第一部分 软件过程

- 软件过程结构
- 过程模型
- 敏捷开发
- 软件工程团队

从顶层了解软件工程实践

- 什么是软件过程?
- 软件过程中,有哪些通用的框架活动?
- 如何建立过程模型?什么是过程模式?
- 什么是惯用过程模型?有哪些优缺点?
- 为什么现代软件工程关注敏捷问题?
- 什么是敏捷软件开发?它与传统的过程模型有什么区别?

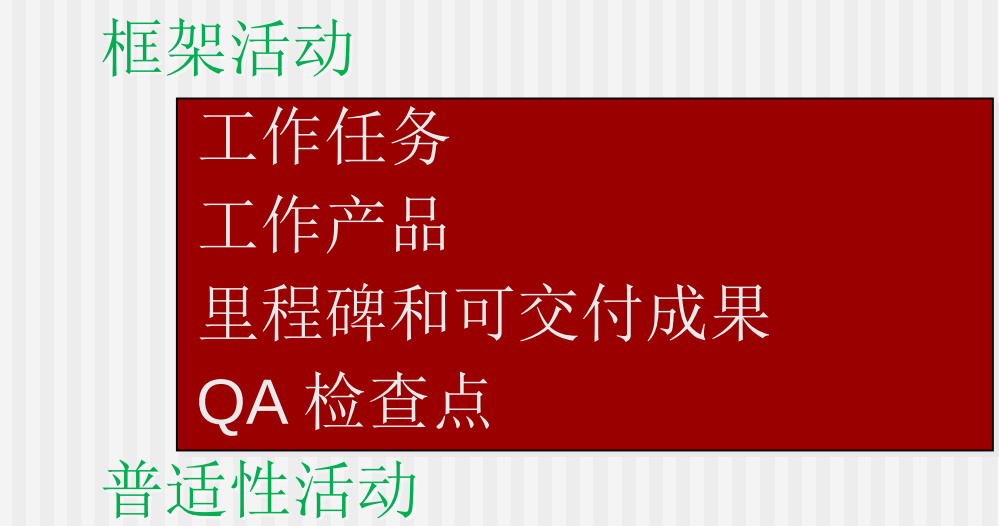
第3章　软件过程结构

软件过程

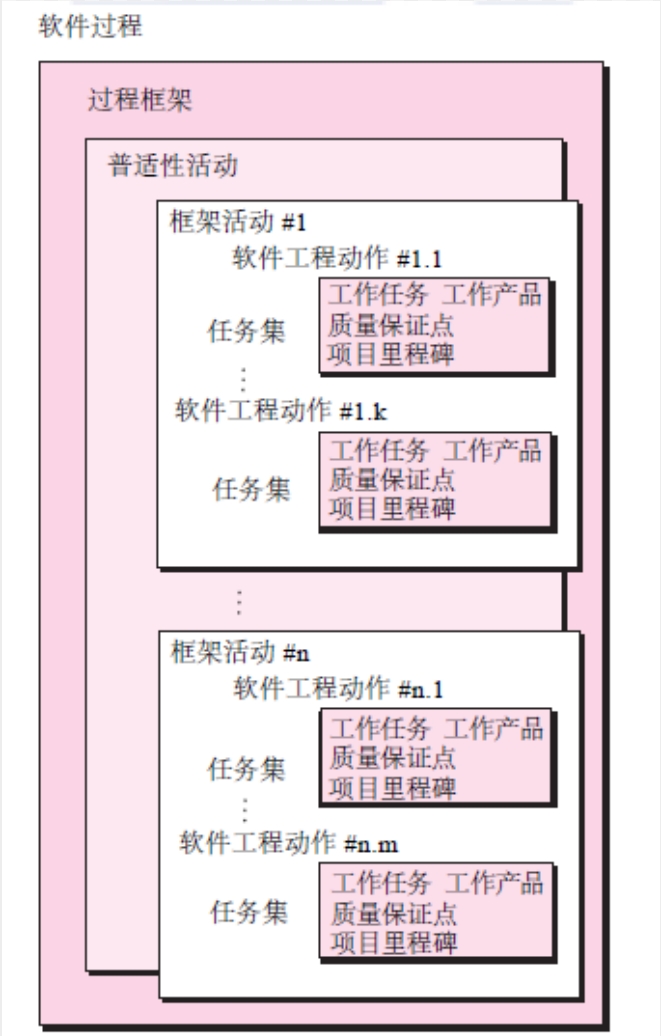
- 软件过程是构建软件产品时所执行的一系列活动、动作和任务的集合。
  - 活动（activity）主要实现宽泛的目标，如需求分析
  - 动作（action）包含了主要工作产品生产过程中的一系列任务，如动作体系结构设计要执行体系结构设计模型所决定的构建软件体系结构需要的一系列任务。
  - 任务（task）关注小而明确的目标，能够产生实际产品，如构建一个单元测试

