软件工程

第一部分 软件过程

徐本柱 软件学院

2018-09

- 什么是软件过程？
- 有哪些通用框架活动？
- 建立过程模型？过程模式？
- 惯用过程模型？优缺点？
- 敏捷过程模型？
- 敏捷软件开发？

第3章　软件过程结构

软件过程[BAE98]

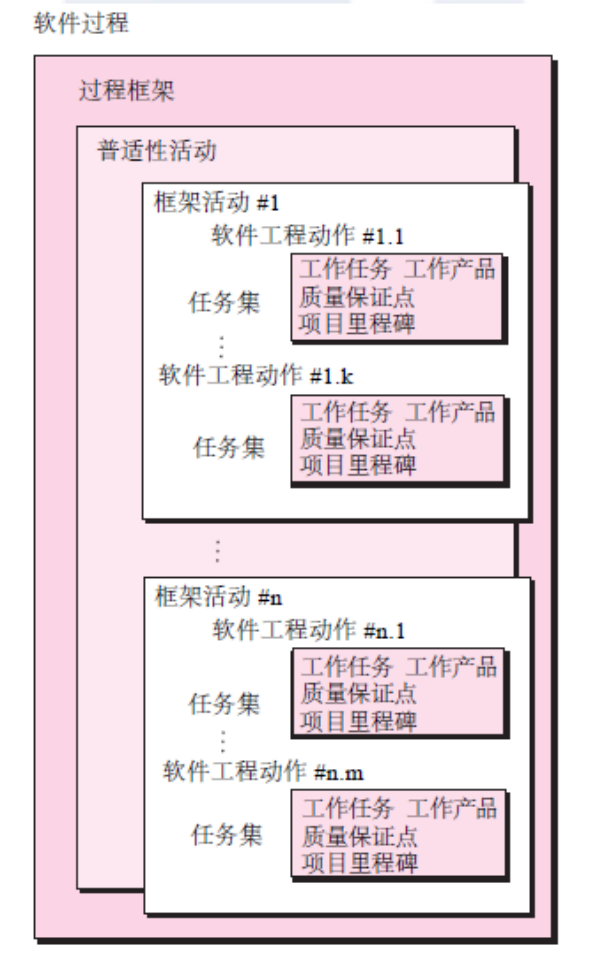
- 软件是知识的具体体现
- 知识最初都是以分散的、不明确的、隐蔽的且不完整的形式广泛存在的
- 软件开发是一个社会学习的过程。
- 软件过程是一个对话，在对话中，软件所必需的知识被收集在一起并在软件中实现。
- 过程**提供**了用户与设计人员之间、用户与不断演化的工具之间以及设计人员与不断演化的工具（技术）之间的交互途径。
- 软件开发是一个迭代的过程，在这个过程中，演化的工具本身就作为沟通的媒介，每新一轮对话都可以从参与的人员中获得更有用的知识。

软件过程

- 软件过程可定义为一个为创建高质量软件
  - 所需要完成的**活动**、**动作**和**任务**的框架。
  - 软件过程定义了软件开发中采用的方法，
  - 但软件工程还包含该过程中应用的技术——（技术方法和自动化工具）。
- 软件工程是由有创造力、用知识的人完成的
  - 根据产品的构建需要和市场需求选取成熟的软件过程。

3.1 通用过程模型

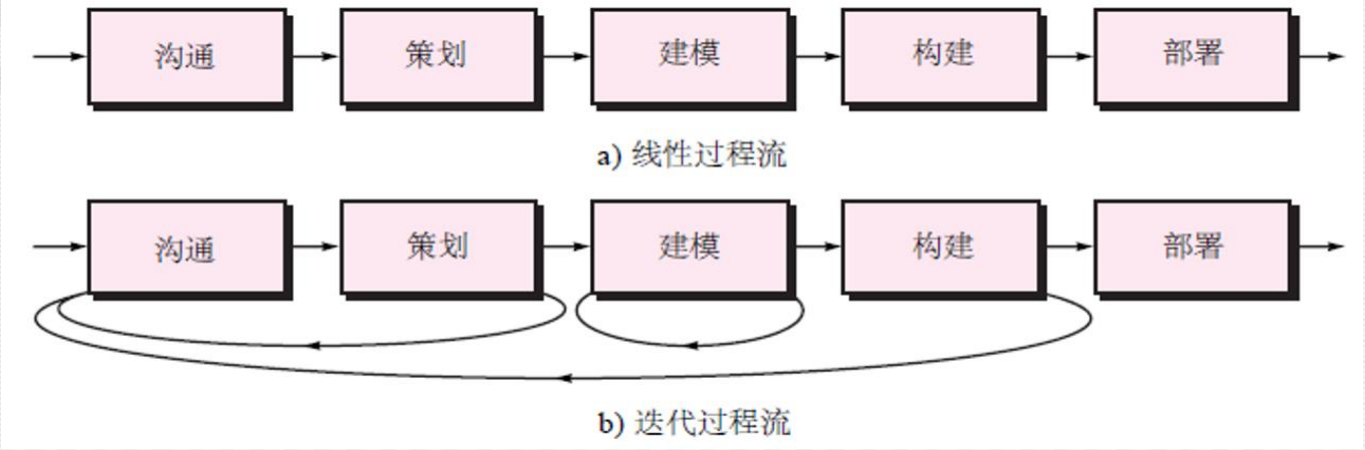
- 框架活动由一系列软件工程动作构成
- 软件工程动作由任务集合来定义，
- 任务集合明确
  - 将要完成的工作任务
  - 将要产生的工作产品
  - 所需要的质量保证点，
  - 表明过程状态的里程碑。

