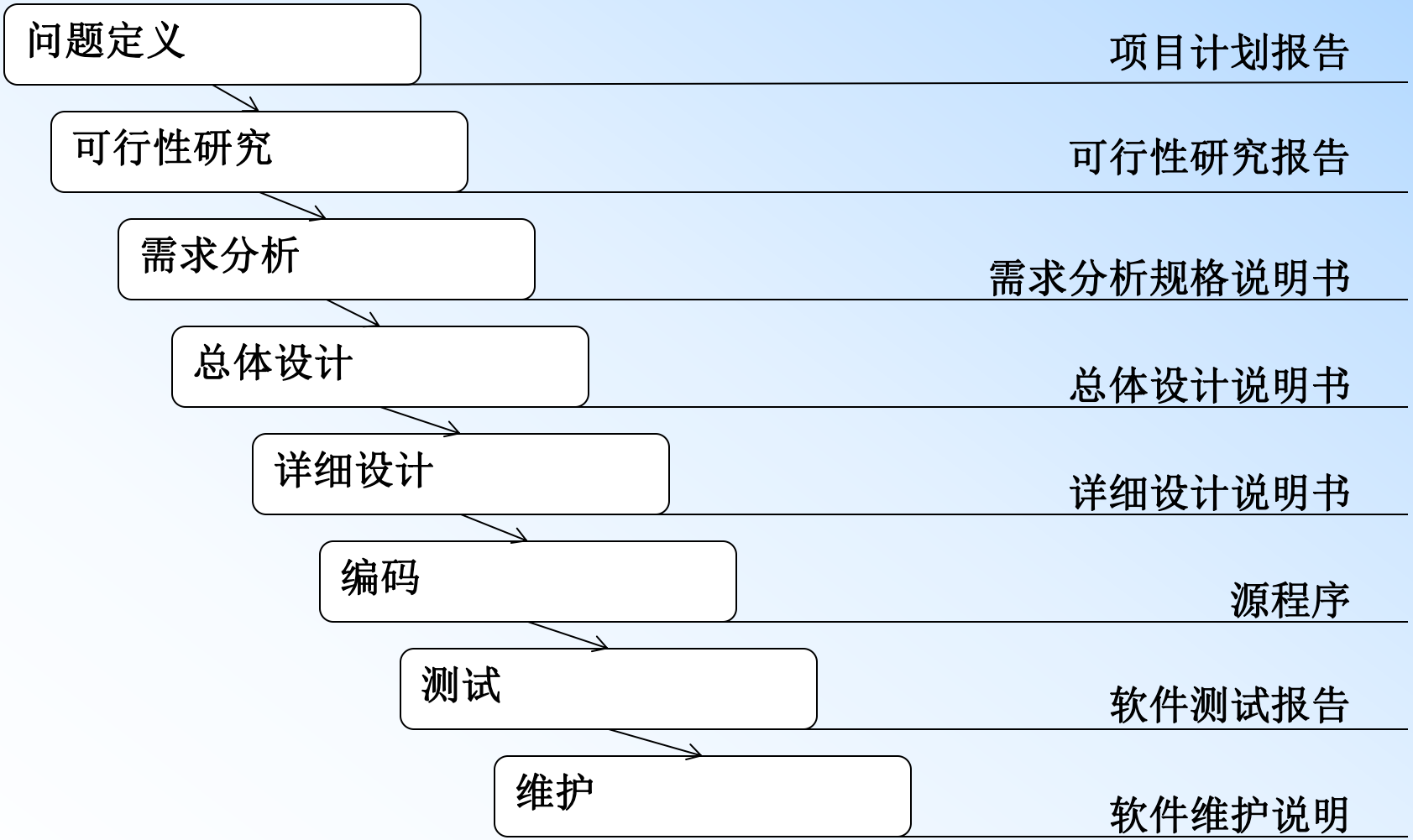
- 软件生命周期
- 软件过程和软件过程模型
- 能力成熟度模型
- 传统软件过程模型
- 现代软件过程模型
- 案例分析

软件生命周期

定义

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。

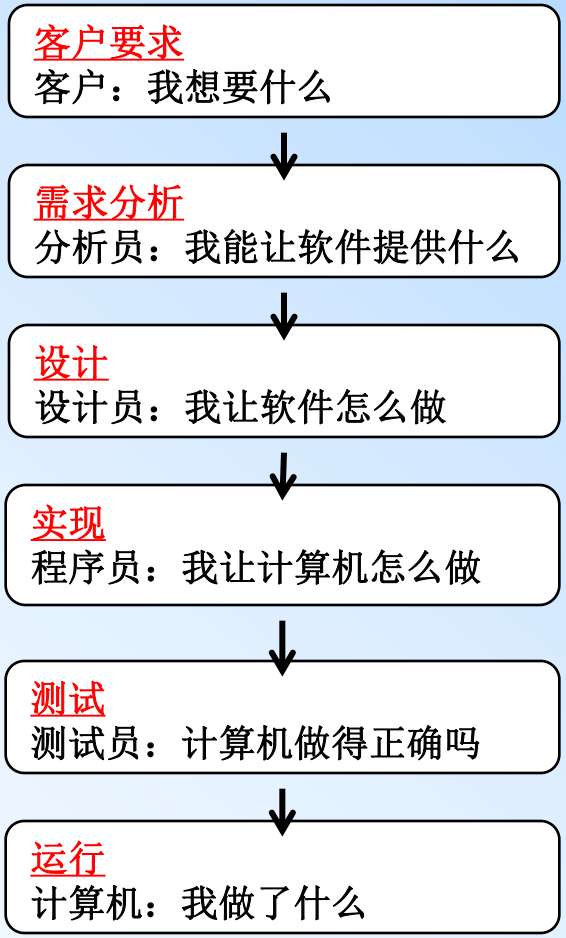
软件生命周期


问题定义
项目计划报告
可行性研究
可行性研究报告
需求分析
需求分析规格说明书
总体设计
总体设计说明书
详细设计
详细设计说明书
编码
源程序
测试
软件测试报告
维护
软件维护说明

软件过程

定义

软件过程定义了软件生产的一系列活动，这些活动贯穿于软件开发的整个过程。



软件过程模型

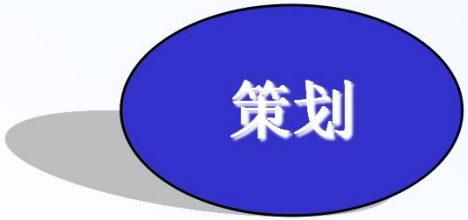
- 软件过程模型是软件开发全部过程、活动和任务的结构框架。它能直观表达软件开发全过程，明确规定要完成的主要活动、任务和开发策略。
- 软件过程模型也常称为
  - 软件开发模型
  - 软件生存周期模型