过程框架

过程框架(process framework)定义了若干框架活动(framework activity)，为实现完整的软件工程过程建立基础，还包括一些全局影响的普适性活动，如：



通用过程模型

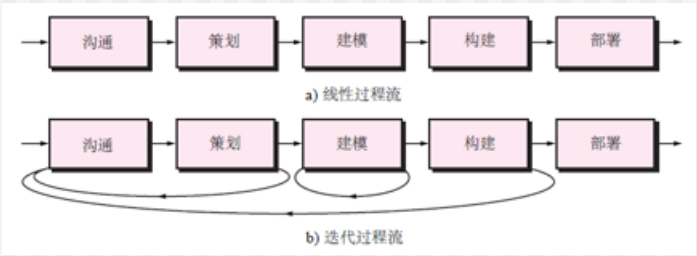


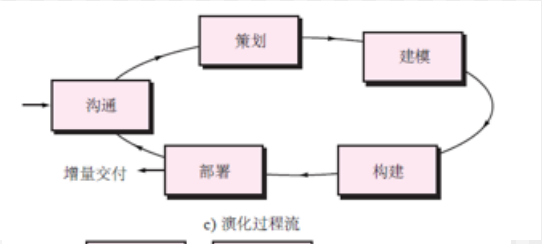
用来刻画软件过程中的对象及其关系

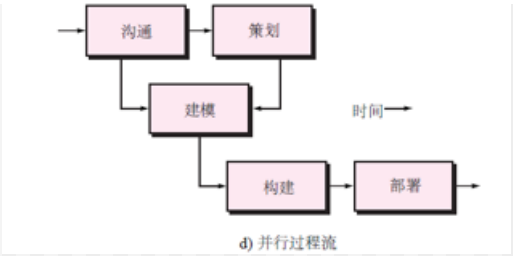
过程流

过程流(process flow)描述了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。它描述了软件过程的动态特性。下面是四种基本过程流。

- 线性过程流
- 迭代过程流
- 演化过程流
- 并行过程流







定义框架活动

- 对于具体的软件开发方法或者具体的软件开发过程，需要进一步定义框架活动的细节，描述包含哪些具体的动作，以及动作之间的关系。

例如一个小型的应用的沟通活动可以简单定义为电话交流动作，该动作可能包括下面任务集：

1.通过电话与软件的利益相关方取得联系；

2.讨论需求并记录；

3.将笔记整理成一份简单的书面需求；

4.通过Email请利益相关方审阅并认可。

明确任务集

- 每一个软件工程的动作都由若干个任务集构成
- 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。包括：
  - 所需完成任务的列表
  - 所需生产的工作产品列表
  - 应用质量保证技术的列表

示例（小型项目需求获取任务集）

对于一个小型、相对简单的项目而言,需求获取的任务集可能包括：

1.制定项目的利益相关者列表。

2.邀请所有利盖相关者参加一个非正式会议。

3.征询每个人对于软件特性和功能的需求。

4.讨论需求，并确定最终的需求列表。

5.划定需求优先级。

6.标出不确定域。

示例（大型项目需求获取任务集）

对于大型、复杂的软件工程项目，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.和利益相关者的每个成员分别单独讨论，获取所有的要求。