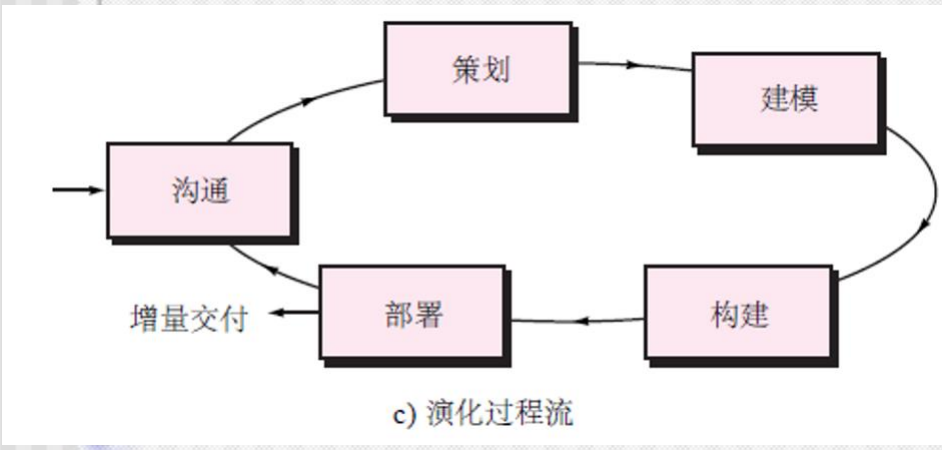


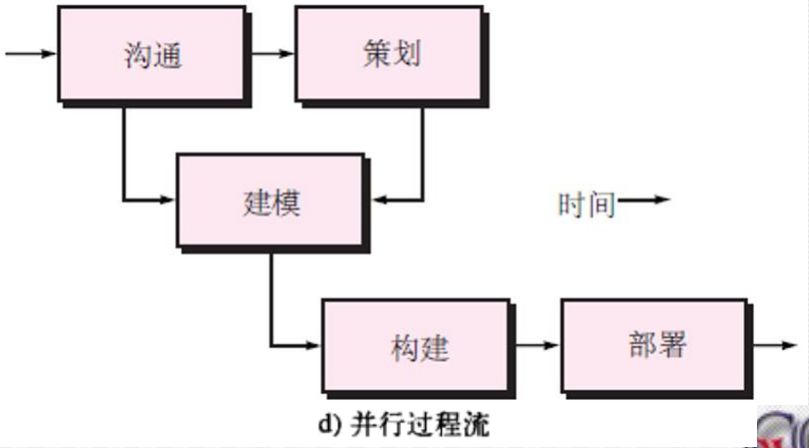
软件活动

- 通用过程框架定义了五种框架活动
  - 沟通、策划、建模、构建以及部署。
- 普适性活动贯穿软件过程始终。
  - 项目跟踪控制、风险管理、质量保证、配置管理、技术评审以及其他活动
- 过程流描述在执行顺序和执行时间上如何组织框架中的活动、动作和任务
  - 线性、迭代、演化、并行过程流

过程流


沟通
策划
建模
构建
部署
a) 线性过程流
沟通
策划
建模
构建
部署
b) 迭代过程流


策划
建模
沟通
增量交付
部署
构建
c) 演化过程流


沟通
策划
建模
时间
构建
部署
d) 并行过程流

3.2 定义框架活动

- 5种通用框架活动定义
  - 沟通、策划、建模、构建和部署
- 但软件团队要在软件过程中
  - 具体执行这些活动中的任何一个
  - 还需要更多信息
- 一个关键问题：
  - 针对给定的问题、开发人员和利益相关者
  - 哪些动作适合于框架活动？

3.3 明确任务集

- 任务集定义了
  - 为达到一个软件工程动作的目标所需要完成的工作
- 每一个软件工程动作都由若干个任务集构成，
- 每一个任务集都
  - 由工作任务、工作产品、质量保证点和项目里程碑组成
- 选择最满足项目需要和适合开发团队特点的任务集
- 软件工程动作可以
  - 根据软件项目的特定需要和开发队伍的特点
  - 作适当的调整。

小型任务集（需求获取）

对于一个小型、相对简单的项目而言，获取需求的任务集可能包括：

1. 制定项目的利益相关者列表。
2. 邀请所有的利益相关者参加一个非正式会议。
3. 征询每一个人对于软件特征和功能的需求。
4. 讨论需求，并确定最终的需求列表。
5. 划定需求优先级。
6. 标出不确定领域。

大型任务集（需求获取）

对于大型、复杂的软件工程项目而言，可能需要如下不同的任务集：

1. 制定项目的利益相关者列表。
2. 和利益相关者的每一个成员分别单独讨论，获取所有的要求。
3. 基于任务集2中的调查，建立初步的功能和特征列表。
4. 安排一系列促进需求获取的会议。
5. 组织会议。
6. 在每次会议上建立非正式的用户场景。
7. 根据利益相关者的反馈，进一步细化用户场景。
8. 建立一个修正的需求列表。
9. 使用质量功能部署技术，划分需求优先级。
10. 将需求打包以便于软件可以增量交付。
11. 标注系统的约束和限制。
12. 讨论系统验证方法。

3.4 过程模式

- **软件过程**可以定义为**一系列模式的组合**，
  - 过程模式描述软件工程工作中遇到的过程相关问题
    - 明确了问题环境，并给出针对该问题一种/几种可证明的解决方案
    - 定义了一系列的软件开发中需要的活动、动作、工作任务、工作产品及其相关的行为
  - 过程模式提供了一个模板
    - 一种在软件工程背景下统一描述问题解决方案的方法。
  - 通过模式组合，软件团队可以解决问题，并且定义最符合项目需求的开发过程。

过程模式类型

- *步骤模式(Stage patterns)*——定义了与过程的框架活动相关的问题。
- *任务模式(Task patterns)*——定义了与软件工程动作或是工作任务相关、关系软件工程实践成败的问题。
- *阶段模式(Phase patterns)*——定义在过程中发生的框架活动序列，即使这些活动流本质上是迭代的。

过程模式模板

- 模式名称：应能清楚地表述该模式在软件过程中的功能。
- 驱动力：模式使用环境及主要问题， 以明确主要难点并可能影响解决方案。
- 类型：定义模式类型（步骤模式、任务模式、阶段模式）
- 启动条件：描述模式应用的前提条件。
- 问题：描述模式将要解决的问题。
- 解决办法：描述模式的实现。
- 结束条件：描述模式成功执行之后的结果。
- 相关模式：以层次或其他图的方式列举与该模式相关的其他模式。
- 已知应用实例：介绍该模式的具体实例。

过程模式实例

一个过程模式的例子

当利益相关者对工作成果有大致的想法，但对具体的软件需求不确认时，下述简化的过程模式描述了可采用的方法。

模式名称。需求不清。

目的。该模式描述了一种构建模型（或是原型系统）的方法，使得利益相关者可以反复评估，以便识别和确定软件需求。

类型。阶段模式。

启动条件。在模式启动之前必须满足以下四个条件：（1）确定利益相关者；（2）已经建立起利益相关者和软件开发队伍之间的沟通方式；（3）利益相关者确定了需要解决的主要问题；（4）对项目范围、基本业务需求和项目约束条件有了初步了解。

问题。需求模糊或者不存在，但都清楚地认识到项目存在问题，且该问题需要通过软件解决。利益相关者不确认他们想要什么；即他们无法详细描述软件需求。

解决办法。描述了原型开发过程，详见第2.3.3节。

结束条件。开发了软件原型，识别了基本的需求（例如，交互模式、计算特征、处理功能等），并获得了利益相关者的认可。随后，可能有两种结果：（1）原型系统可以通过一系列的增量开发，演化成为软件产品；或是（2）原型系统被抛弃，采用其他过程模式建立产品软件。