- 什么人
- 什么时候
- 做什么事
- 怎么做

通用软件过程模型

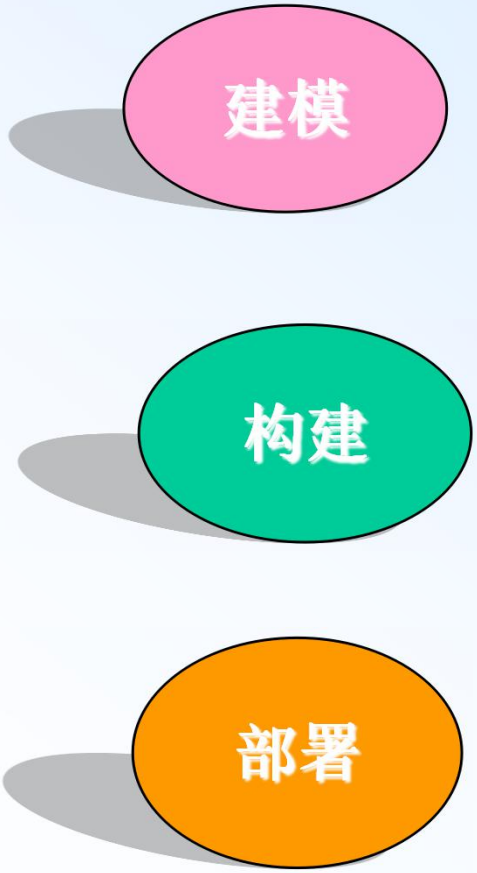
- 软件过程模型多种多样
- 但所有软件过程模型都具有以下共同活动：

软件开发方与客户沟通，客户提出要求，软件开发方收集材料，以及其它相关活动。



软件开发方讨论使用何种方法及何种工具来实现客户需求。

通用软件过程模型



软件开发方讨论选择何种模型来满足需求。不同的需求需要不同的模型。

编码和测试。

软件交付给客户。客户给出建议和反馈，软件开发方改进软件。

过程流（Process flow）

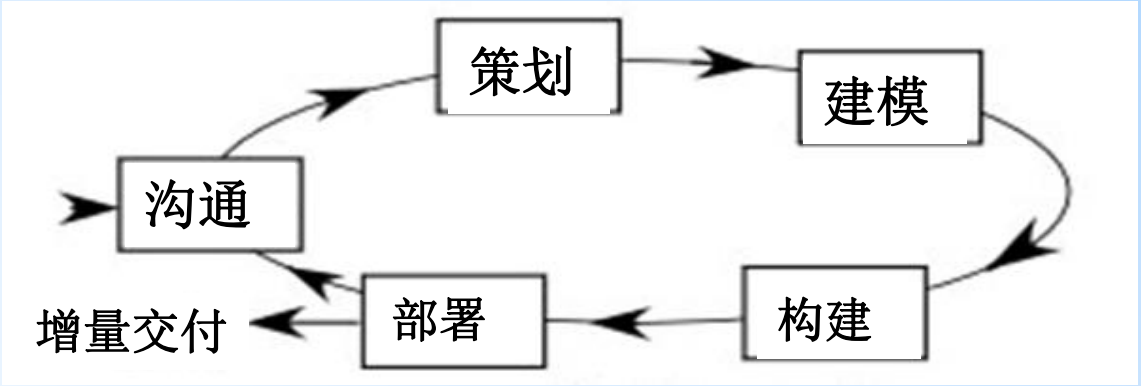
- 过程流描述在执行顺序和执行时间上，如何组织各项活动和任务。

1. 线性过程流：顺序执行各活动

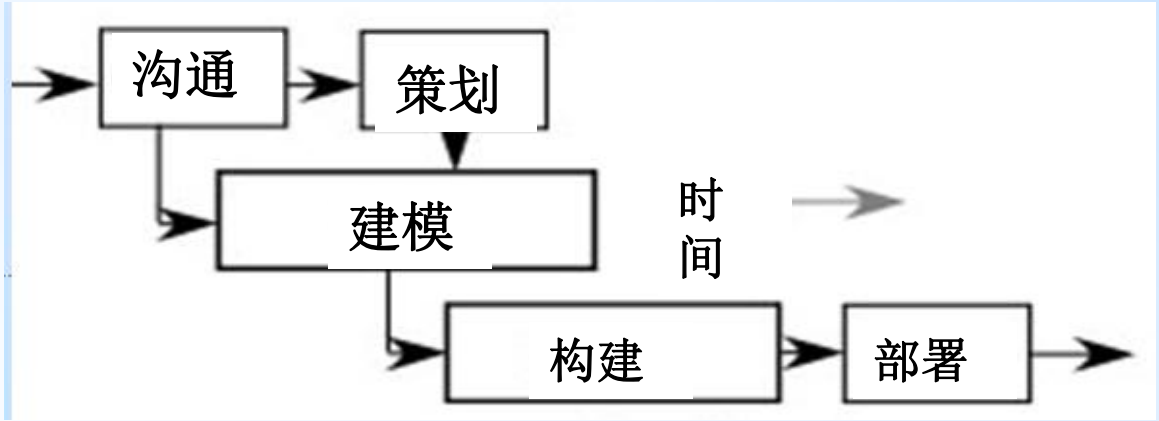
2. 迭代过程流：执行下一活动前重复执行之前的一个或多个活动

过程流

3. 演化过程流：
   采用循环的方式执行各个活动，每次循环都能产生更为完善的软件版本



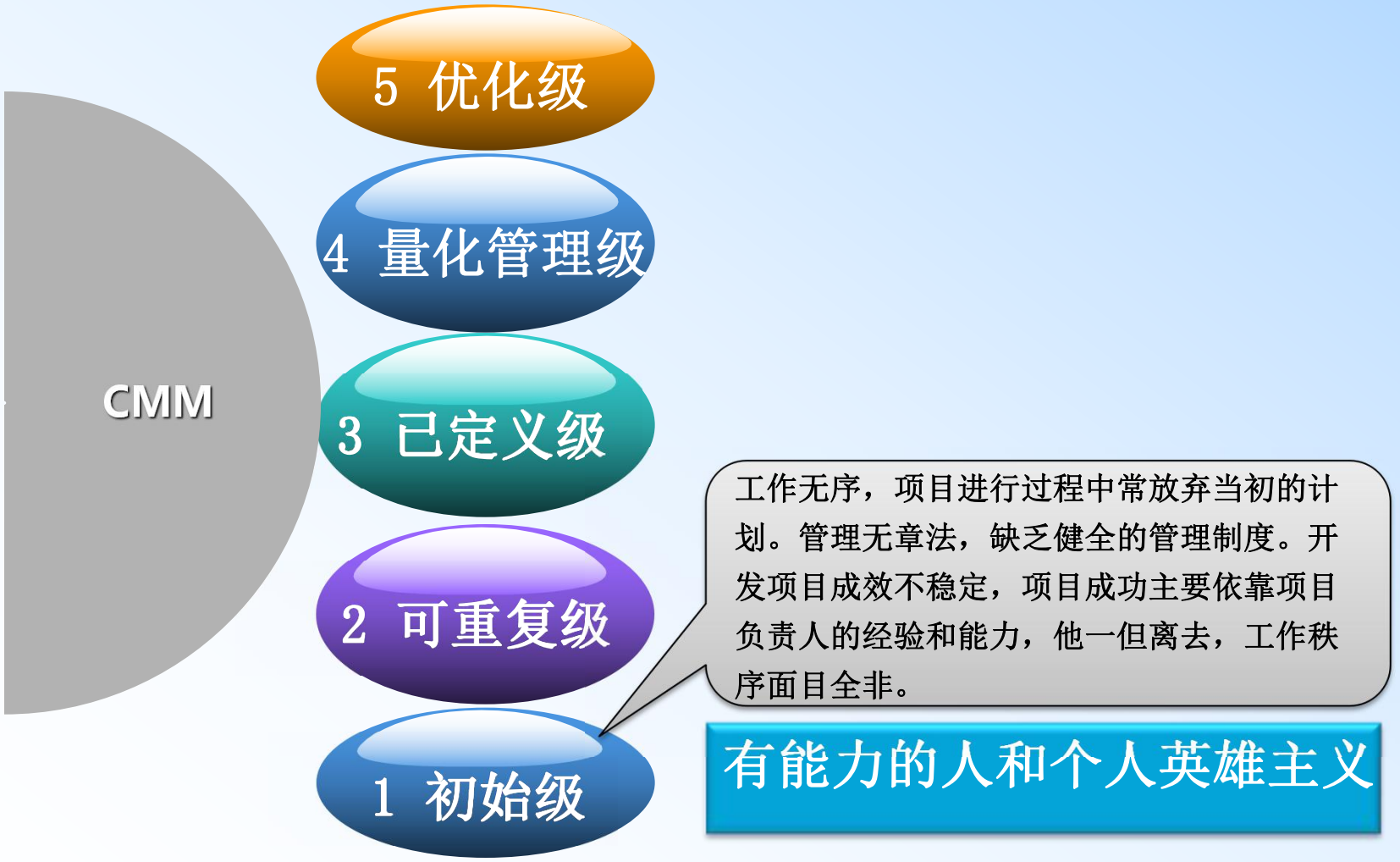
4. 并行过程流：
   将一个或者多个活动与其它活动并行执行



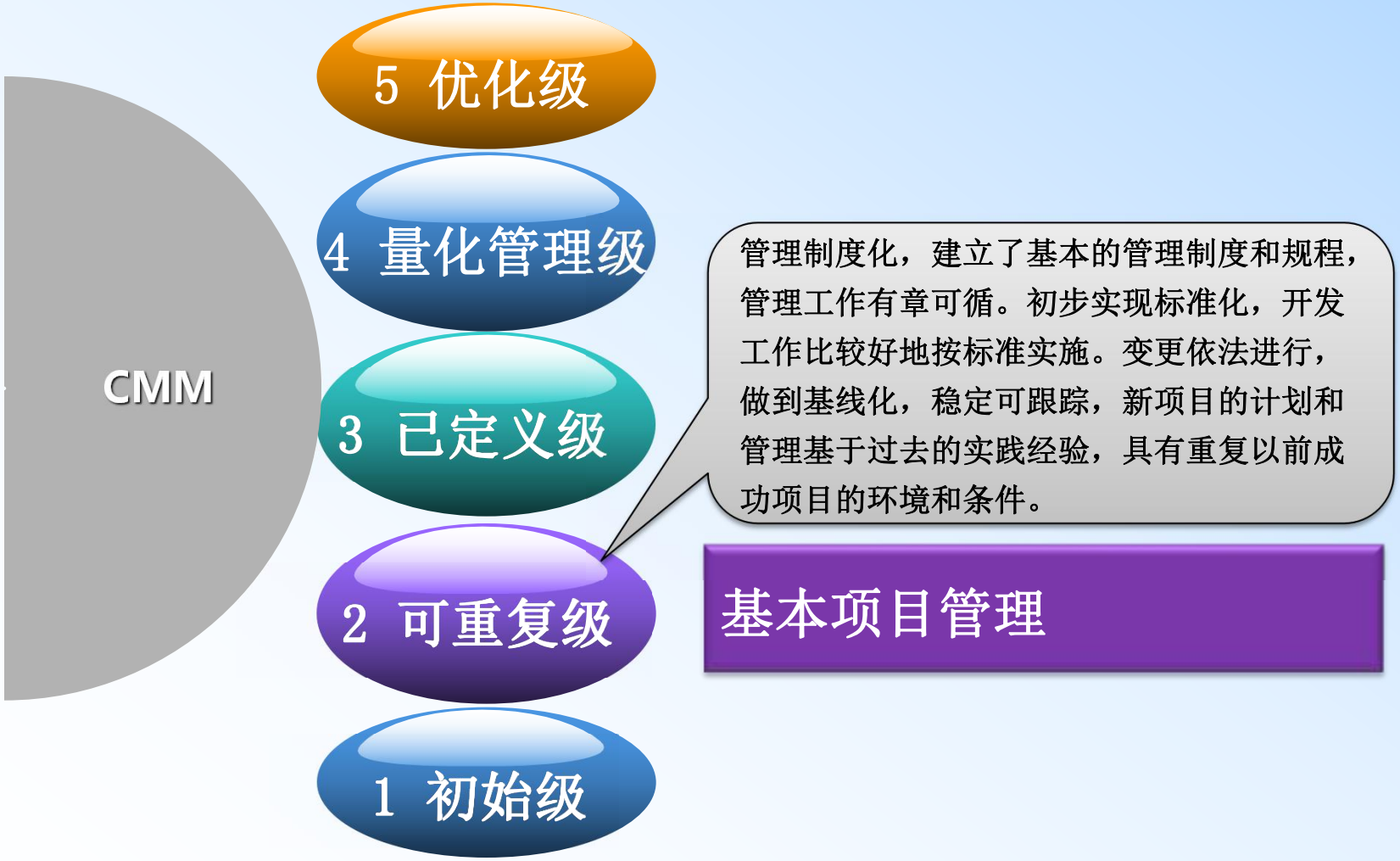
软件过程的三个流派

- CMU-SEI的CMM — 能力成熟度模型
  - Capability Maturity Model
  - 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
- ISO 9000质量标准体系
- ISO/IEC 15504（SPICE）— 信息技术软件过程评估

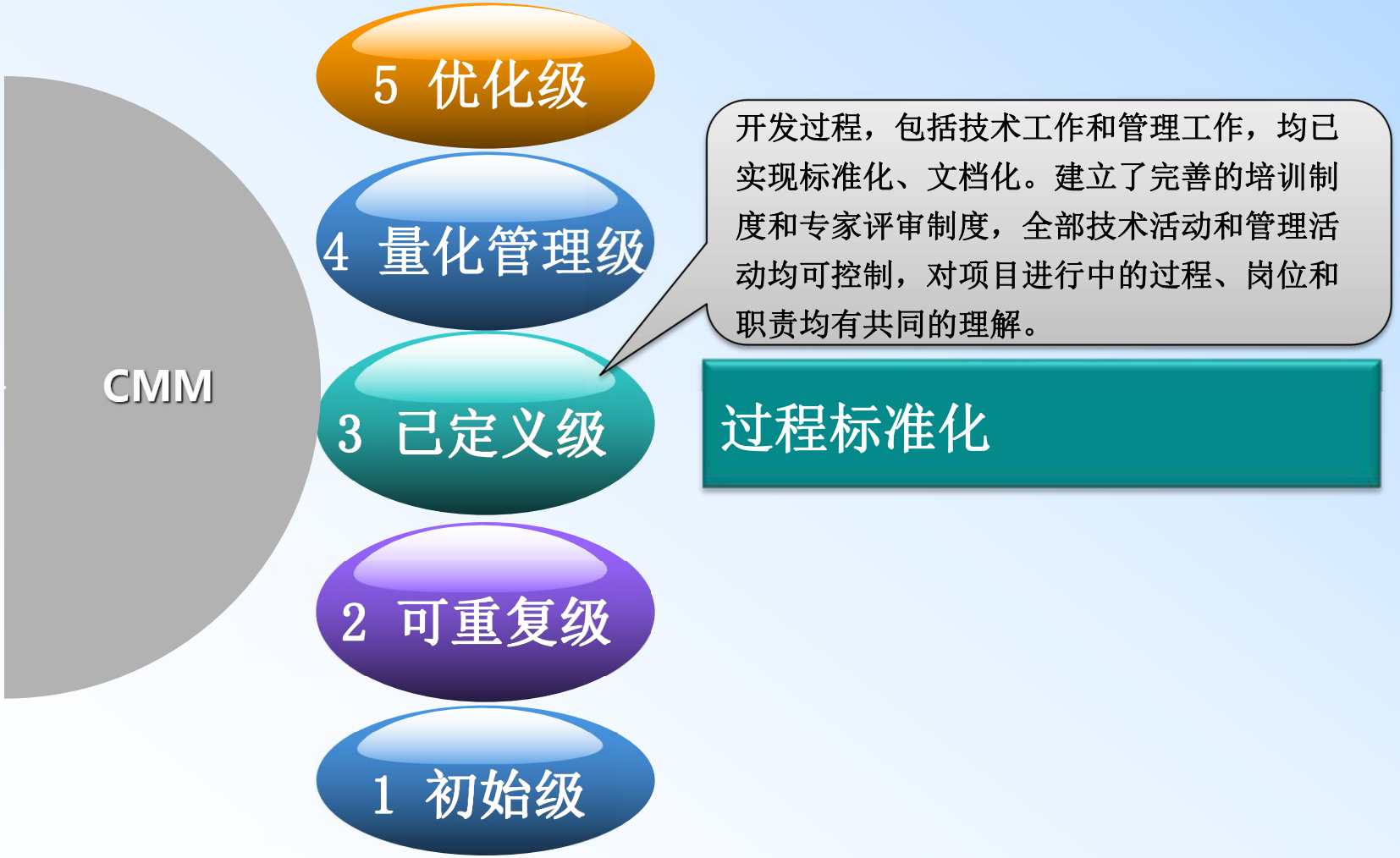
能力成熟度模型CMM


CMM
5 优化级
4 量化管理级
3 已定义级
2 可重复级
1 初始级
工作无序，项目进行过程中常放弃当初的计划。管理无章法，缺乏健全的管理制度。开发项目成效不稳定，项目成功主要依靠项目负责人的经验和能力，他一但离去，工作秩序面目全非。
有能力的人和个人英雄主义

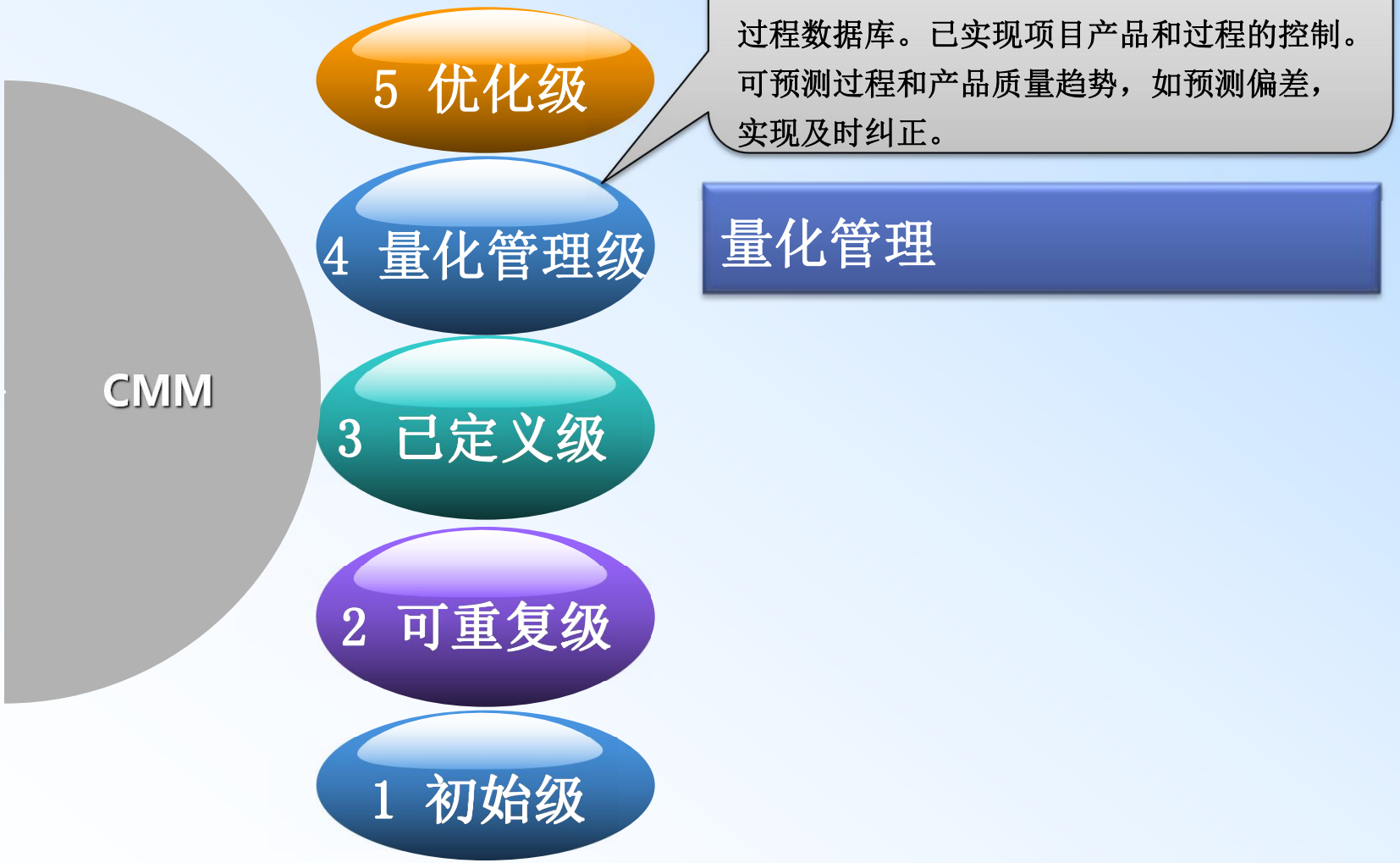
能力成熟度模型CMM


CMM
5 优化级
4 量化管理级
3 已定义级
2 可重复级
1 初始级
管理制度化，建立了基本的管理制度和规程，管理工作有章可循。初步实现标准化，开发工作比较好地按标准实施。变更依法进行，做到基线化，稳定可跟踪，新项目的计划和管理基于过去的实践经验，具有重复以前成功项目的环境和条件。
基本项目管理