3.基于利益相关者的输入，建立初步的功能和特性列表。

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.建立一个修正的利益相关者需求列表。

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实施增量交付。

11.标注系统的约束和限制。

12.讨论系统验证方法。

过程模式（Process Patterns）

- 过程模式是复用已有的经过验证的软件过程的某些解决方案。
- 一个过程模式
  - 描述了软件工程工作中遇到的过程相关的问题
  - 明确了问题环境（上下文）
  - 给出了针对该问题的一种或几种可证明的解决方案
- 通俗地讲，过程模式提供了一个模版 [Amb98]
  一种在软件过程的背景下，统一描述问题解决方案的方法。
- 模式是可以组合的。

过程模式的描述模板

Ambler提出了下面的过程模式的描述模板：

- 模式名称：模式名称应能清楚地表述该模式在软件过程中的含义（例如技术评审）。
- 驱动力：模式的使用环境及主要问题，这些问题会显现在软件过程中并可能影响解决方案。
- 类型：定义模式类型。
- 启动条件：描述模式应用的前提条件。
- 问题：描述模式要解决的具体问题。
- 解决方案：描述如何成功实现模式。
- 结果：描述模式成功执行之后的结果。模式完成时需要明确：(1)完成的活动(2)结束状态(3)产生的信息
- 相关模式：列举与该模式相关的其他过程模式。描述模式之间的引用和组合关系。
- 已知应用和实例：说明该模式可应用的具体实例。

过程模式类型

- 步骤模式(stage pattern)：定义了与过程的框架活动相关的问题。例如建立沟通步骤模式
- 任务模式(task pattern)：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。例如需求收集任务模式
- 阶段模式(phase pattern)：定义框架活动在过程中出现的序列，即使这些活动流本质上是迭代的。例如螺旋模型和原型开发都可以是阶段模式。

过程模式实例（需求不清）

- 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团队之间的沟通方式；(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。利益相关者不确定他们想要什么，即他们无法详细描述软件需求。

过程模式实例（需求不清）

- 解决方案。描述了原型开发过程，具体见后续章节。
- 结果。开发了软件原型，识别了基本的需求（例如交互模式、计算特性、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；(2)原型系统被抛弃，采用其他过程模式建立软件产品。
- 相关模式。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。

过程评估与改进

- **用于过程改进的 CMMI标准评估方法**— 提供了五步的过程评估模型：启动、诊断、建立、执行和学习。
- **用于组织内部过程改进的CMM评估**—采用SEI的CMM作为评估的依据[Dun01]，提供了一种诊断方法，用以分析软件开发机构相对成熟度。
- **SPICE—The SPICE (ISO/IEC15504)** 标准定义了软件过程评估的一系列要求。该标准的目的是帮助软件开发组织建立客观的评价体系，以评估定义的软件过程的有效性。[ISO08]
- **软件ISO 9001:2000**—这是一个通用标准，任何开发组织如果希望提高所提供的产品、系统或服务的整体质量，都可以采用这个标准。因此，该标准可直接应用于软件组织和公司。[Ant06]

第4章　过程模型

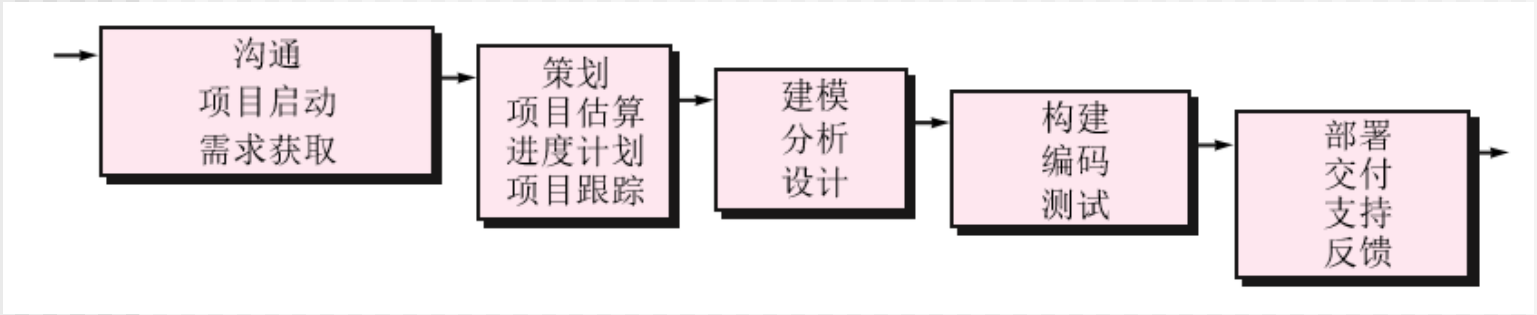
- 过程模型为软件工程工作提出了大量有用的结构，并为软件团队提供了有效的路线图，规定了所有活动的流程、动作、任务、迭代的程、工作产品及要完成的工作应如何组织。
- 重要性：软件过程提高了软件工程活动的稳定性、可控性和有组织性。
- 过程模型试图在找出混乱世界中的秩序和适应不断发生的变化之间寻求平衡