相关的模式。以下模式与该模式相关：**客户沟通、迭代设计、迭代开发、客户评价、需求抽取。**

已知应用和实例。当需求不确定时，推荐原型开发方法。

第4章　过程模型

4.1 惯用过程模型

- 惯用过程模型力求达到软件开发的结构和秩序

  这将产生一些问题

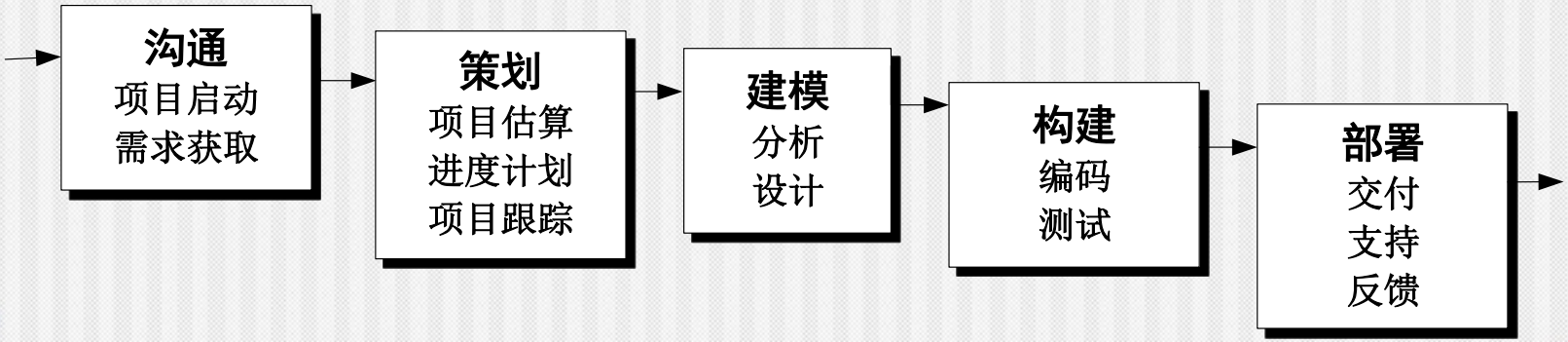
- 如果惯用过程模型力求达到软件开发的结构和秩序，那么，对于富于变化的软件世界，这一模型是否适合呢？
- 如果我们抛弃传统过程模型（以及模型所规定的秩序），取而代之以一些不够结构化的模型，是否会使软件工作无法达到协调和一致？

惯例过程模型

- 惯例过程模型规定了一套过程元素
  - 框架活动、软件工程动作、任务、工作产品、质量保证以及每个项目的变更控制机制
  - 每个过程模型还定义了过程流
- 所有软件过程模型都支持通用框架活动，
- 但每一个模型都对框架活动有不同的侧重，
- 并定义了不同的工作流如何以不同的方式执行每一个框架活动。

4.1.1 瀑布模型

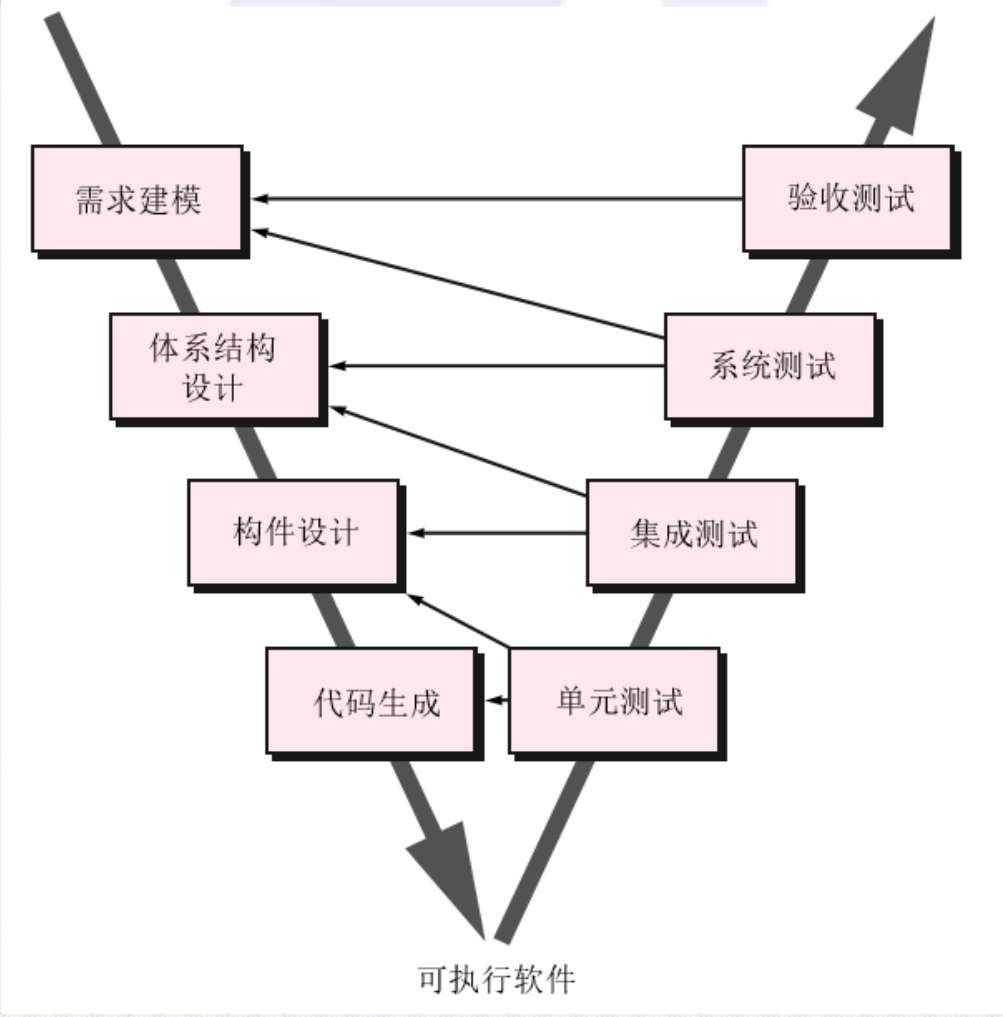
- 瀑布模型(the waterfall model)，又被称为经典生命周期
- 提出了一个系统的、顺序的软件开发方法，
- 从用户需求规格说明开始，通过策划、建模、构建和部署的过程，
- 最终提供一个完整的软件并提供持续的技术支持。



瀑布模型基本思想

- 将软件开发过程划分为
  - 分析、设计、编码、测试等阶段。
- 软件开发要遵循过程规律，按次序进行
- 每个阶段均有里程碑和提交物。
- 工作以线性方式进行，上一阶段的输出是下一阶段的输入。