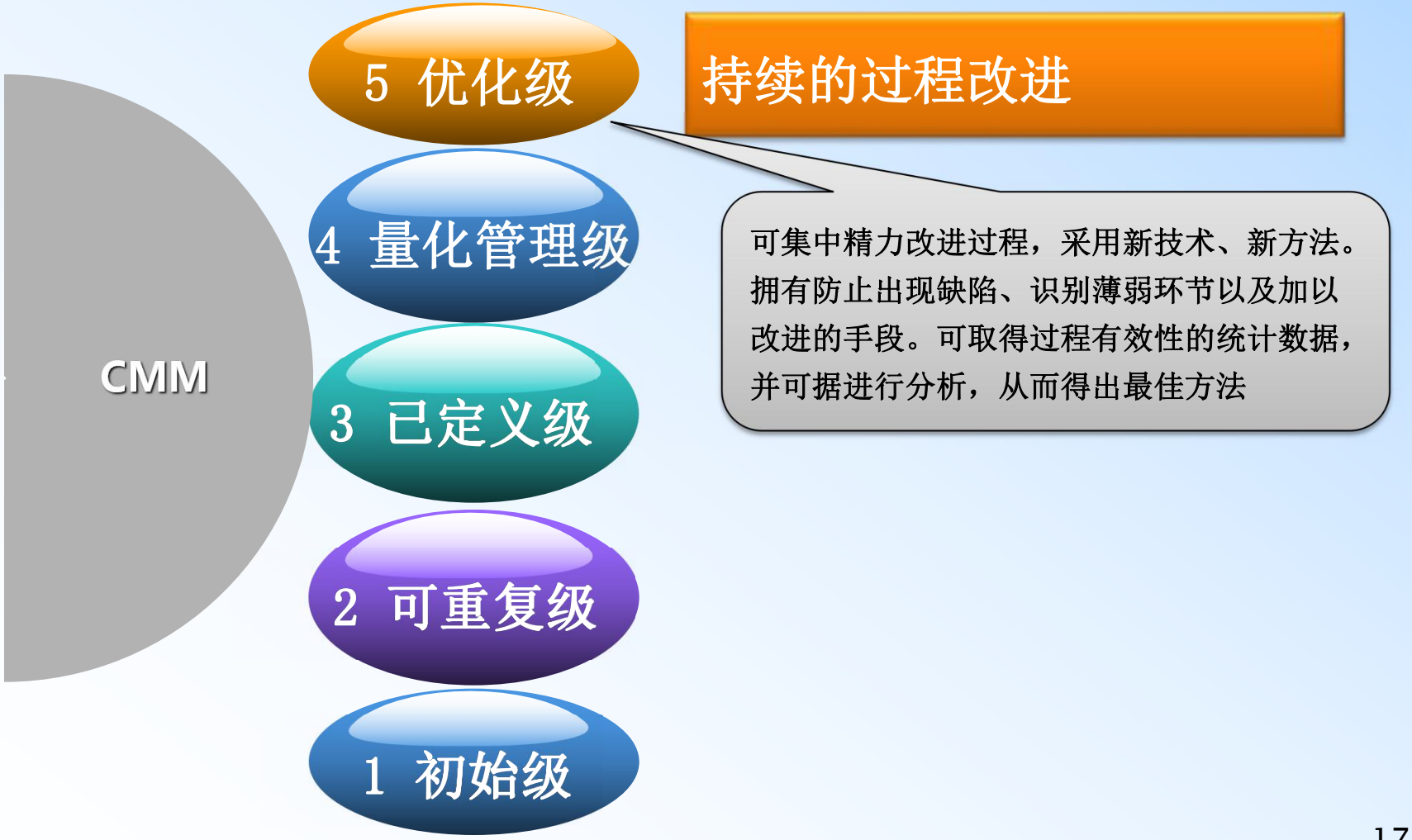
能力成熟度模型CMM


CMM
5 优化级
4 量化管理级
3 已定义级
2 可重复级
1 初始级
开发过程，包括技术工作和管理工作，均已实现标准化、文档化。建立了完善的培训制度和专家评审制度，全部技术活动和管理活动均可控制，对项目进行中的过程、岗位和职责均有共同的理解。
过程标准化

能力成熟度模型CMM


CMM
5 优化级
4 量化管理级
3 已定义级
2 可重复级
1 初始级
产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势，如预测偏差，实现及时纠正。
量化管理

能力成熟度模型CMM


CMM
5 优化级
4 量化管理级
3 已定义级
2 可重复级
1 初始级
持续的过程改进
可集中精力改进过程，采用新技术、新方法。拥有防止出现缺陷、识别薄弱环节以及加以改进的手段。可取得过程有效性的统计数据，并可据进行分析，从而得出最佳方法

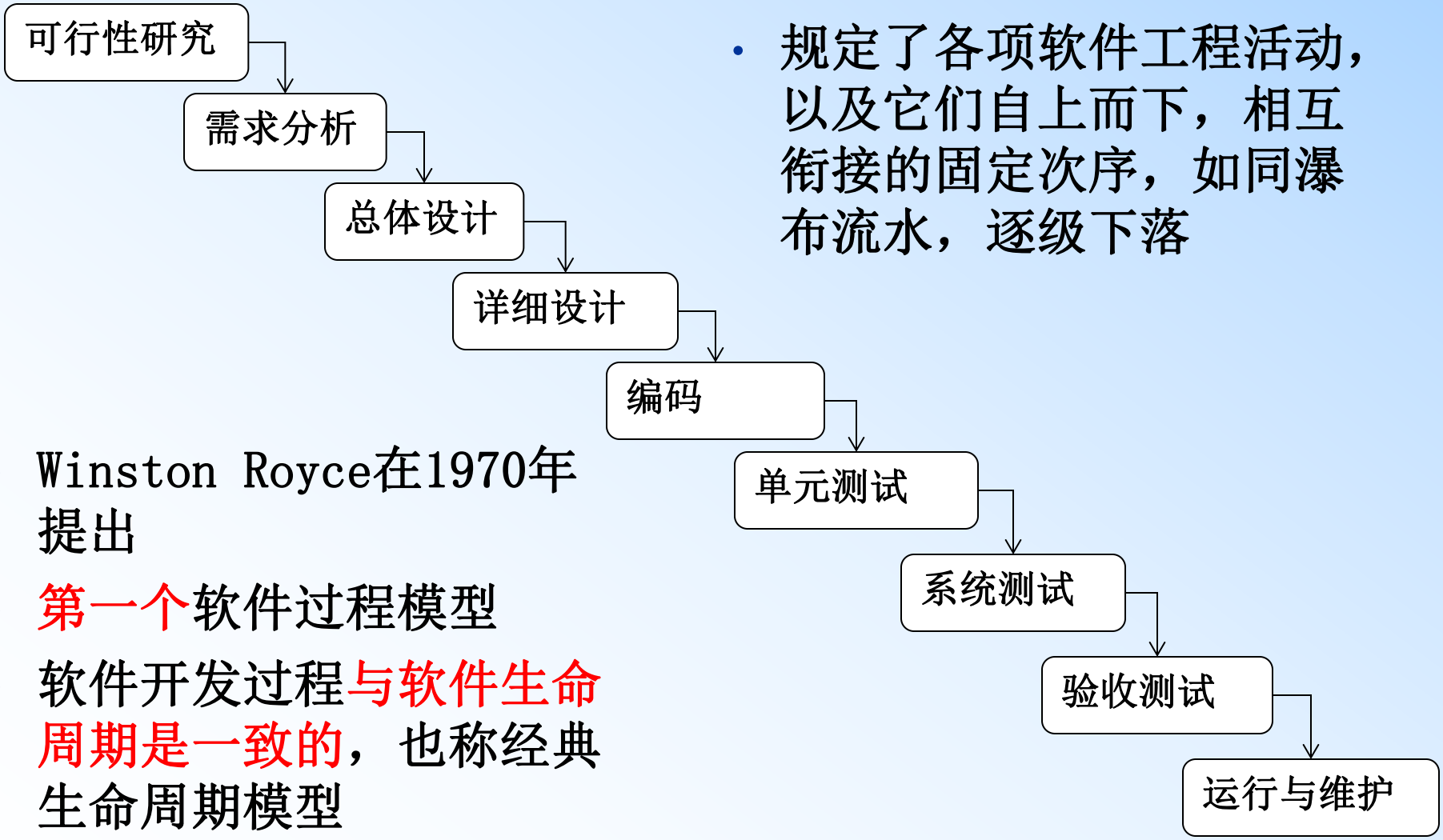
中国通过CMM5认证的企业（部分）

- 摩托罗拉中国软件中心（2000年9月）
- 沈阳东软股份有限公司（2002年12月）
- 华为印度研究所（2003年8月）
- 惠普中国软件研发中心（2004年6月）
- 北京用友软件工程有限公司（2004年12月）
- 埃森哲全球信息技术中心（2005年4月）
- 普天信息技术研究院（2006年11月）
- 上海宝信软件股份有限公司（2006年12月）
- 亚信科技（中国）有限公司（2007年2月）
- 大连现代高技术发展有限公司（2004年10月）
- 长沙新宇计算机系统有限公司（2003年11月）
- 联想软件公司（2006年2月）
- ……

软件过程模型

- 传统软件过程模型
  - 瀑布模型
  - 增量模型
  - 原型模型
  - 螺旋模型
  - 协同模型
  - 喷泉模型
- 现代软件过程模型
  - 基于构件的开发模型
  - 形式化方法模型
  - 面向方面的软件开发
  - Rational统一过程
  - 敏捷软件开发

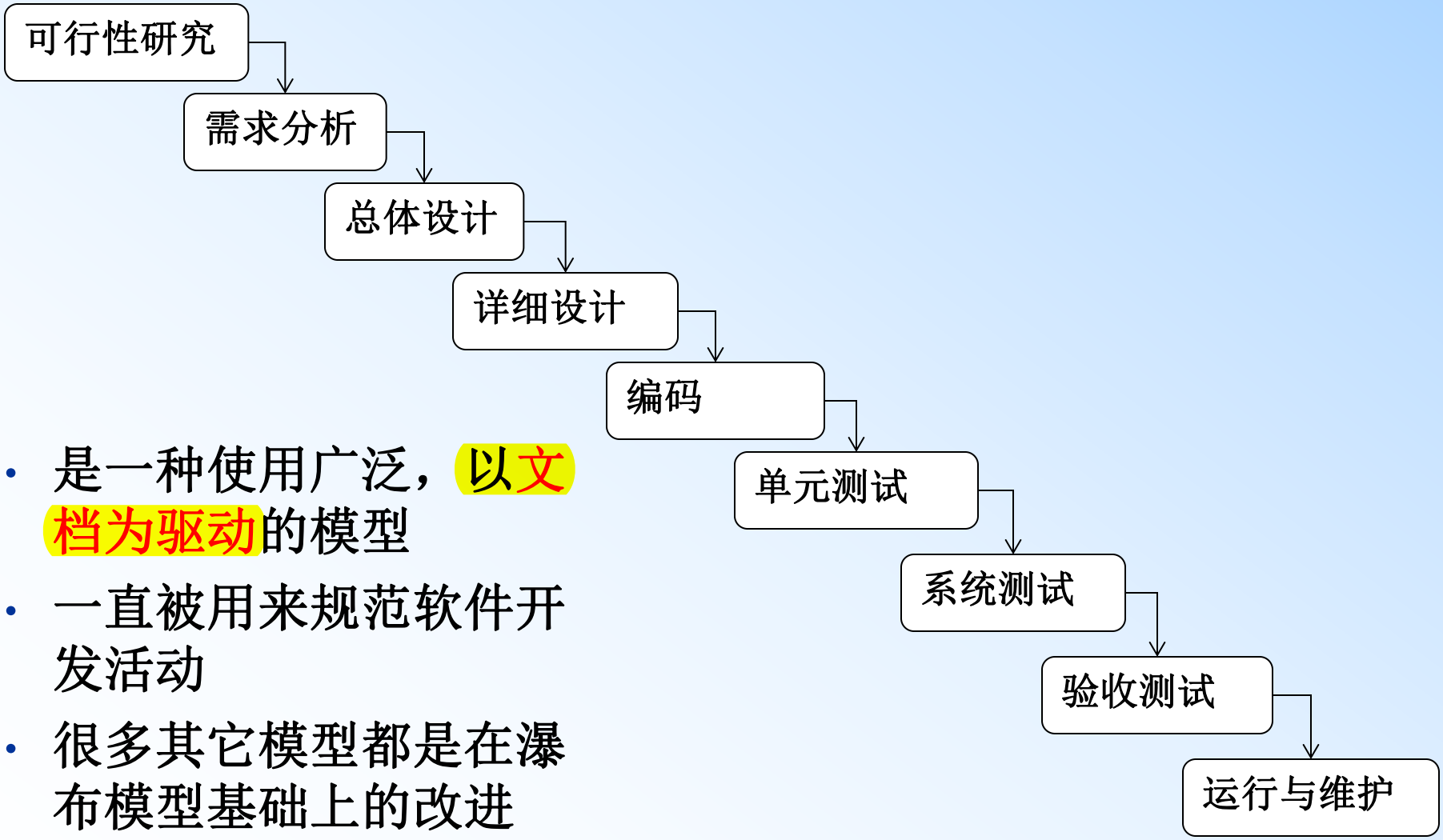
瀑布模型（Waterfall model）



- 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

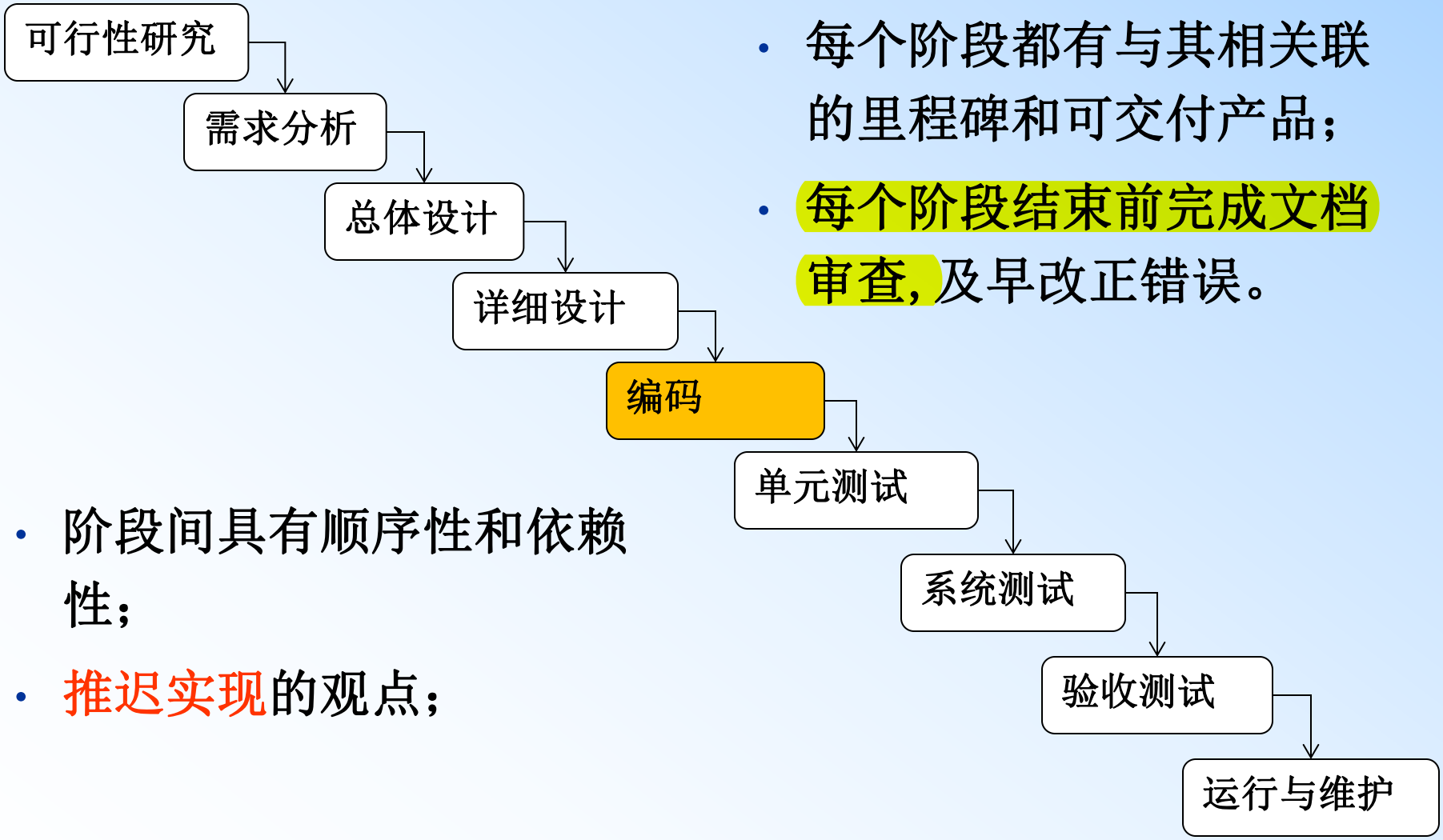
- Winston Royce在1970年提出
- 第一个软件过程模型
- 软件开发过程与软件生命周期是一致的，也称经典生命周期模型