1 惯用过程模型

- 惯用过程模型提倡有序的软件工程方法，这将导致一些问题...
- 如果惯用过程模型力求实现结构化和有序，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（以及它们带来的秩序），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？

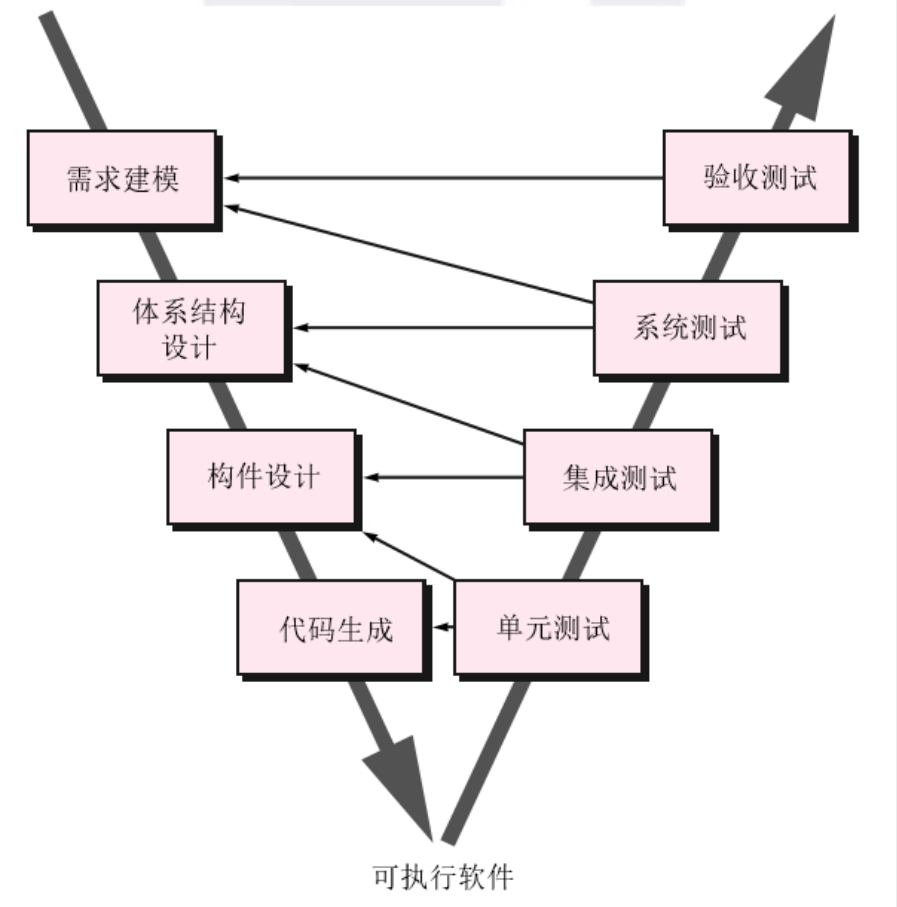
瀑布模型

- 瀑布模型（waterfall model）又称经典生命周期，是，是一个系统的、顺序的开发方法。...