瀑布模型变体——V模型



瀑布模型的优点

- 简单、易懂、易用。
- 为项目提供了按阶段划分的检查点，项目管理比较规范。
- 每个阶段必须提供文档，而且要求每个阶段的所有产品必须进行正式、严格的技术审查。

运用瀑布模型遇到的问题

- 实际的项目很少遵守瀑布模型提出的顺序。
- 客户通常难以清楚地描述所有的需求
- 只有在项目接近尾声的时候，才能得到可执行的程序。

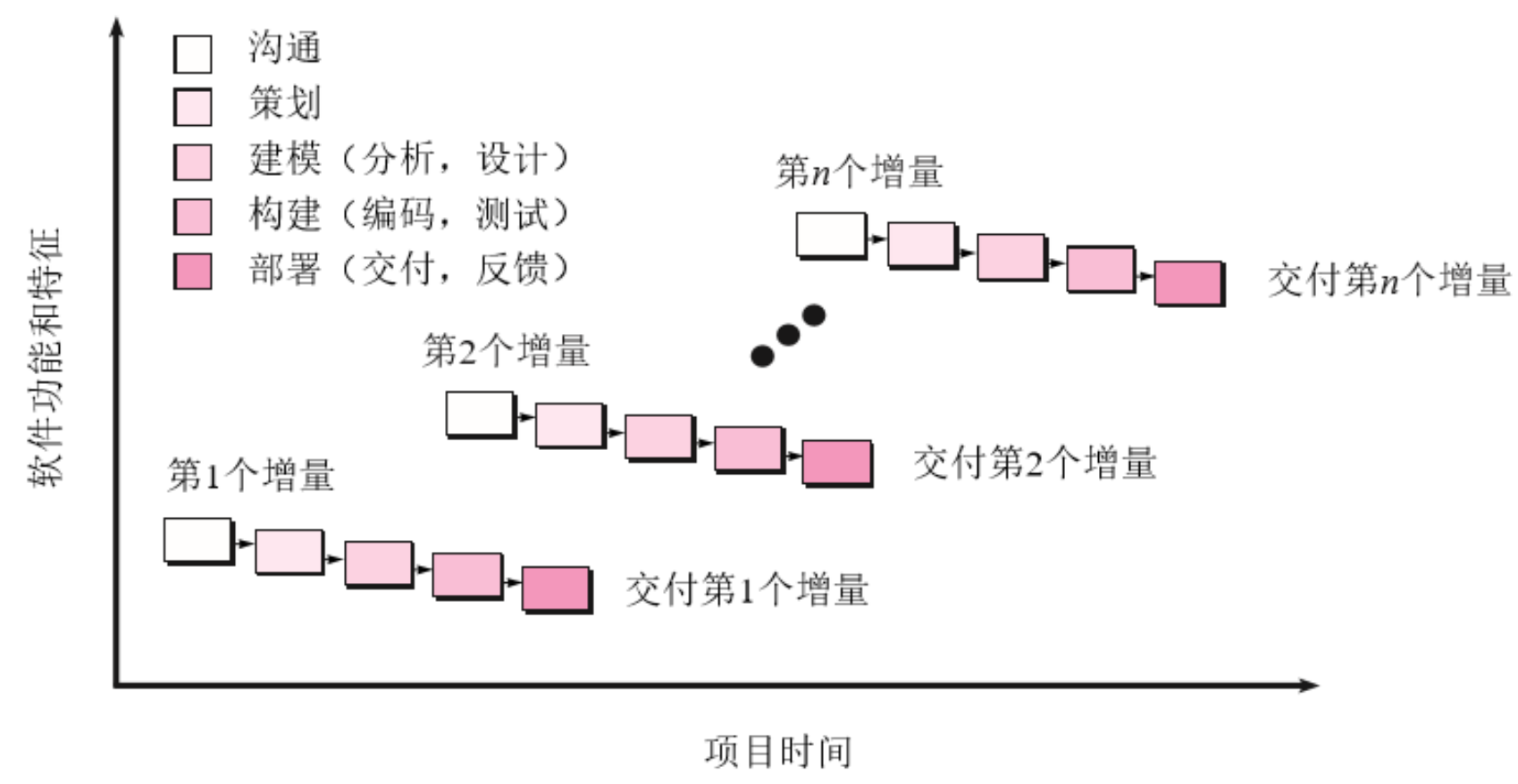
瀑布模型的适用场合

- 需求相当稳定，客户需求被全面的了解风险管理。
- 开发团队对于这一应用领域非常熟悉。
- 外部环境的不可控因素很少。
- 小型清晰的项目或长周期的项目。

4.1.2 增量过程模型

- 增量模型综合了线性过程流和并行过程流的特征
- 随着时间的推移，增量模型在每个阶段运用线性序列。
- 每个线性序列生产出一个软件的可交付增量。

增量模型


沟通
策划
建模（分析，设计）
构建（编码，测试）
部署（交付，反馈）
第n个增量
交付第n个增量
第2个增量
交付第2个增量
第1个增量
交付第1个增量
软件功能和特征
项目时间

增量模型的使用方法

- 软件被作为一系列的增量来进行开发，每一个增量都提交一个可以操作的产品，可供用户评估
- 第一个增量往往是核心产品：
  - 满足了基本的需求，但是缺少附加的特性。
- 客户使用上一个增量的提交物并进行自己评价，制定下一个增量计划，说明需要增加的特性和功能。
- 重复上述过程，直到最终产品产生为止。

增量模型应用举例

- 应用举例：开发一个类似于Word的字处理软件。
  - 增量1：提供基本的文件管理、编辑和文档生成功能。
  - 增量2：提供高级的文档编辑功能。
  - 增量3：实现拼写和语法检查功能。
  - 增量4：完成高级的页面排版功能。

增量模型的优点

- 提高对用户需求的响应：用户看到可操作的早期版本后会提出一些建议和需求，可以在后续增量中调整。
- 人员分配灵活：如果找不到足够的开发人员，可采用增量模型，早期的增量由少量人员实现，如果客户反响较好，则在下一个增量中投入更多的人力。
- 可规避技术风险：不确定的功能放在后面开发。

增量模型存在的问题

- 每个附加的增量并入现有的软件时，必须不破坏原来已构造好的东西。
- 加入新增量时应简单、方便 ——该类软件的体系结构应当是开放的。
- 仍然无法处理需求变更的情况。
- 管理人员须有足够的技术能力来协调好各增量之间的关系。