瀑布模型



- 是一种使用广泛，以文档为驱动的模型
- 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

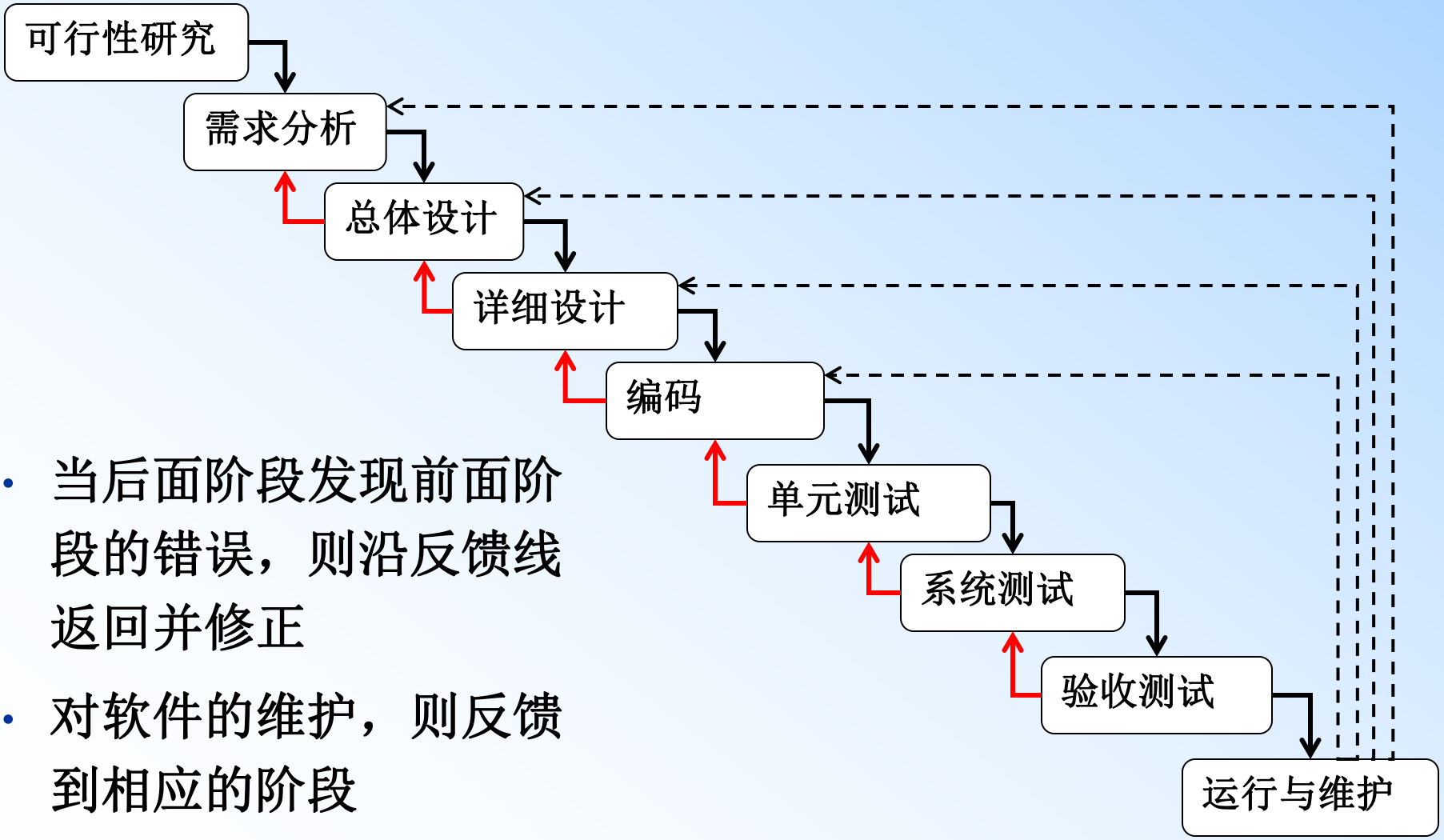
瀑布模型的特点和优点



- 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。

- 阶段间具有顺序性和依赖性；
- 推迟实现的观点；

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

瀑布模型的缺点

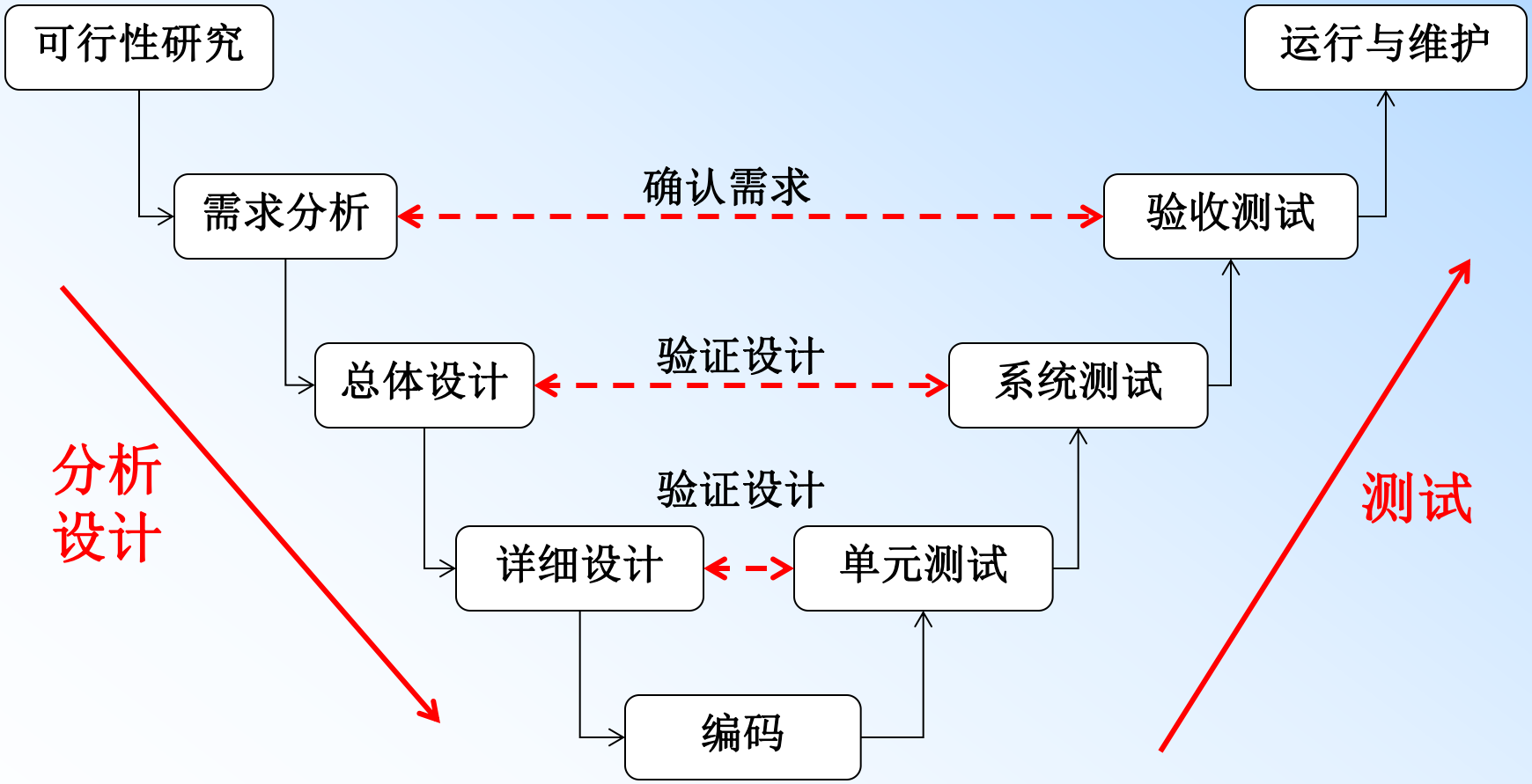
- 各个阶段的划分完全固定，阶段之间产生大量的文档，极大地增加了工作量；
- 由于开发模型是线性的，用户只有等到整个过程的末期才能见到开发成果，从而增加了开发的风险；
- 早期的错误可能要等到开发后期的测试阶段才能发现，进而带来严重的后果；
- 无法适应需求不明确和需求的变化；
- 不能反映实际的开发方式，软件开发需要迭代。