V模型（瀑布模型的变体）

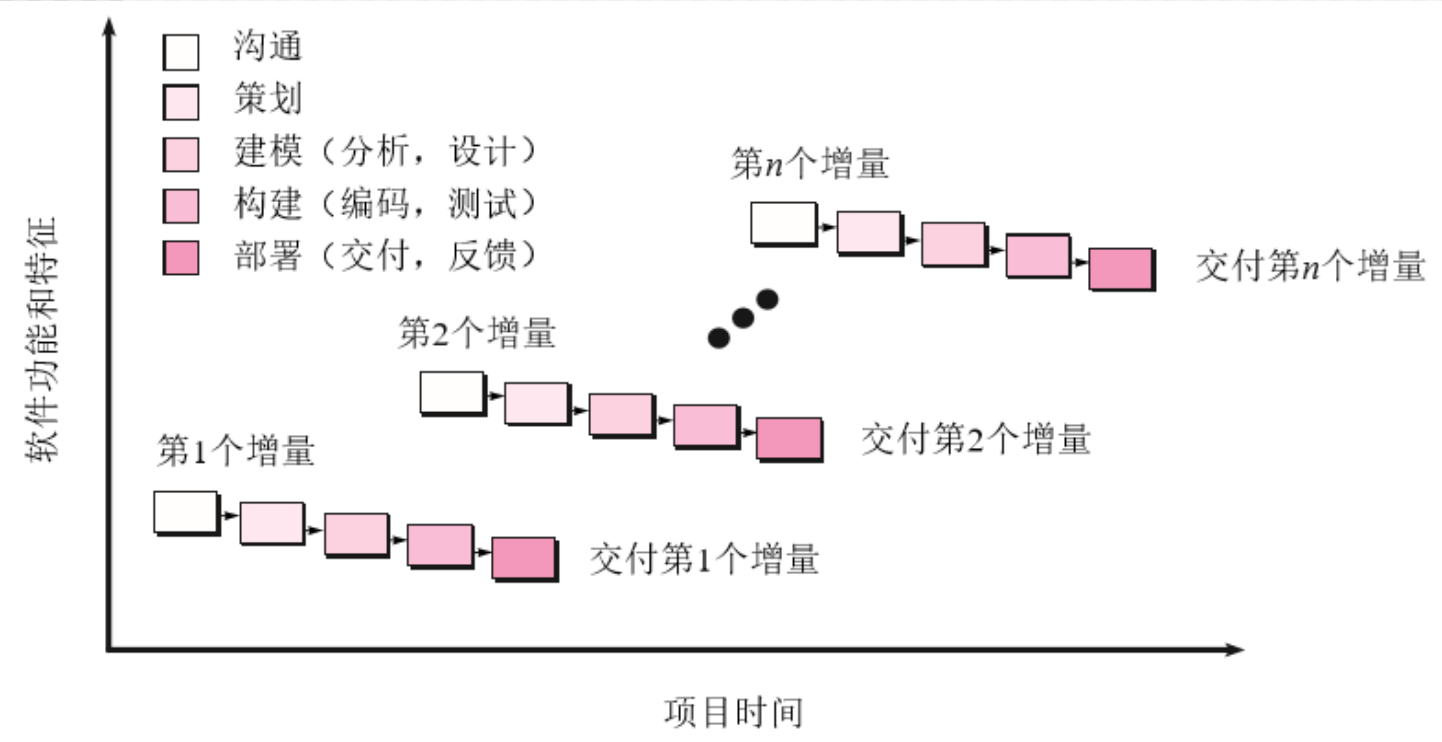
- V模型和瀑布模型没有本质区别，只是更多地强调质量保证工作，通过一系列测试去验证前期分析、设计阶段所形成的模型。…