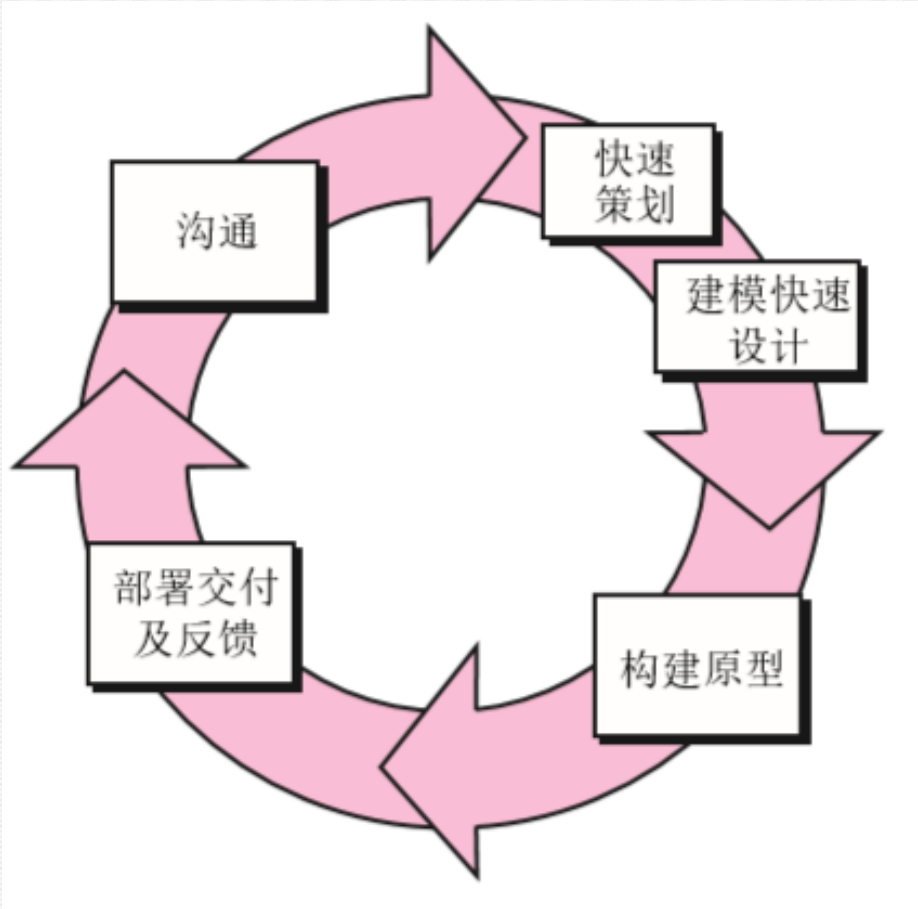
4.1.3 演化过程模型

- 演化模型是迭代的过程模型，使得软件工程师能够逐步开发出更完整的软件版本。
- 迭代的思想：
  - 对一系列活动的重复应用，用以评估一系列论断
  - 解决一系列风险，达成一系列开发目标，
  - 逐步增量地建立并完善一个有效的解决方案。
- 常见的演化过程模型有原型开发、螺旋模型等

原型开发模型

- 客户提出了软件的一些基本功能，但是没有详细定义输入、处理和输出需求。
- 另一种情况下，开发人员可能对算法的效率、操作系统的兼容性和人机交互的形式等情况不确定。
- 在这些情况下，原型开发范型是较好的解决方法。

原型开发模型



原型开发模型

原型模型的使用方法

- 使用方法一：
  - 步骤1：开发人员与其他利益相关者沟通，定义软件的整体目标，明确已知的需求，并大致勾画出以后再进一步定义的东西
  - 步骤 2：迅速策划一个原型开发迭代并进行建模（快速设计）集中在最终用户能够看到的方面（如人机接口布局或输出显示格式等）；
  - 步骤3：快速设计产生一个原型，对原型进行部署，由利益相关者进行评估；
  - 步骤4：根据反馈信息，抛弃掉不合适的部分，进一步精炼软件的需求；
  - 步骤5：原型系统不断调整，逐步清楚用户的需求。
- 使用方法二：
  - 作为需求分析的工具，明确需求后，原型系统被抛弃。

原型开发应用举例

- 应用举例：开发一个教务管理系统。
  - 第一次迭代：完成基本的学籍管理、选课和成绩管理功能。（6周）
  - 客户反馈：基本满意，但是对大数据量运行速度慢效率，不需要学生自己维护学籍的功能等。
  - 第二次迭代：修改细节，提高成绩统计和报表执行效率（2周）。
  - 客户反馈：需要严格的权限控制，报表打印格式不符合要求。
  - 第三次迭代：完善打印和权限控制功能。（2周）
  - 客户反馈：可以进行正式应用验证。