瀑布模型的适用场合

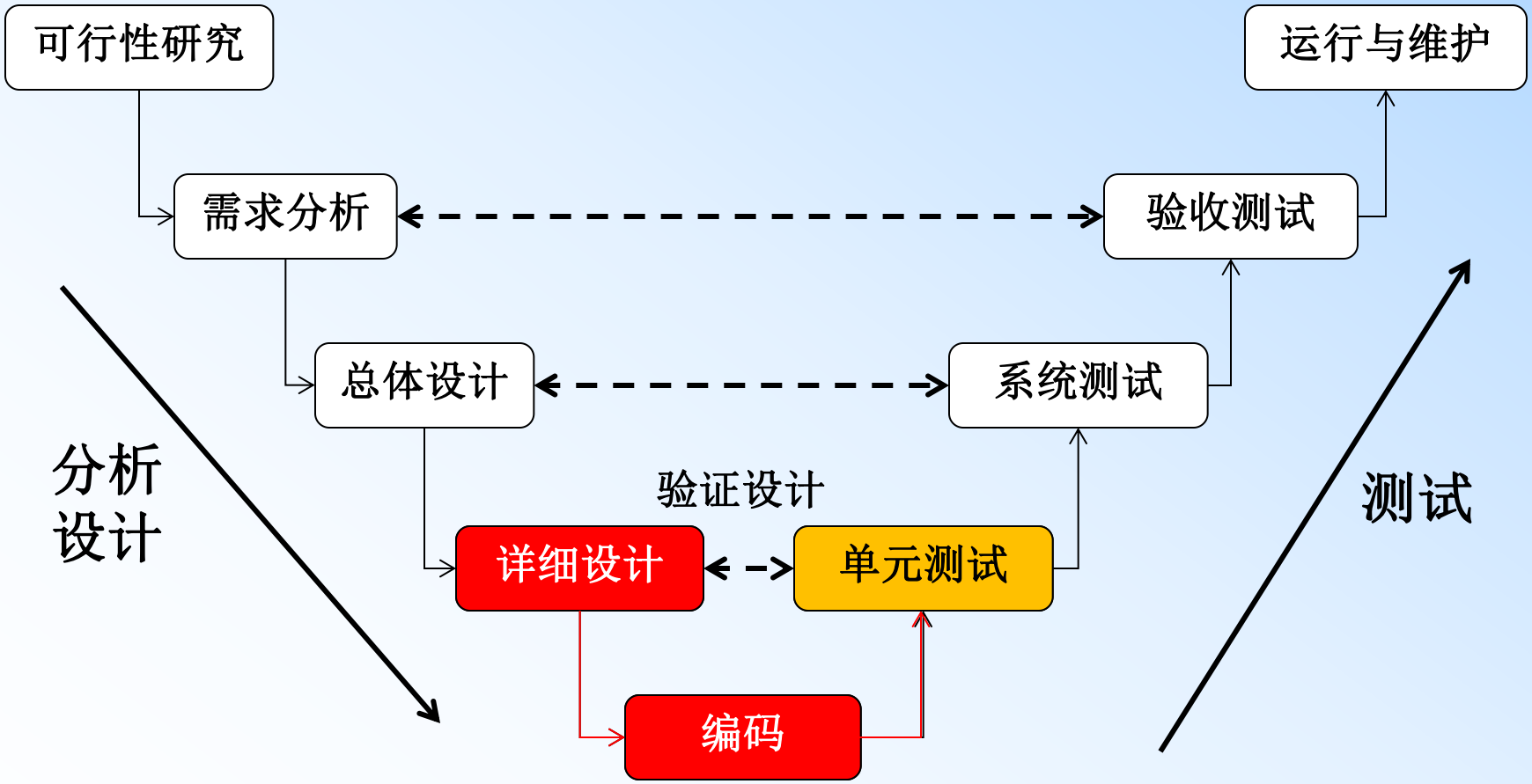
适用场合

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。

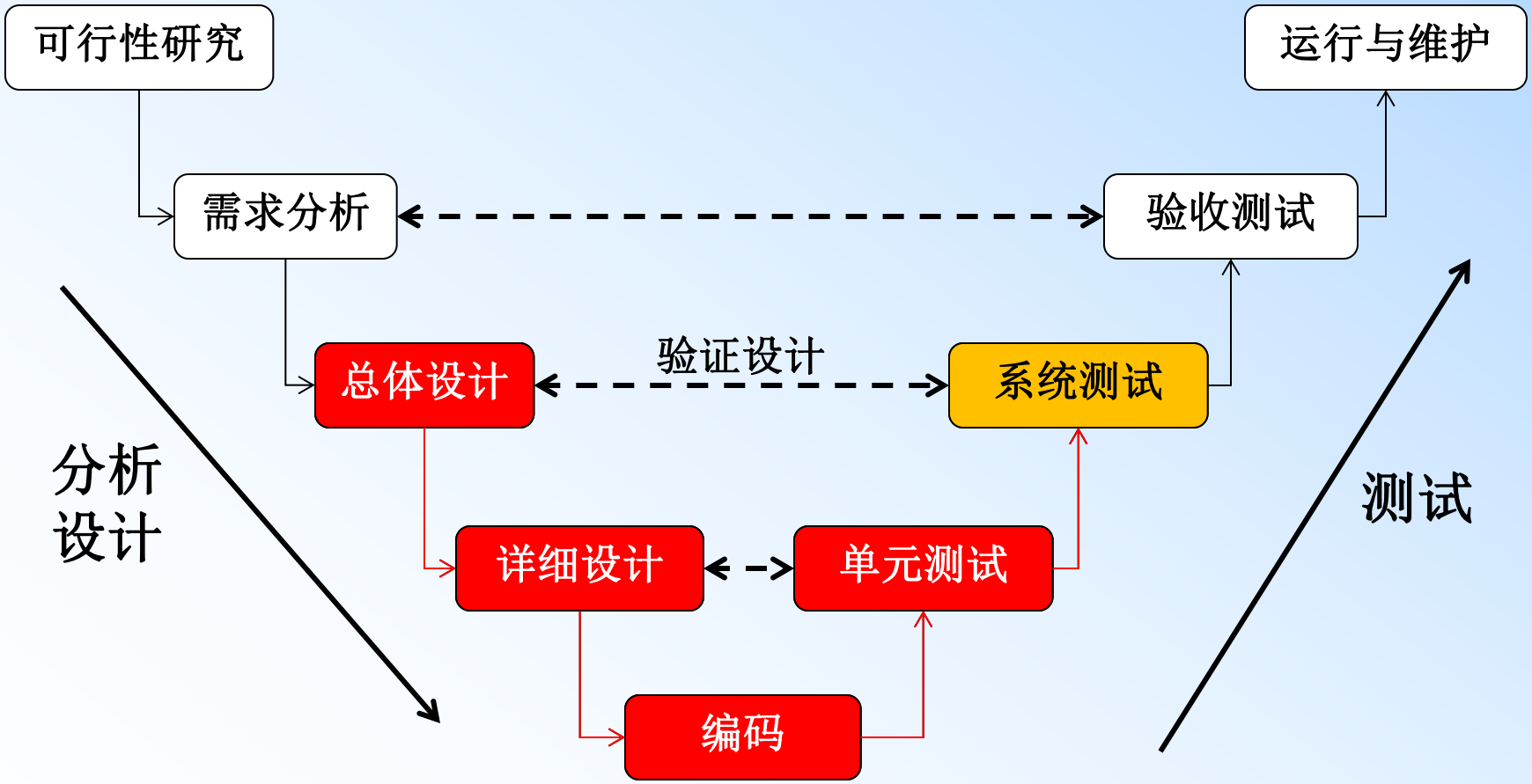
V模型（V-model）：瀑布模型的变种



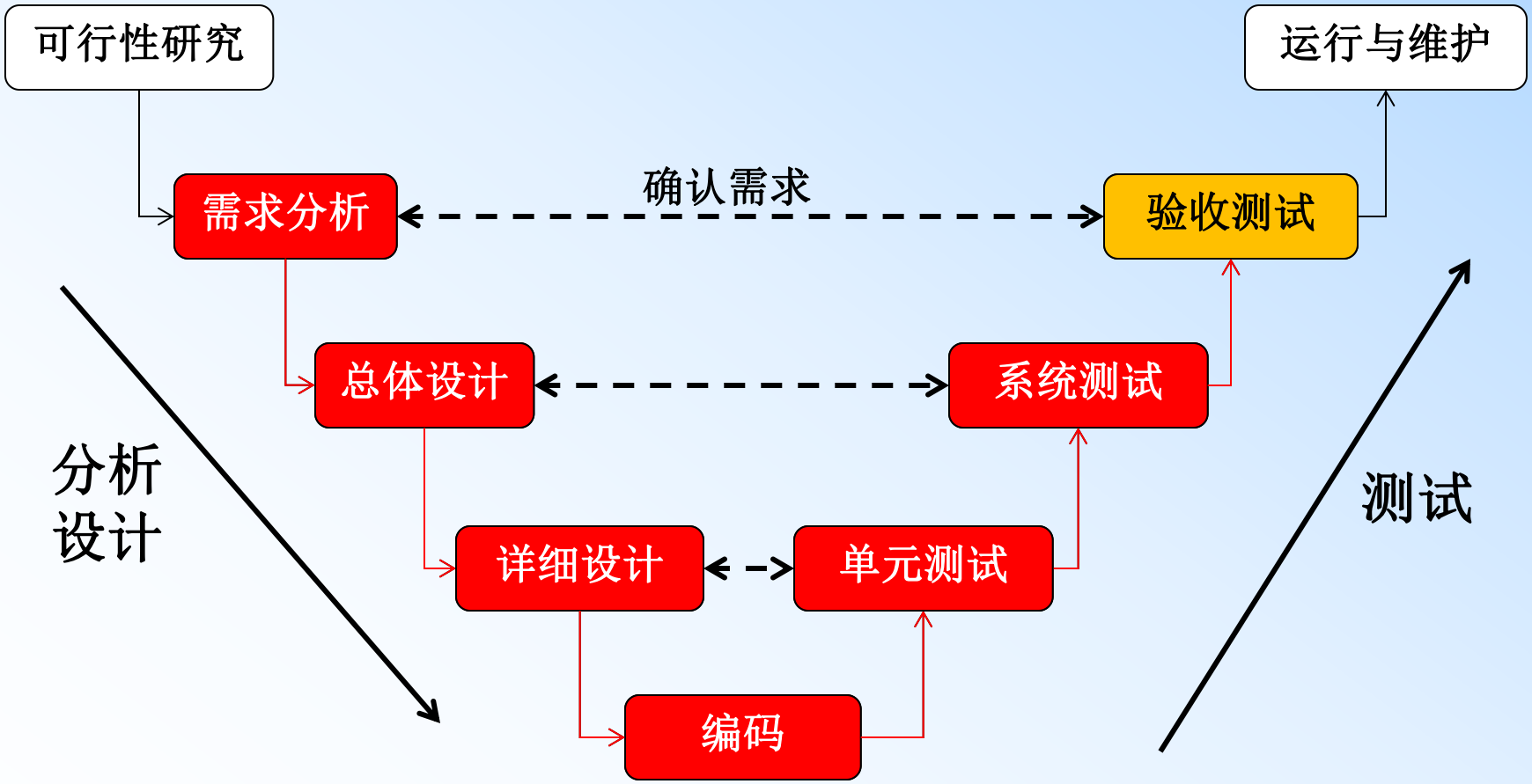
V模型：单元测试发现问题



V模型：系统测试发现问题



V模型：验收测试发现问题



原型模型（Prototype model）

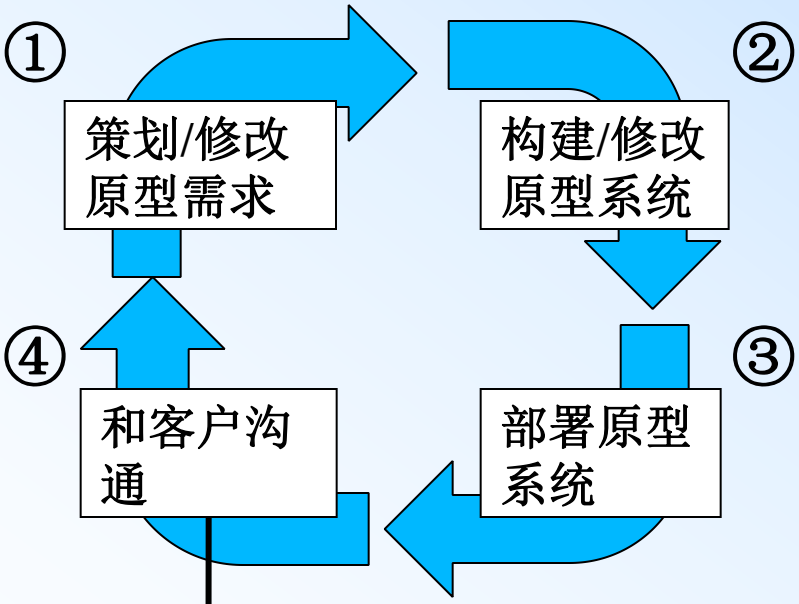
- 原型化模型、快速原型模型
- 原型（prototype）：
  - 一个部分开发的产品，使客户和开发人员能够对计划开发的系统的相关方面进行检查。
- 举例1：
  - 图书馆管理系统：主要界面
- 举例2：
  - 智能家居系统：少量的室内信息监视和电器控制

原型化模型

- 原型化的目的：
  - 明确并完善需求，如演示原型
  - 研究技术选择方案，如技术验证原型

- 原型结果
  - 抛弃原型
  - 把原型发展成最终产品

原型化模型



快！

Rapid！

用户
满意

需求分析 → 设计 → 实现 → 测试 → 维护

原型化模型的优点和缺点

- 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

原型化模型的适用场合

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，原型法是很好的选择。

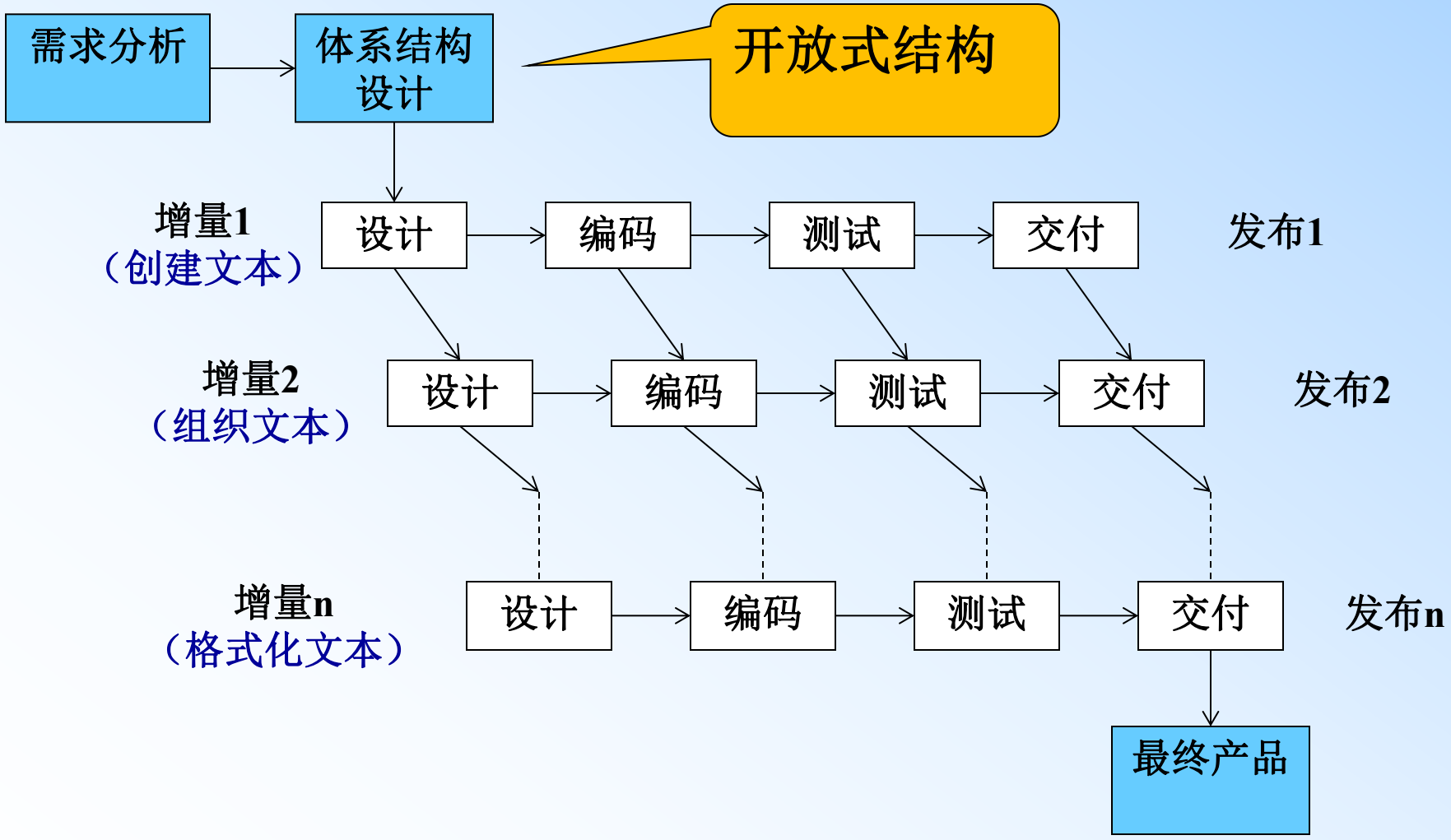
增量模型（Incremental model）

- 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
  - 第一个增量：创建文本
  - 第二个增量：组织文本
  - 第三个增量：格式化文本
  - 第一个发布：创建文本
  - 第二个发布：创建文本、组织文本
  - 第三个发布：创建文本、组织文本、格式化文本