瀑布模型的不足

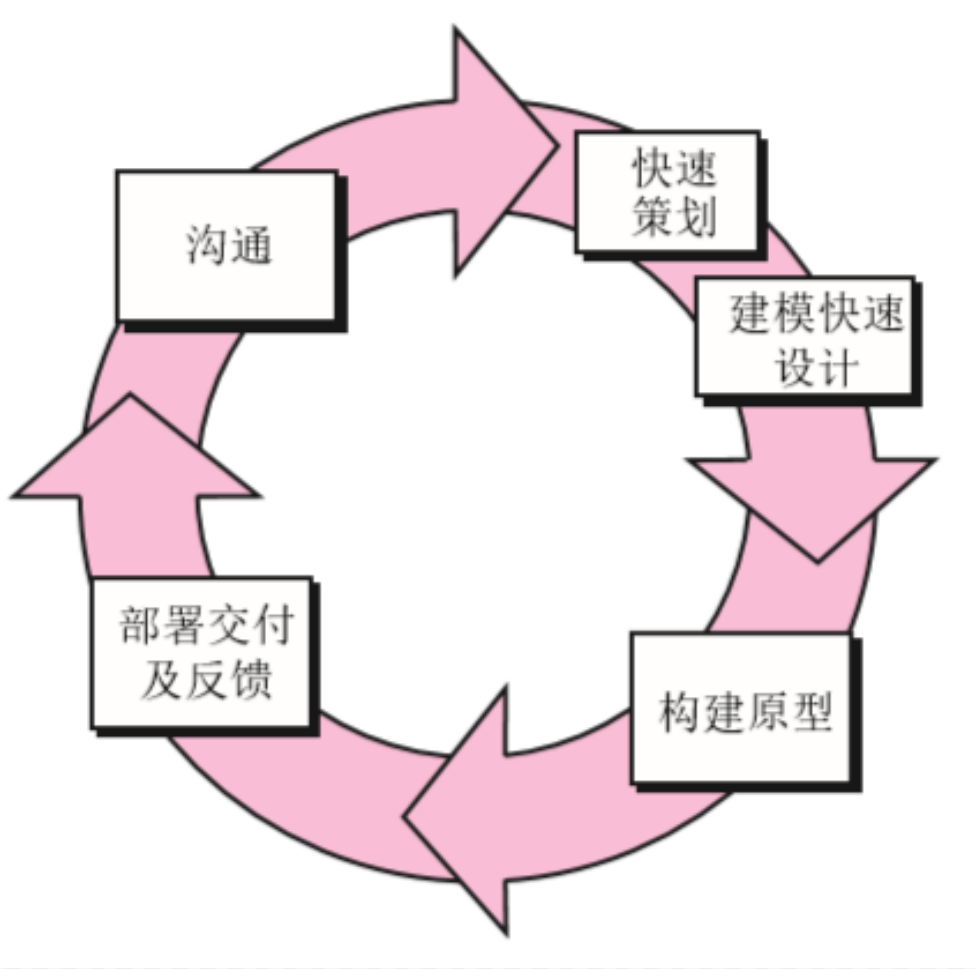
- 实际的项目很少遵守瀑布模型提出的顺序。…
- 客户通常难以清楚地描述所有的需求。而瀑布模型却要求客户一开始就要明确需求。
- 可执行的系统要在最后阶段才能提供给用户，未发现的重大缺陷或大的需求遗漏，将导致重大损失。
- 此外，严格的顺序性将导致团队成员间的等待。（阻塞状态）