原型开发的优点

- 快速开发出可以演示的系统，方便沟通
- 采用迭代技术，逐步弄清客户的需求。

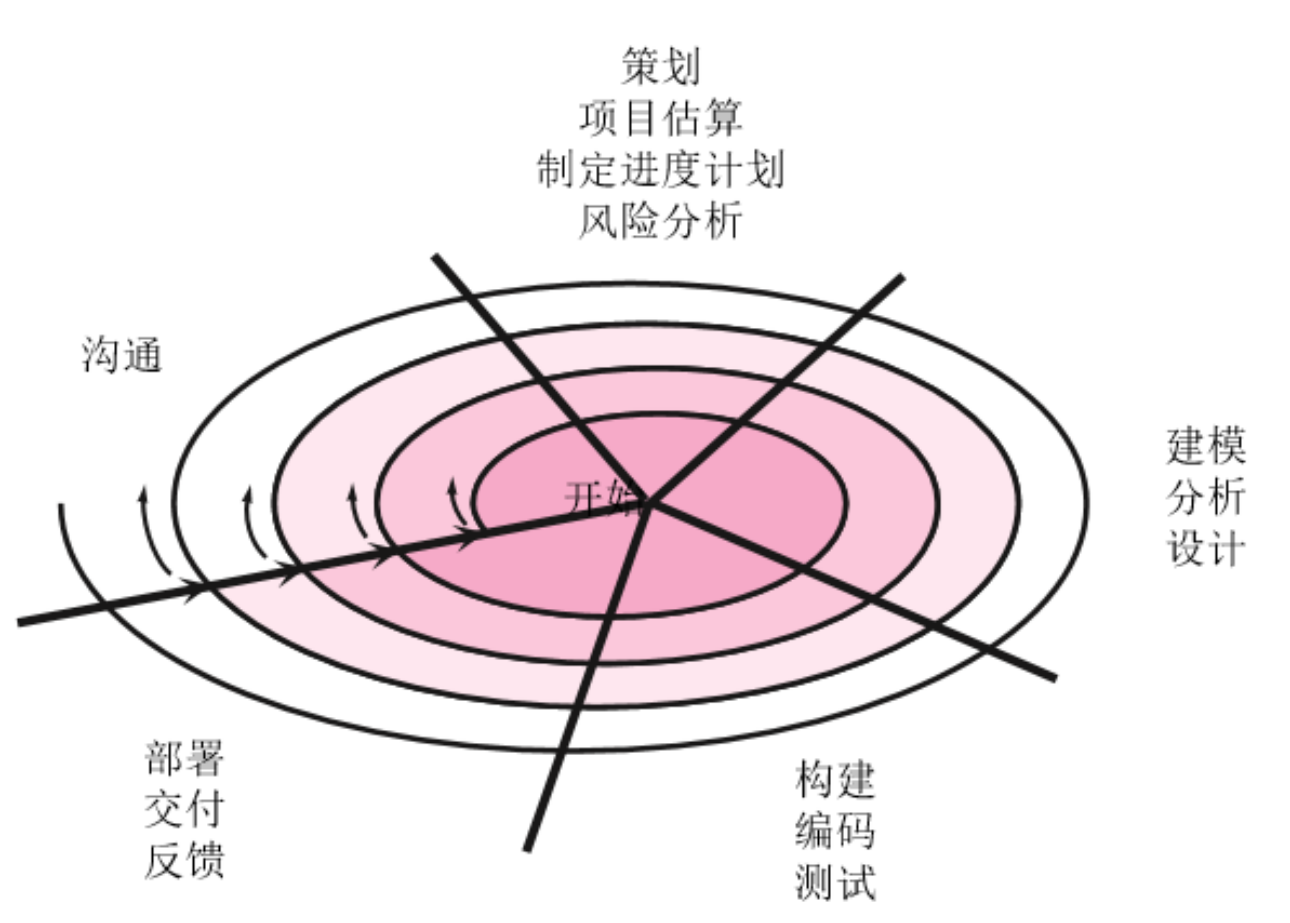
原型开发存在的问题

- 为了尽快完成原型，
  - 没有考虑整体软件的质量和长期的可维护性
  - 系统结构通常较差。
- 用户可能混淆原型系统和最终系统
  - 原型系统在完全满足用户需求之后可能会被直接交付给客户使用。

螺旋模型

- 一种风险驱动型过程模型生成器，
- 可以指导多个利益相关者的协同工作
- 两个显著的特点：
  - 一是采用循环的方式逐步加深系统定义和实现的深度，同时降低风险
  - 二是确定一系列里程碑，确保利益相关者都认可是可行的、令各方满意的系统解决方案

螺旋模型



螺旋模型

螺旋模型的优点

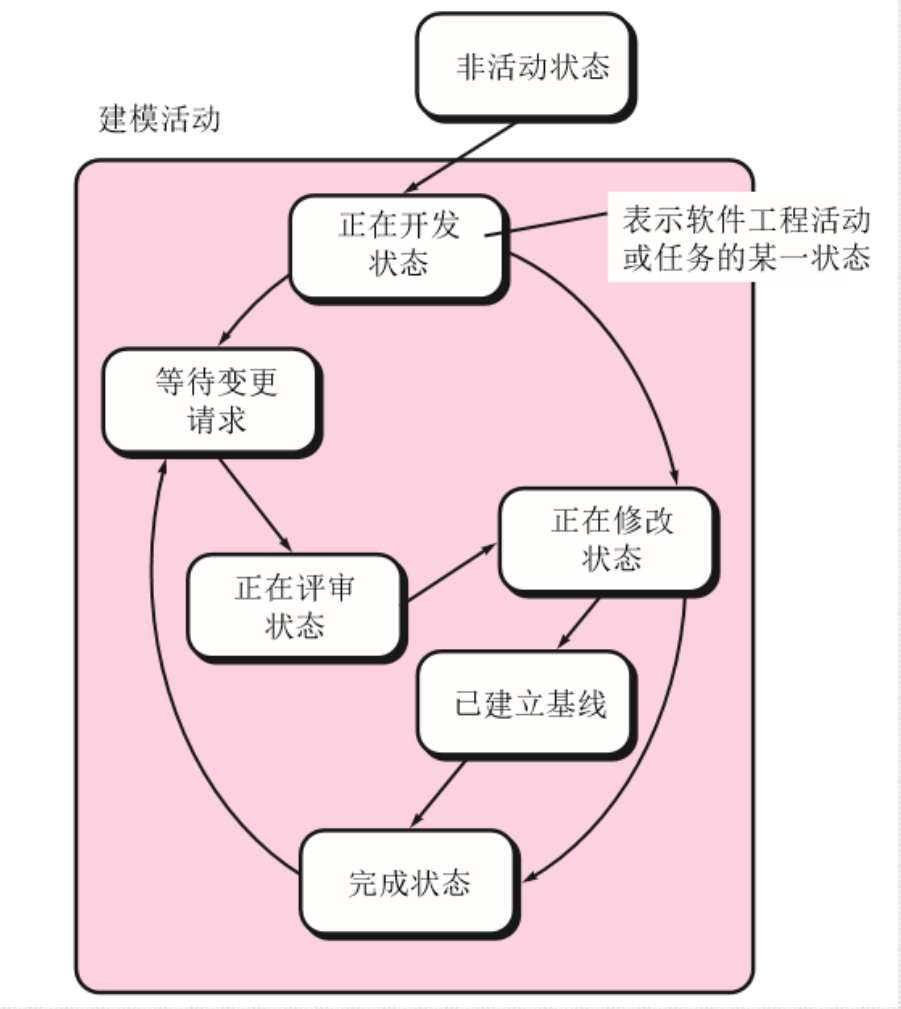
- 结合了原型的迭代性质与瀑布模型的系统性和可控性，是一种风险驱动型的过程模型。
- 采用循环的方式逐步加深系统定义和实现的深度，同时更好地理解、应对和降低风险。
- 确定一系列里程碑，确保各方都得到可行的系统解决方案。
- 始终保持可操作性，直到软件生命周期的结束。
- 由风险驱动，支持现有软件的复用。

螺旋模型存在的问题

- 依赖大量的风险评估专家来保证成功
  - 如果有较大的风险没有被发现和管理，会发生问题。
- 软件开发人员应该
  - 擅长寻找可能的风险，
  - 准确的分析风险，
  - 否则将会带来更大的风险。

4.1.4 并发模型

- 并发开发模型
  - 也叫并发工程
  - 迭代或并发进行



4.1.5 演化过程模型评述

- 软件总是在持续改变，
  - 变更通常要求在非常短的期限内实现
  - 要充分满足客户/用户的要求。
- 及时投入市场是最重要的管理需求。
  - 市场时间错过，软件项目自身可能会变得毫无意义。
- 演化过程模型就是为了解决上述问题的
- 但作为一类通用的过程模型，也有缺点。

演化过程模型评述[NOG00]