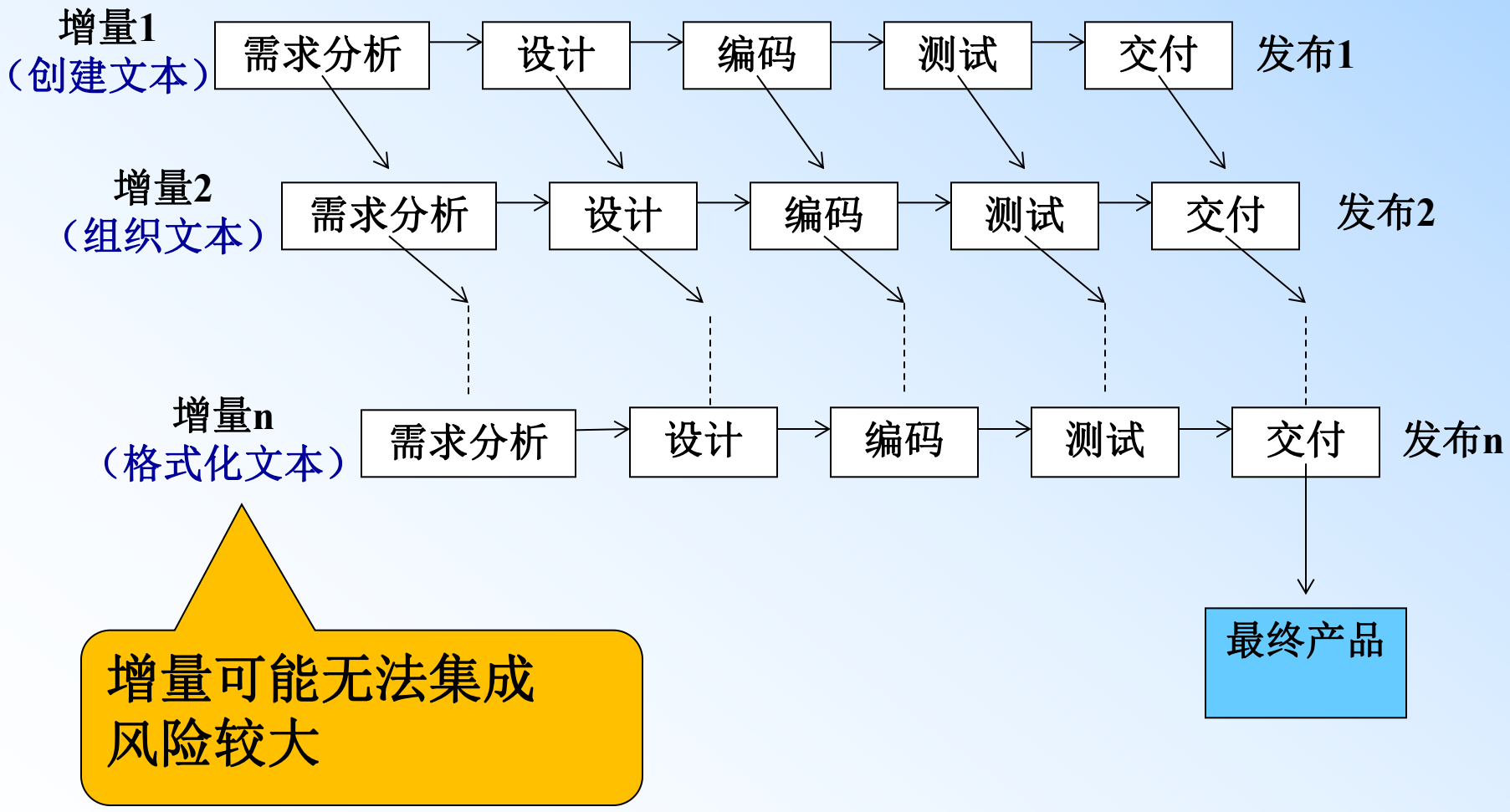
增量模型

- 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。它允许从部分需求定义出发，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

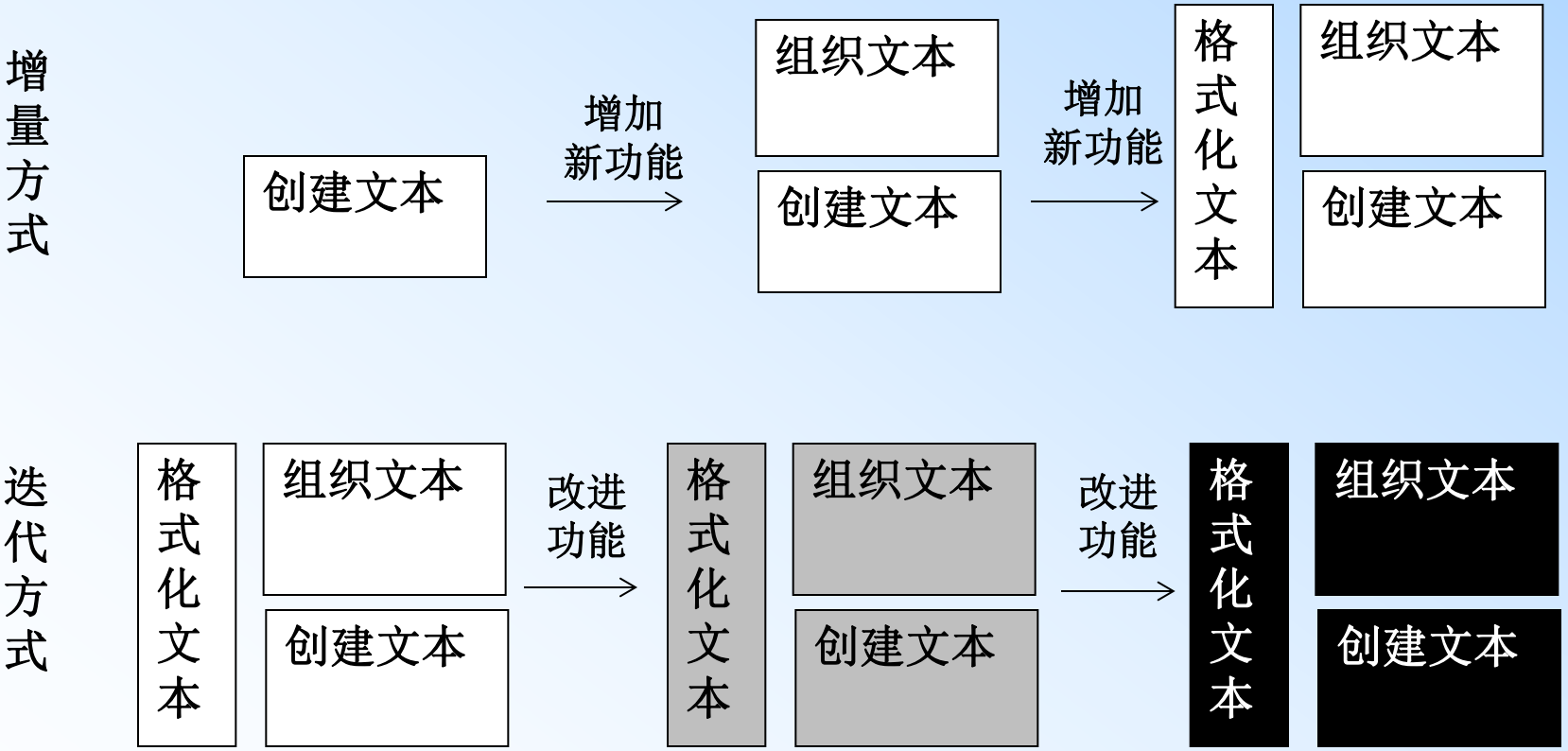
增量模型


需求分析
体系结构设计
开放式结构
增量1
（创建文本）
设计
编码
测试
交付
发布1
增量2
（组织文本）
设计
编码
测试
交付
发布2
增量n
（格式化文本）
设计
编码
测试
交付
发布n
最终产品

增量模型


增量1
（创建文本）
需求分析
设计
编码
测试
交付
发布1
增量2
（组织文本）
需求分析
设计
编码
测试
交付
发布2
增量n
（格式化文本）
需求分析
设计
编码
测试
交付
发布n
最终产品
增量可能无法集成
风险较大

增量的方式



实际使用中，常常是两种方式的结合

增量模型优点和缺点

- 优点：
  - 增量包概念的引入，以及它不需要提供完整的需求，只要有一个增量包出现，开发就可以进行。
  - 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
  - 软件能够更早投入市场
  - 开放式体系结构，便于维护
- 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

增量模型的适用场合

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。

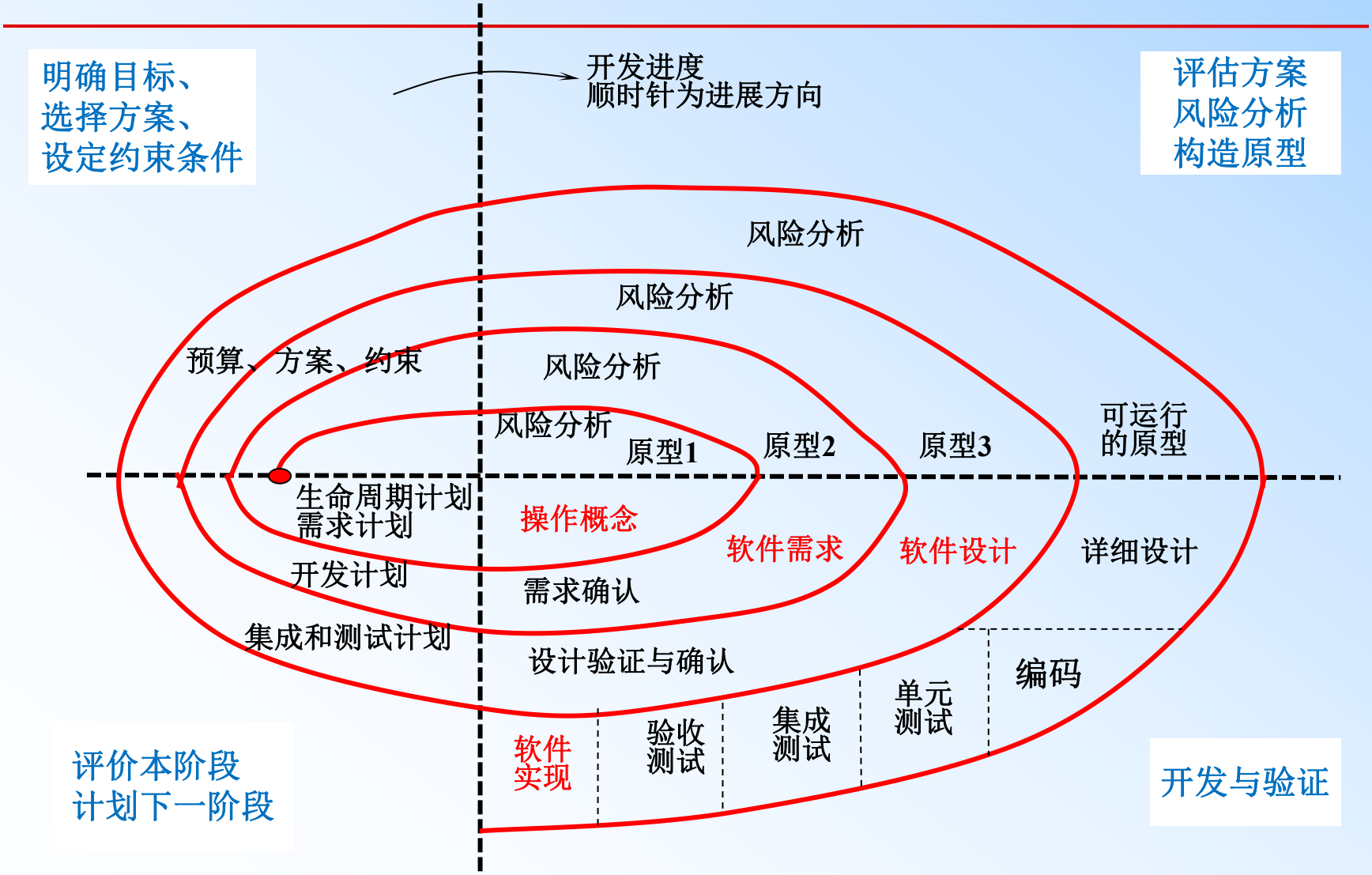
螺旋模型（Spiral model）

- 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
- 模型结合了瀑布模型和原型模型的特点

螺旋模型

- 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，考虑如何识别和消除风险；
  ③ 实施软件开发和验证；
  ④ 评价该阶段的工作成果，提出修正建议，并计划下一阶段工作。

螺旋模型


明确目标、
选择方案、
设定约束条件
开发进度
顺时针为进展方向
评估方案
风险分析
构造原型
风险分析
风险分析
预算、方案、约束
风险分析
风险分析
原型1
原型2
原型3
可运行
的原型
生命周期计划
需求计划
操作概念
软件需求
软件设计
详细设计
开发计划
需求确认
集成和测试计划
设计验证与确认
编码
单元
测试
集成
测试
验收
测试
软件
实现
评价本阶段
计划下一阶段
开发与验证

螺旋模型

- 以图为例：
  - 第一次迭代产生操作概念
  - 第二次迭代产生软件需求
  - 第三次迭代产生系统设计
  - 第四次迭代产生软件实现
- 每一次迭代都进行风险分析，并通过原型化进行验证，降低或消除风险
- 实际迭代的次数可能更多

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。

螺旋模型的缺点

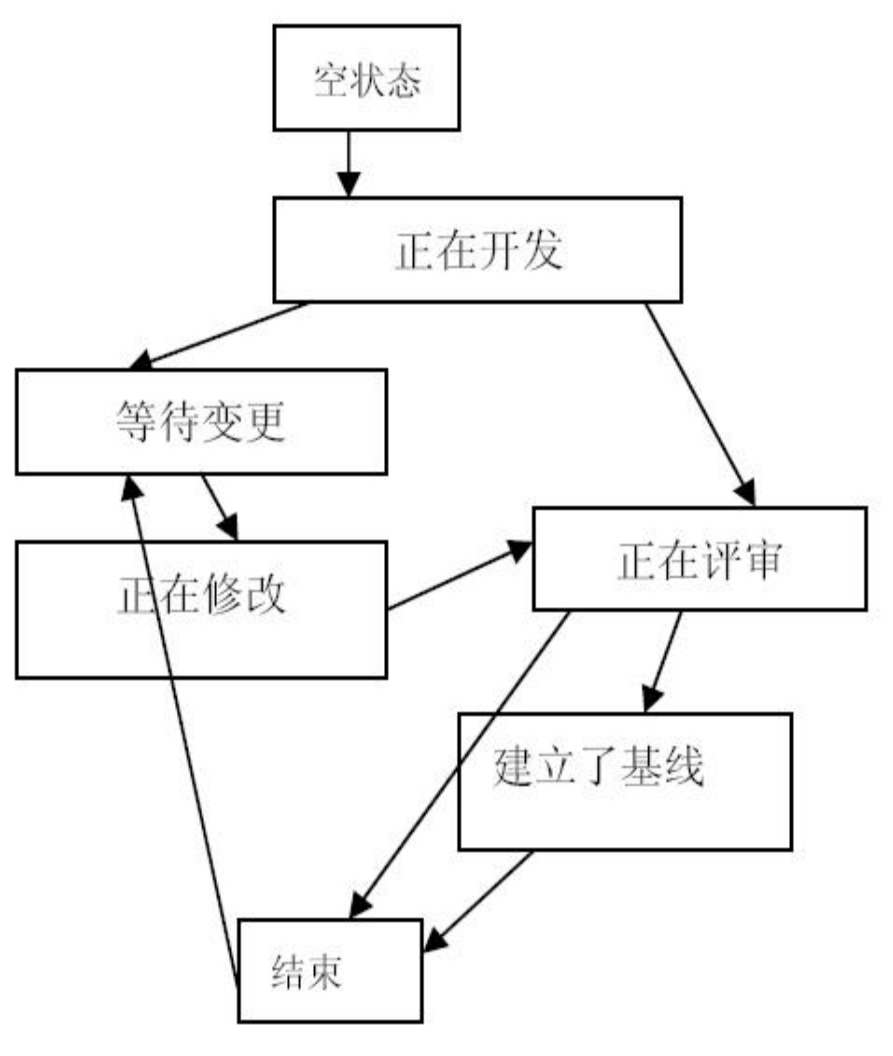
- 如果每次迭代的效率不高，致使迭代次数过多，将会增加成本并推迟提交时间；
- 使用该模型需要有相当丰富的风险评估经验和专门知识，要求开发队伍水平较高，否则会带来更大风险。