增量过程模型

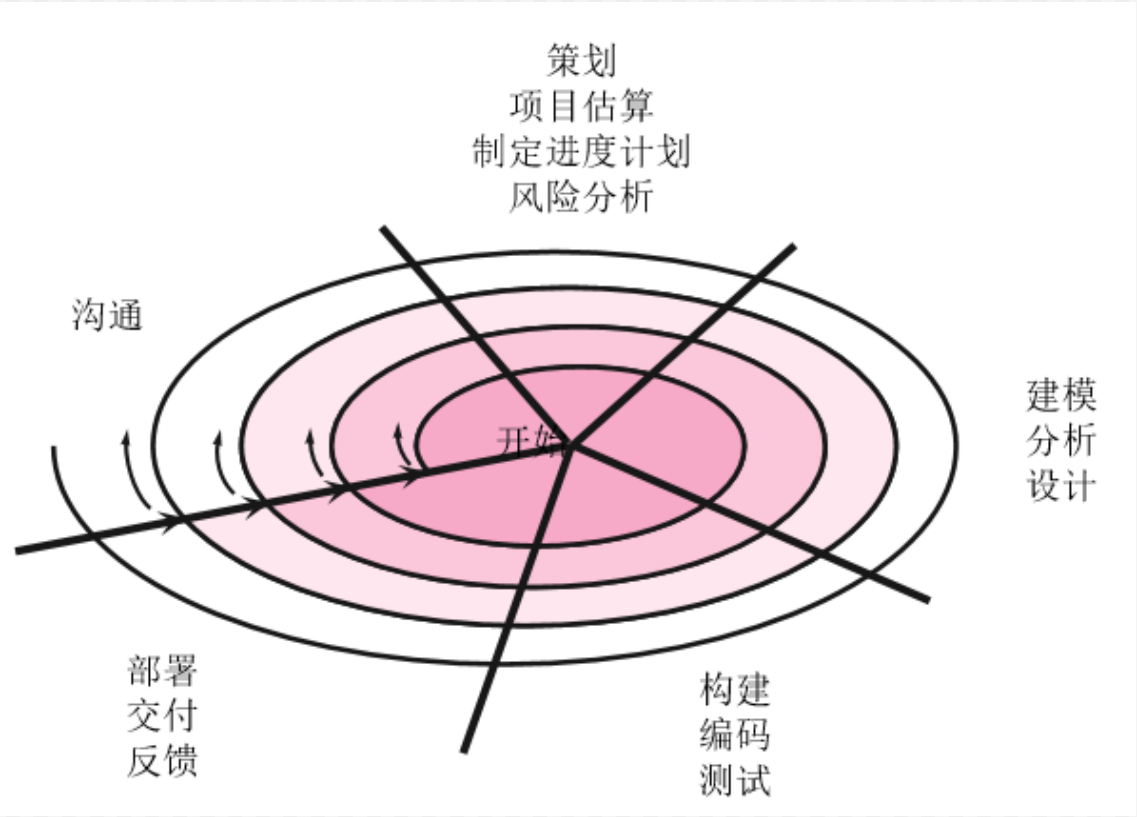


- 综合了线性过程流和并行过程流。
- 第一个增量通常是核心产品。

演化过程模型：原型开发



演化过程模型：螺旋模型

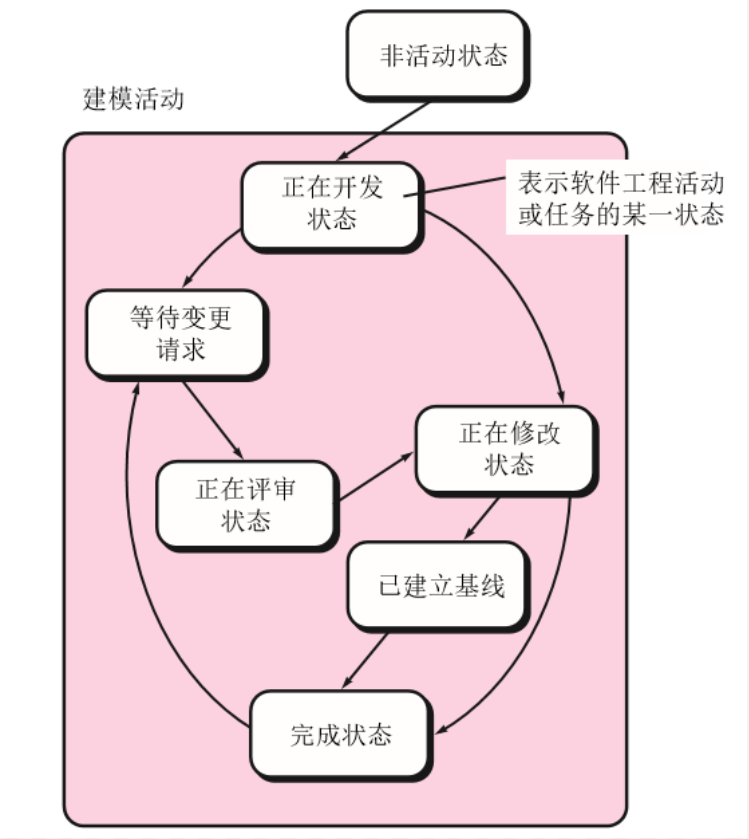


结合了原型（迭代性）和瀑布模型（可控性、系统性）

- 与原型模型一样，采用循环的方式逐步加深系统定义和实现的深度。
- 确定一系列里程碑作为支撑点

并发模型

- 并发开发模型是一种更为通用的过程模型，它可以用来表达前面的所有过程模型。
- 右图表达了建模活动可能处于的状态，任何一种状态都可能。
- 所有的软件工程活动都同时存在，并处于不同的状态。
- 状态转换通常是由事件触发的。