- 首先，原型开发（和其他更加复杂的演化过程）由于构建产品需要的周期数目不确定，给项目策划带来了困难。
- 其次，演化软件过程没有确定演进的最快速度。如果演进的速度太快，完全没有间歇时间，项目肯定会陷入混乱；反之，如果演进速度太慢，则会影响生产率……
- 再次，软件过程应该侧重于灵活性和可扩展性，不是高质量。为了追求高质量而延长开发时间势必造成产品推迟交付，从而失去进入市场的良机。

演化过程模型评述

- 演化模型的初衷
  - 采用迭代或者增量的方式开发高质量软件。
- 演化模型可以做到强调
  - 灵活性、可扩展性和开发速度。
- 软件开发团队及其经理所面临的挑战
  - 在这些严格的项目和产品参数与客户满意度之间
  - 找到一个合理的平衡点。

4.2 专用过程模型

- 往往应用面较窄，
- 只适用于某些特定的软件工程方法。
- 专用过程也许更确切地应该称为技术的集合或方法论，
- 是为了实现某一特定的软件开发目标而制定的。
- 但它们确实也提出了一种过程。

4.2.1 基于构件的开发

- 商品化成品软件构件，由厂家作为产品供应，它们可以在构建软件时使用。这些构件通过良好定义的接口提供特定的功能，能够集成到软件中。
- 建模和构建活动开始于识别可选构件。这些构件有些设计成传统的软件模块，有些设计成面向对象的类或软件包。

基于构件的开发

- 基于构件开发模型能够使软件复用，软件复用为软件工程师带来极大收益。有研究报告指出：基于构件开发缩短了**70%**的开发周期，减少了**84%**的项目开销，生产率指数可达到**26.2**，而工业标准值为**16.9**。

4.2.2 形式化方法模型

- 主要活动是生成计算机软件形式化的数学规格说明。
- 形式化方法使软件工程师可以应用严格的数学符号来说明、开发和验证基于计算机的系统。
- 形式化方法提供了一种机制，使得在软件开发中可以避免一些问题，如歧义性问题、不完整问题、不一致问题等。

形式化方法模型的缺点

- 目形式化模型开发非常耗时，成本也很高。
- 只有极少数程序员具有应用形式化方法的背景，因此需要大量的培训。
- 对于技术水平不高的客户，很难用这种模型进行沟通。

形式化方法模型的应用场合

- 用来开发高度关注安全的软件。
- 开发软件出错将导致重大经济损失的软件。

4.2.3 面向方面的软件开发

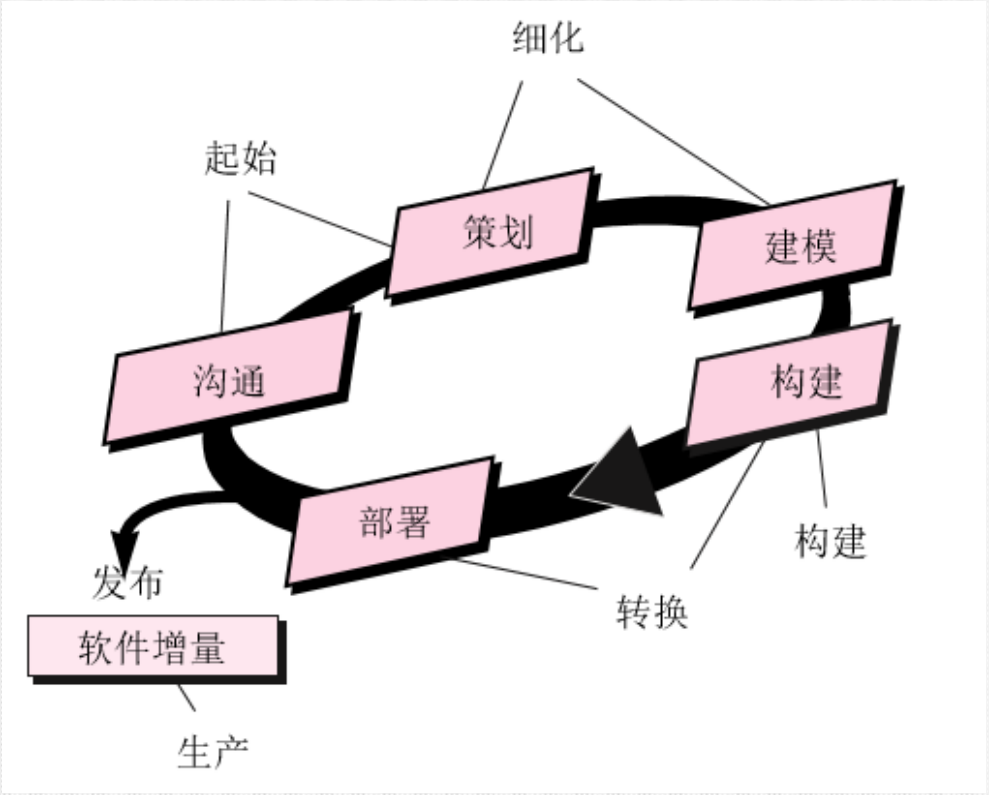
- 复杂软件无一例外地实现了一套局部化的特征、功能和信息内容
- 现代计算机系统变得更加复杂，某些关注点，体现在整个架构设计中。
- 有些关注点是系统的高层属性，还有一些关注点影响了系统的功能，还有一些是系统的。