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。

协同开发模型

- Concurrent development model
- 协同模型、协同工程、并行开发模型
- 表达软件过程模型中的迭代和并发元素
- 提供精确的项目当前状态图
- 例如：建模活动
- 通过定义的事件触发软件工程活动
- 所有软件工程活动同时存在并处于不同状态
- 可用于所有类型的软件开发



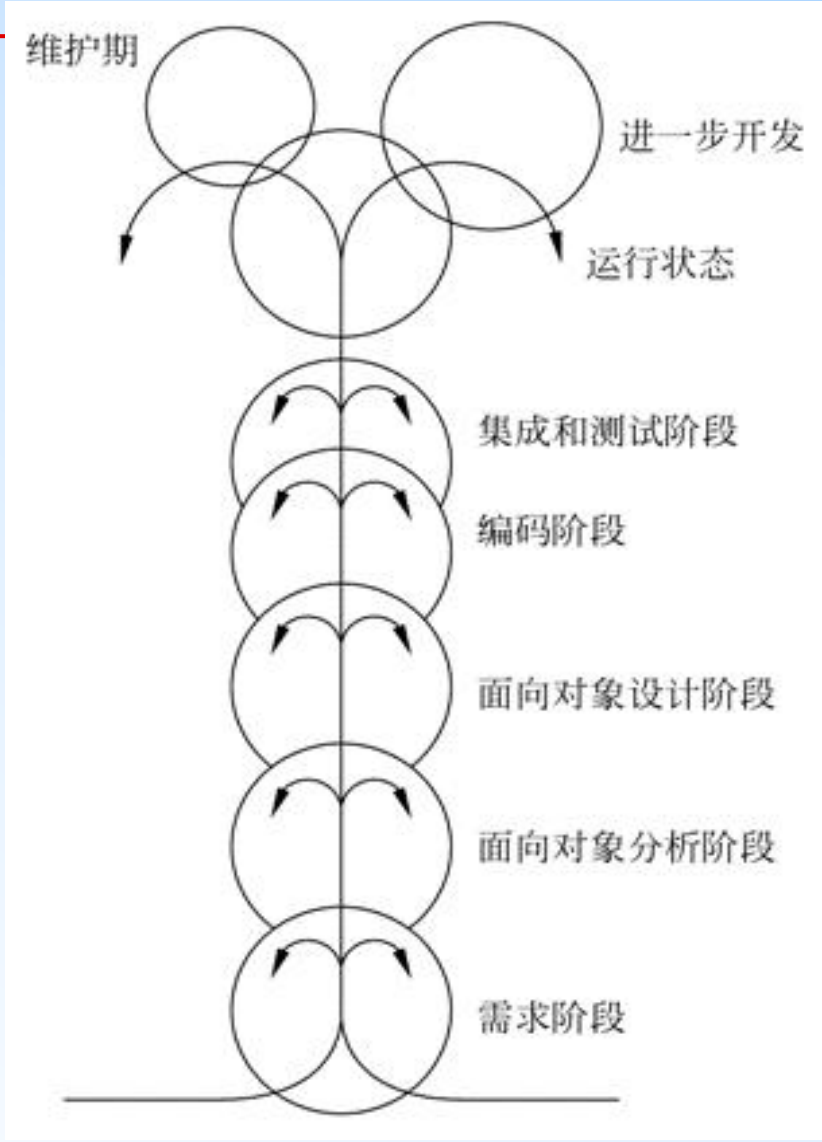
适用场合 不同团队共同开发的项目。

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。
- 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

适用场合

适用于面向对象开发。



喷泉模型

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。此外这种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各种信息、需求与资料的情况。

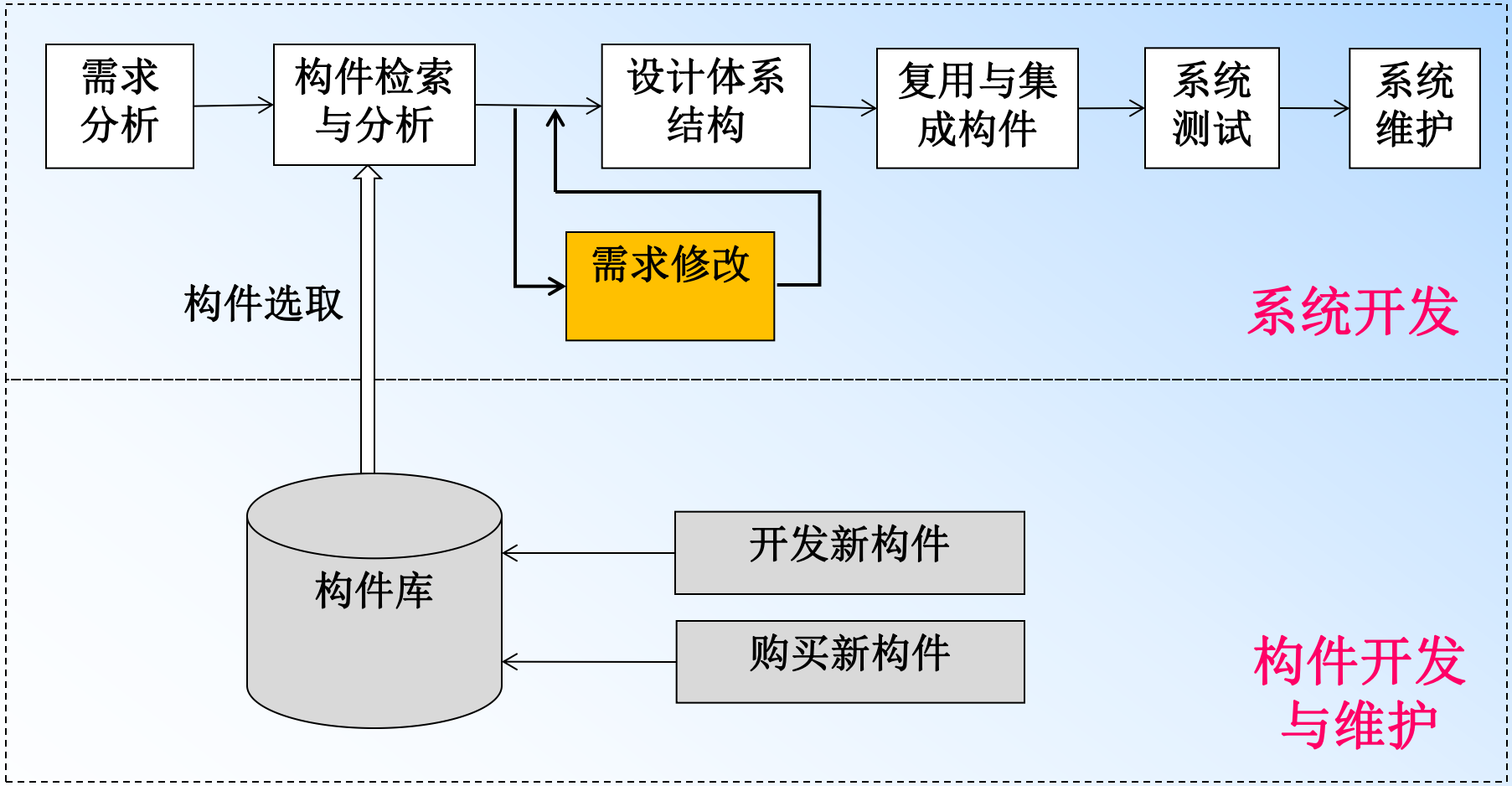
现代软件过程模型

- 基于构件的开发模型
- 形式化方法模型
- 面向方面的软件开发
- Rational统一过程
- 敏捷软件开发

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
- 构件/组件（Component）
  - 系统中模块化的、可更换的部分
  - 它实现特定的功能，并对实现进行封装
  - 暴露一组接口
  - 例如：动态链接库（.dll），浏览器中的插件
- 考虑的焦点是集成，而非实现

基于构件的开发模型


需求分析
构件检索与分析
设计体系结构
复用与集成构件
系统测试
系统维护
需求修改
构件选取
系统开发
构件库
开发新构件
购买新构件
构件开发与维护

基于构件的开发模型

- 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

基于构件的开发模型

适用场合

适用于系统之间有共性的情况。

- 基于构件的标准
  - Microsoft COM、Microsoft .NET
  - OMG/CORBA
  - Sun JavaBeans

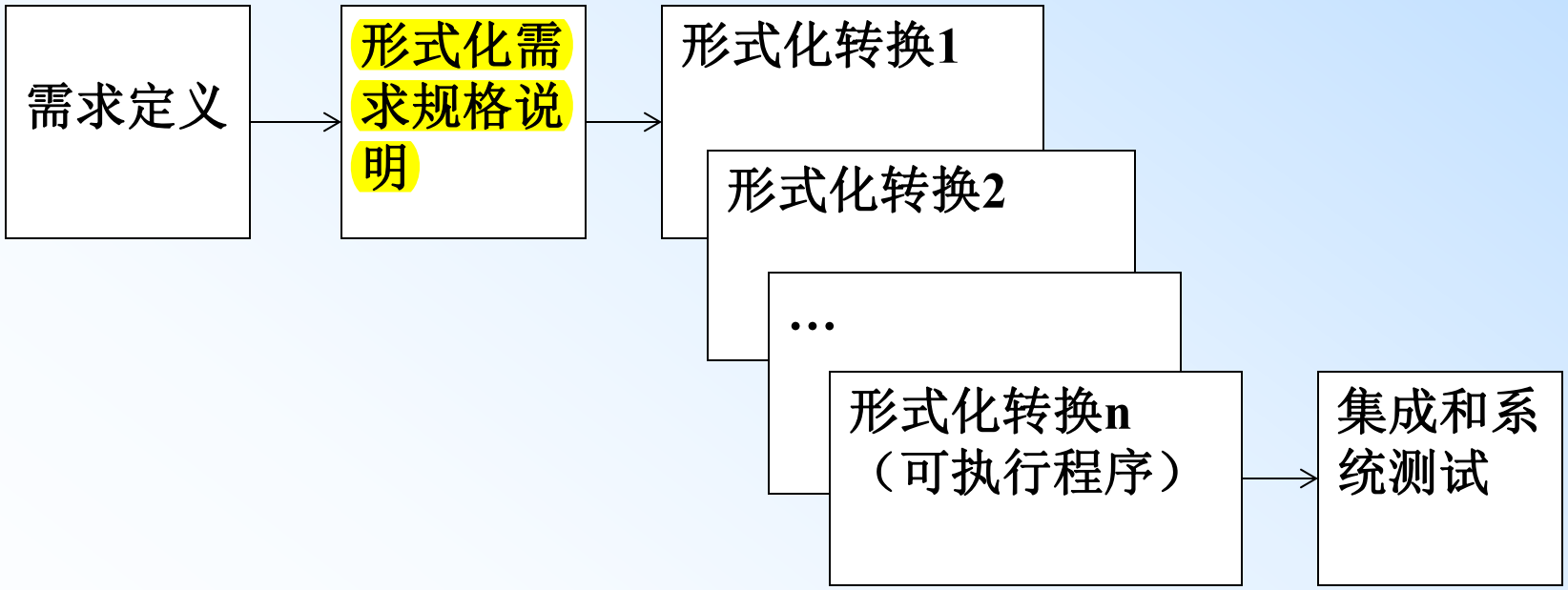
扩展内容

形式化方法模型

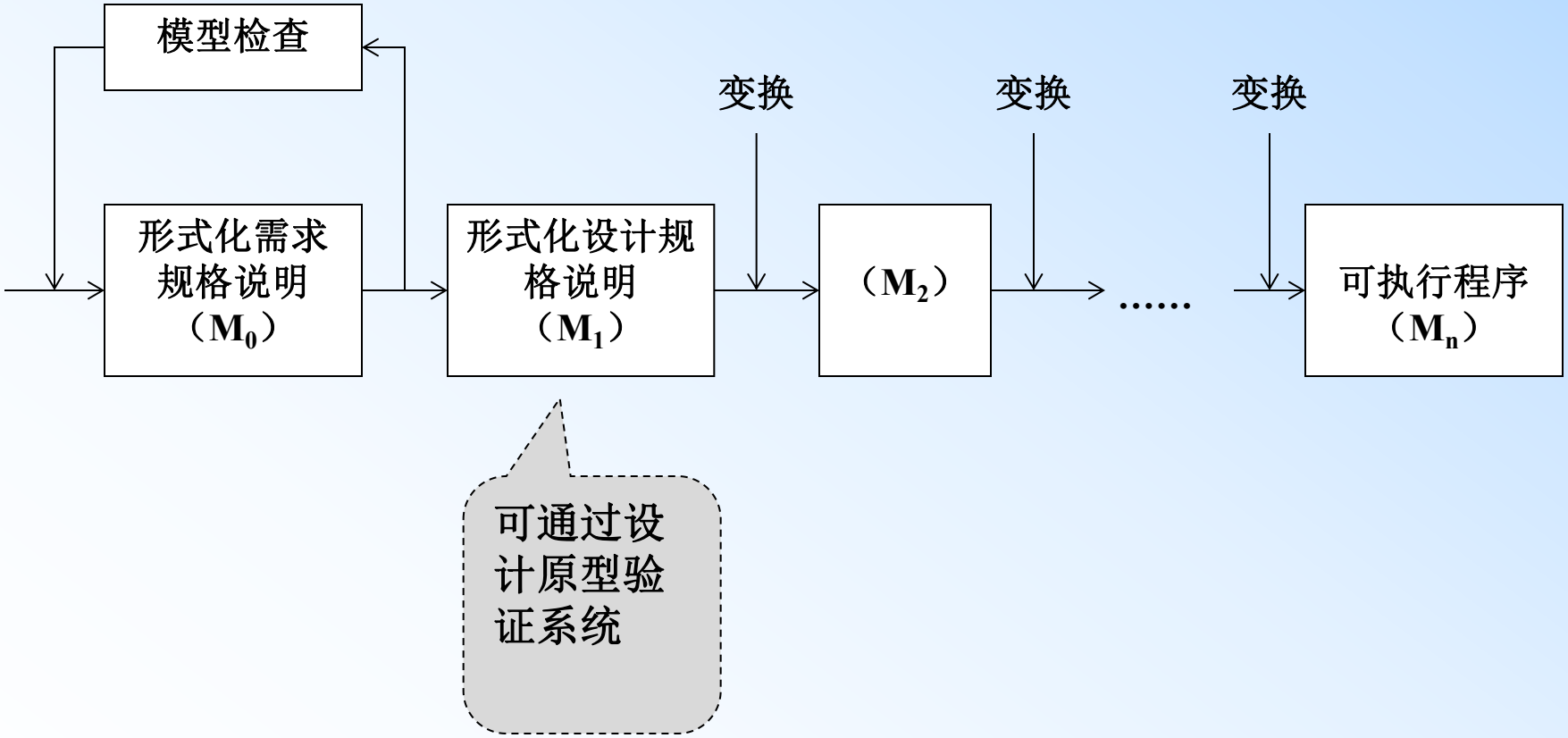
- Formal methods model
- 形式化方法：
  - 基于数学的方法
  - 运用数学符号对系统进行描述、开发和验证
  - 例如：有穷状态机、Petri网、Z语言