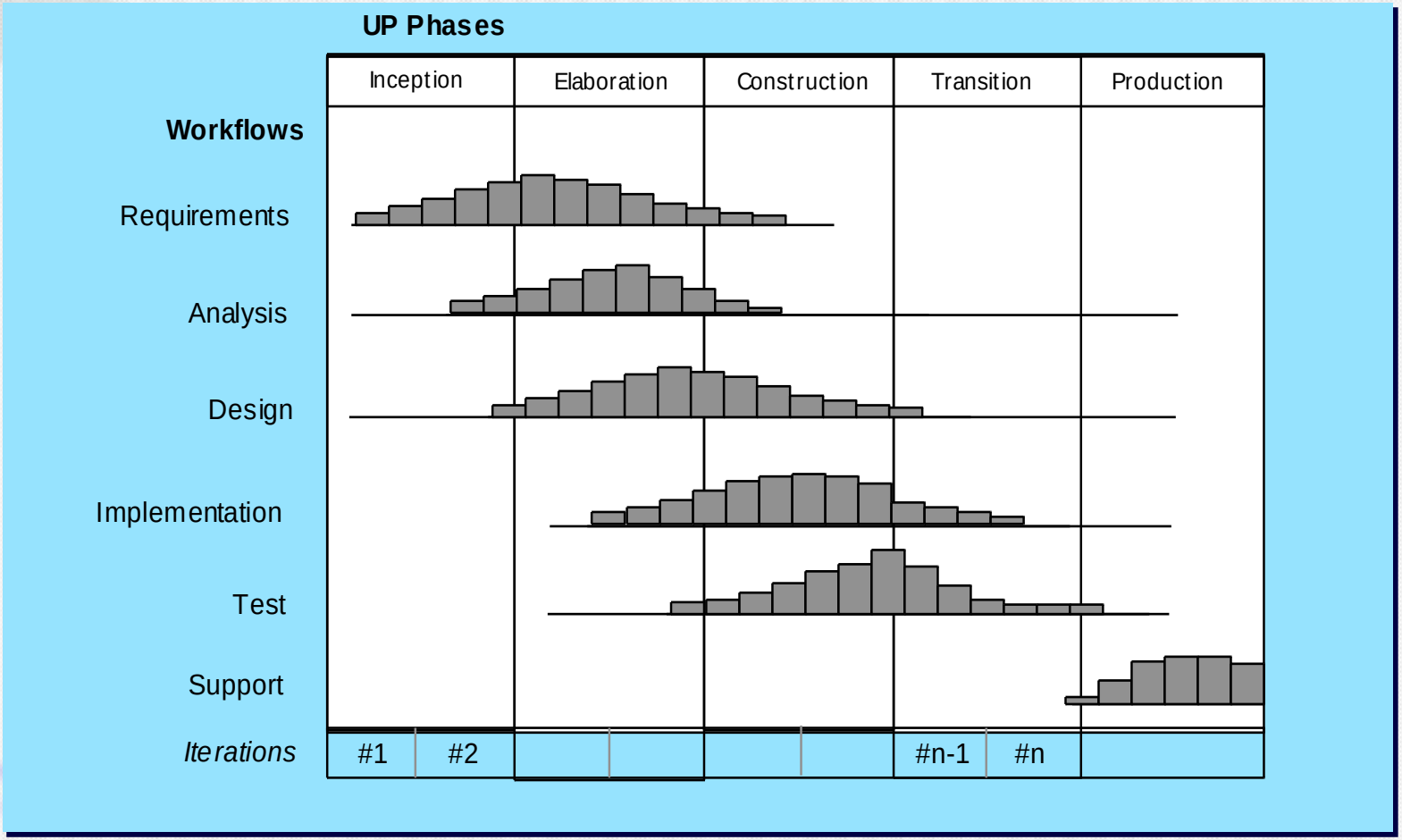
4.3 统一过程(UP)

- Unified Process (UP) 描述了如何为软件开发团队有效的部署经过商业化验证的软件开发方法。
- 迭代开发：软件开发的特性。
- 需求的管理：需求是变化的、持续的。
- 应用基于构件的架构：提高重用、相互独立、适应变化。
- 可视化软件建模：消除理解歧义。
- 持续质量验证：迭代测试、提早发现问题。
- 控制软件变更：软件过程控制与管理。

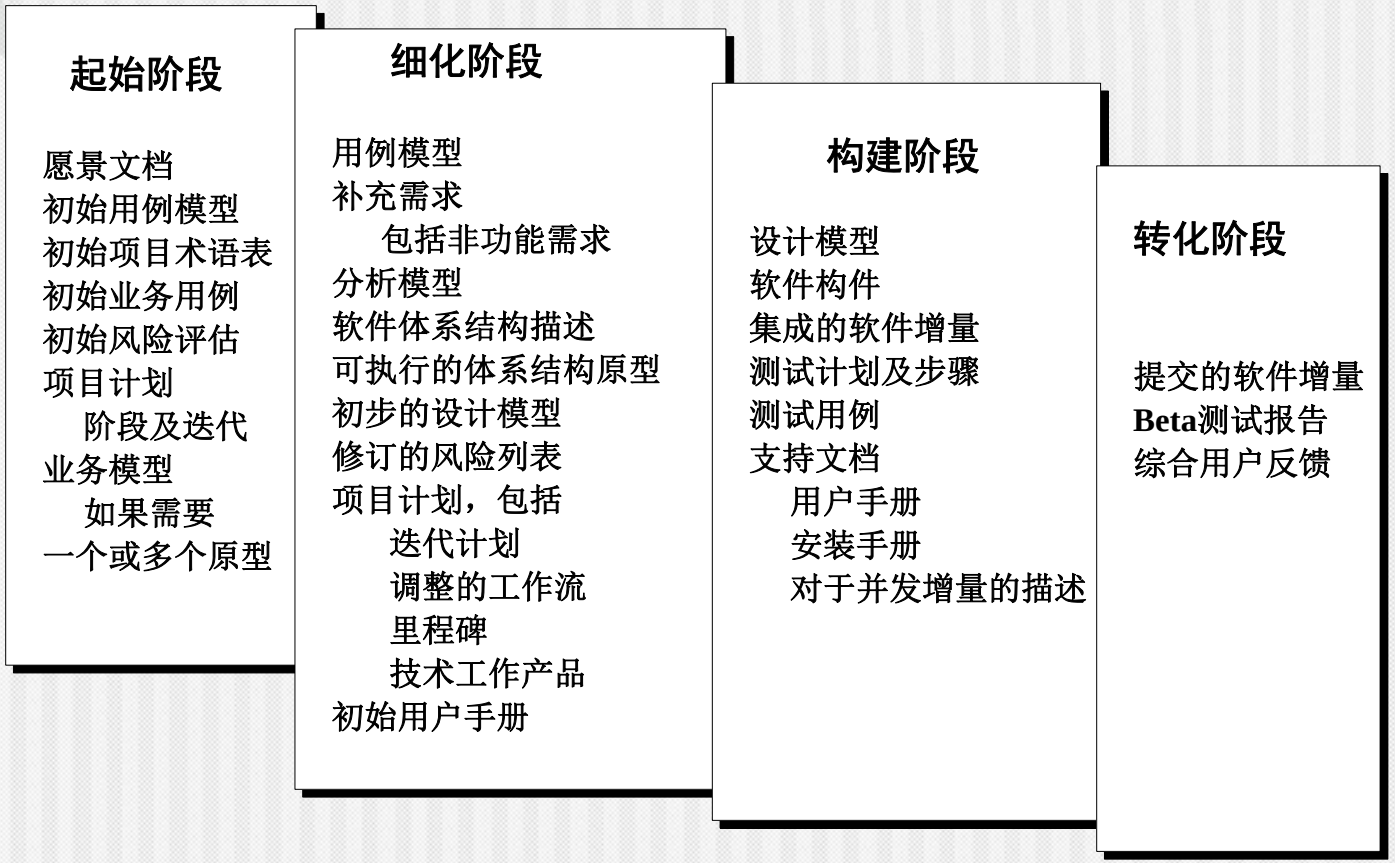
统一过程的阶段

- 阶段
  - 起始(inception)
  - 细化(elaboration)
  - 构建(construction)
  - 转换(transition)




UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

统一过程工作产品



UP每一个阶段产生的主要工作产品

4.4 产品与过程

谢谢！