形式化方法模型

- 通过形式化数学变化，将形式化系统需求规格说明书转换成可执行程序。



形式化方法模型


模型检查
变换
变换
变换
形式化需求规格说明（M_0）
形式化设计规格说明（M_1）
（M_2）
……
可执行程序（M_n）
可通过设计原型验证系统

形式化方法模型

- 优点：
  - 由于采用形式化的数学方法，能够解决歧义性、不完整性、不一致性问题，可以提供无缺陷的软件
  - 不需要测试来寻找错误，系统测试是为了评估可靠性

- 缺点：
  - 形式化方法难于掌握
  - 形式化描述和转换费时费力，成本很高
  - 对于一般客户，很难用这种方法进行沟通

形式化方法模型

适用场合

适用于对安全性和可靠性要求极高的软件系统开发，如飞行器和医疗器械的控制系统。

净室软件工程

扩展内容

- Cleanroom software engineering
- IBM，1987年提出
- 采用增量开发，每一个增量都要给出形式化描述，此描述经过变换得以实现
- 软件正确性通过形式化方法得以证明
- 不进行单元测试，系统测试的重心是评估可靠性
- 目标是零缺陷软件

面向方面的软件开发

- Aspect-Oriented Software Development:AOSD
- 为定义、说明、设计和构建方面/Aspect提供过程和方法
- 关注点（Concern)：
  - 客户需要的属性或者技术兴趣点
  - 安全性、容错能力、商业规则的应用、任务同步等
- 横切关注点（Crosscutting concern)
  - 涉及到系统多个方面的功能、特性和信息的关注点
- 方面（Aspect）：
  - 对整个软件体系结构产生影响的横切关注点
  - 如：用户接口、协同工作、存储器管理、事务处理、安全、完整性
- 尚不成熟

Rational统一过程

- 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）

Rational统一过程

- 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

适用场合　适合大团队大项目。

Rational统一过程

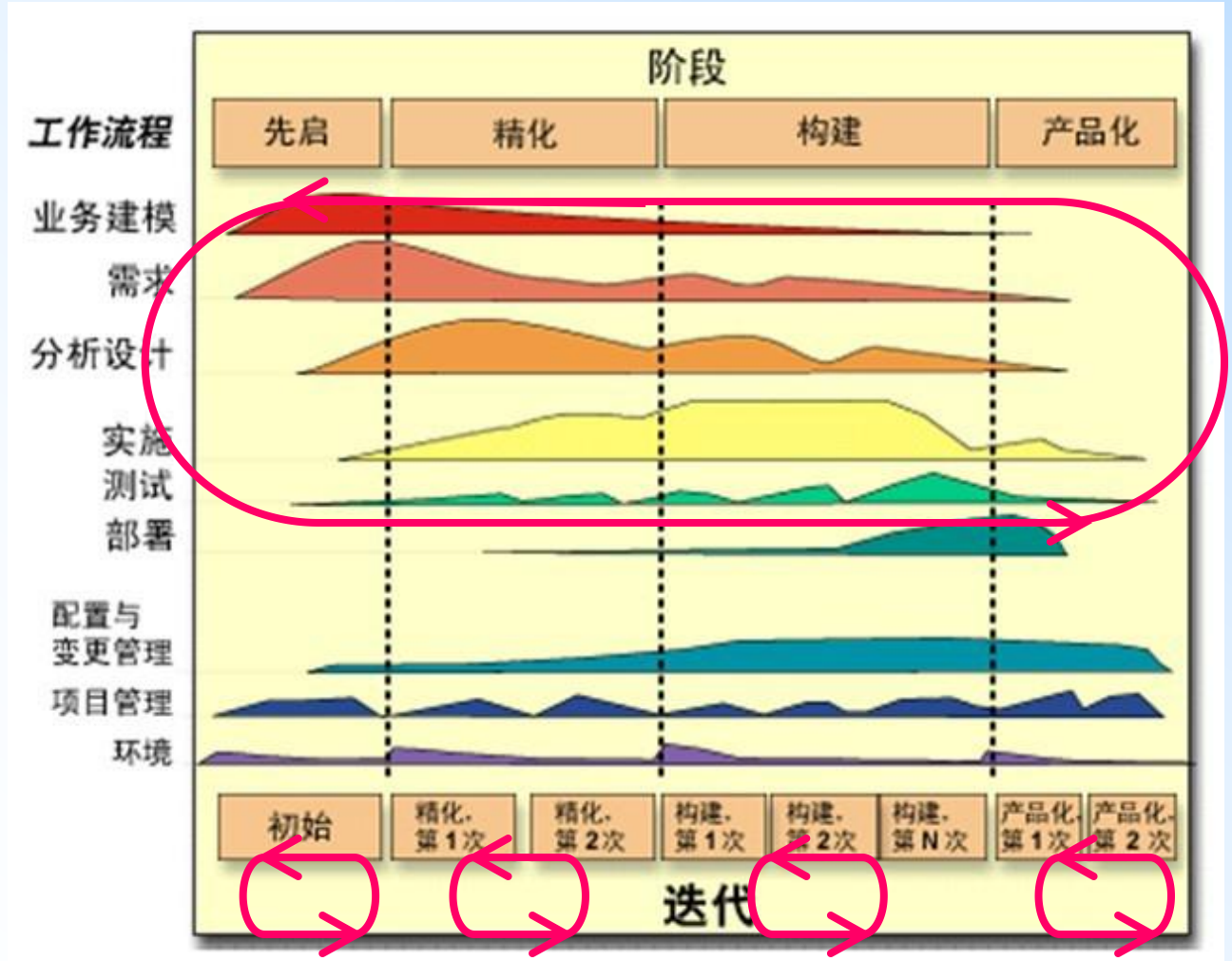
- 实践视角：6条最佳实践
  1. 迭代式开发
     - 需求变更不可避免
     - 每次迭代产生一个可交付版本，用户反馈，减少风险
     - 根据客户的轻重缓急来规划增量，先开发和交付优先级最高的增量
  2. 管理需求
     - 采用用例分析来捕获需求，由用例驱动设计和实现
     - 对需求及其变更进行管理

Rational统一过程

3. 使用基于构件的体系结构
   - 将体系结构组建成基于构件的
   - 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
   - 软件质量评估贯穿于整个开发过程的所有活动中
   - 全体成员参与
6. 控制软件变更
   - 描述了如何控制和跟踪软件的变更

Rational统一过程

静态视角



动态视角

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。

统一过程的静态结构

- 在Rational统一过程的开发流程中定义了“谁”“何时”“如何”做“某事”，并分别使用四种主要的建模元素来进行表达：
  - ① 角色(Workers)，代表了“谁”来做？
  - ② 活动(Activities)，代表了“如何”去做？
  - ③ 产物(Artifacts)，代表了要做“某事”？
  - ④ 工作流（Workflows），代表了“何时”做？
    - 6个核心工程工作流分别为：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个核心支持工作流分别为：项目管理工作流，配置与变更管理工作流，环境工作流。

统一过程的动态结构

- Rational统一过程的动态结构，是通过对迭代式软件开发过程的周期、阶段、迭代过程以及里程碑等的描述来进行表示的。
- 迭代过程分成4个连续阶段
  - ① 初始阶段（Inception）：项目计划、评估风险
  - ② 细化阶段（Elaboration）：设计系统的体系结构、制定项目计划、确定资源需求
  - ③ 构建阶段（Construction）：开发出所有组件和应用程序，集成并进行详尽测试；
  - ④ 移交（Transition）：将产品移交给用户

敏捷软件开发

- Agile software development
- 高效工作、快速响应变化
- 2001年2月，17位编程大师发表敏捷软件开发宣言

① 个体和交互胜过过程和工具

② 可以工作的软件胜过面面俱到的文档

③ 客户合作胜过合同谈判

④ 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

敏捷开发方法

扩展内容

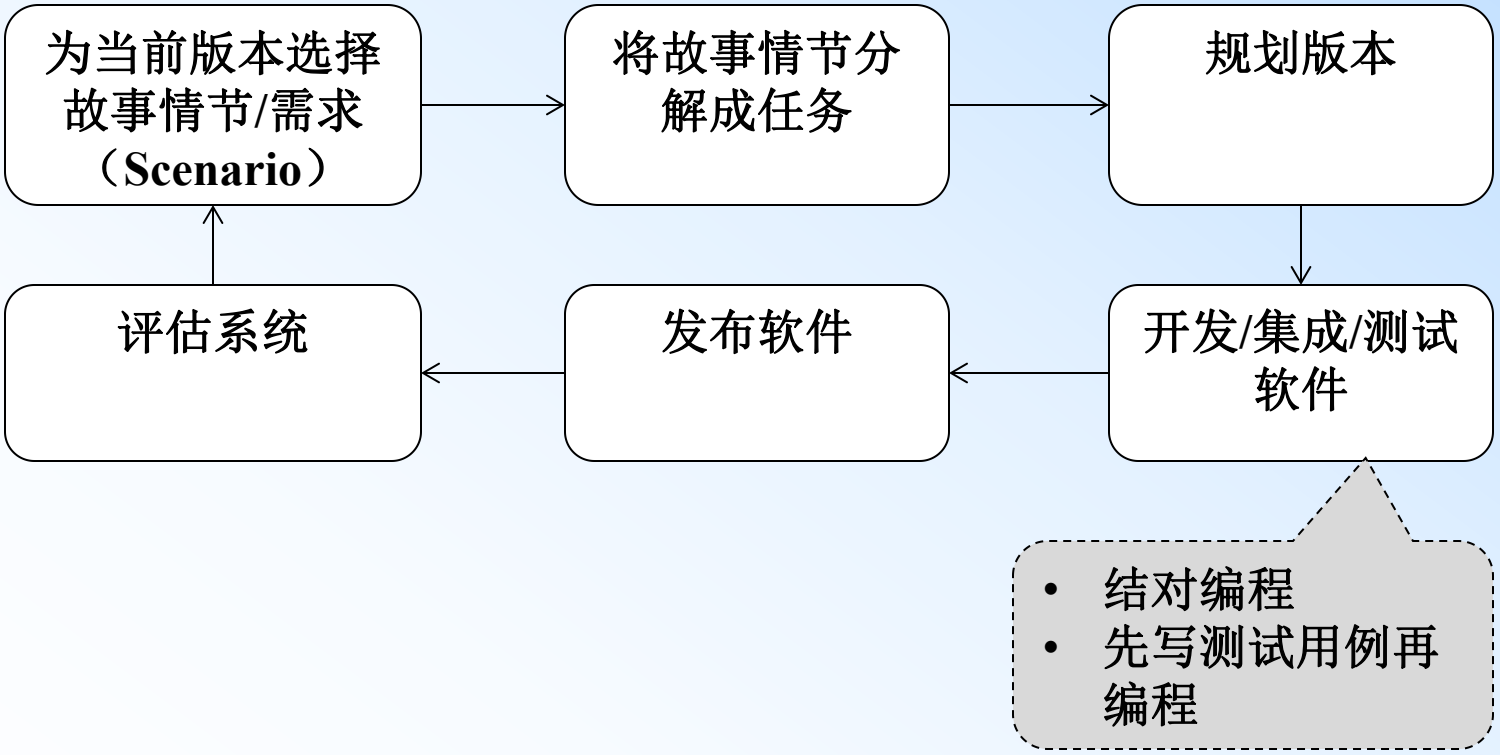
- 极限编程：eXtreme Programming/XP
- 自适应软件开发
  Adaptive Software Development/ASD
- 并列争球法：Scrum
- 动态系统开发方法
  Dynamic System Development Method/DSDM
- 水晶法：Crystal
- 特征驱动开发：Feature-Driven Development/FDD
- 精益软件开发：Lean Software Development/LSD
- …

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。
- 开发准则强调分析和设计的交付，以及开发者和客户之间积极持续的交流。

极限编程

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



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

敏捷开发的优点和缺点

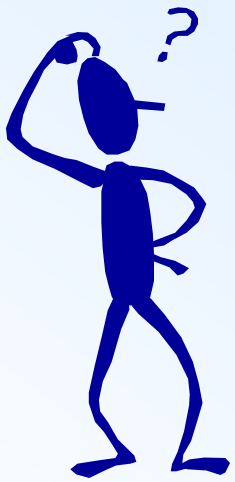
- 优点：
  - 对变化和不确定性有更快速更敏捷的反应
  - 在快速的同时保持可持续的开发速度
  - 能较好的地适应商业竞争环境下对小项目提出的有限资源和有限开发时间的约束
- 缺点：
  - 极限编程中的测试驱动开发可能会导致系统通过了测试但不是用户期望的
  - 重构而不降低系统体系结构的质量是困难的

敏捷开发的适用场合

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。

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
- 也可根据实际创造新的模型

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

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

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

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

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

本章复习要点

- 软件生命周期、软件过程、软件过程模型
- 通用软件过程模型、过程流
- 能力成熟度模型CMM
- 传统软件过程模型
  - 瀑布模型、增量模型、原型模型、螺旋模型、协同模型、喷泉模型
- 现代软件过程模型
  - 基于构件的开发模型、形式化方法模型、面向方面的软件开发、Rational统一过程、敏捷软件开发

作业

1. 比较分析瀑布模型、增量模型、原型模型和螺旋模型的联系和区别。
2. 分别举出使用瀑布模型、原型模型和增量模型的软件项目的例子（使用课件或教材中例子0分）。
3. 为什么增量式开发适合商务软件？它适合实时控制系统吗？
4. 一个软件组织是否应该对它所有的软件开发都采用同一种软件过程模型？讨论它的利与弊。

请在两周内提交作业纸质版！