演化模型小结

- 一定程度上克服了瀑布模型的缺点，比较容易适应需求不清、需求变更等场景。…
- 可以尽早导出可运行的产品，容易得到用户的认可，对开发人员也是一种鼓舞。
- 演化模型造成项目开发混乱的可能性大于瀑布模型。
- 演化模型通常强调的是灵活性、可延展性和开发速度，而不是高质量。

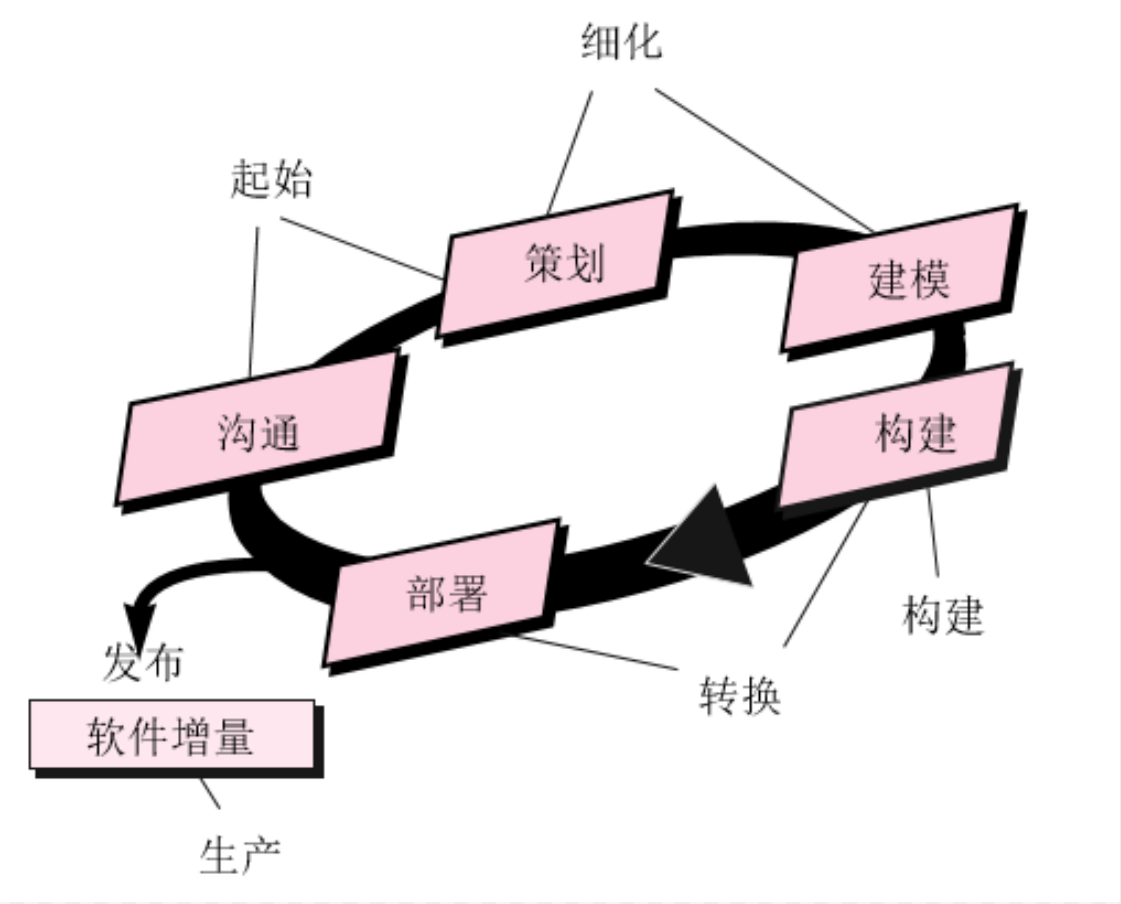
2 专用过程模型

- 基于构件的开发模型——能够促进软件复用的过程模型
- 形式化方法模型——强调软件形式化的数学规格说明（描述、开发和验证）
- 面向方面的软件开发（AOSD）—为定义、说明、设计和构建方面（基于关注点）提供过程和方法

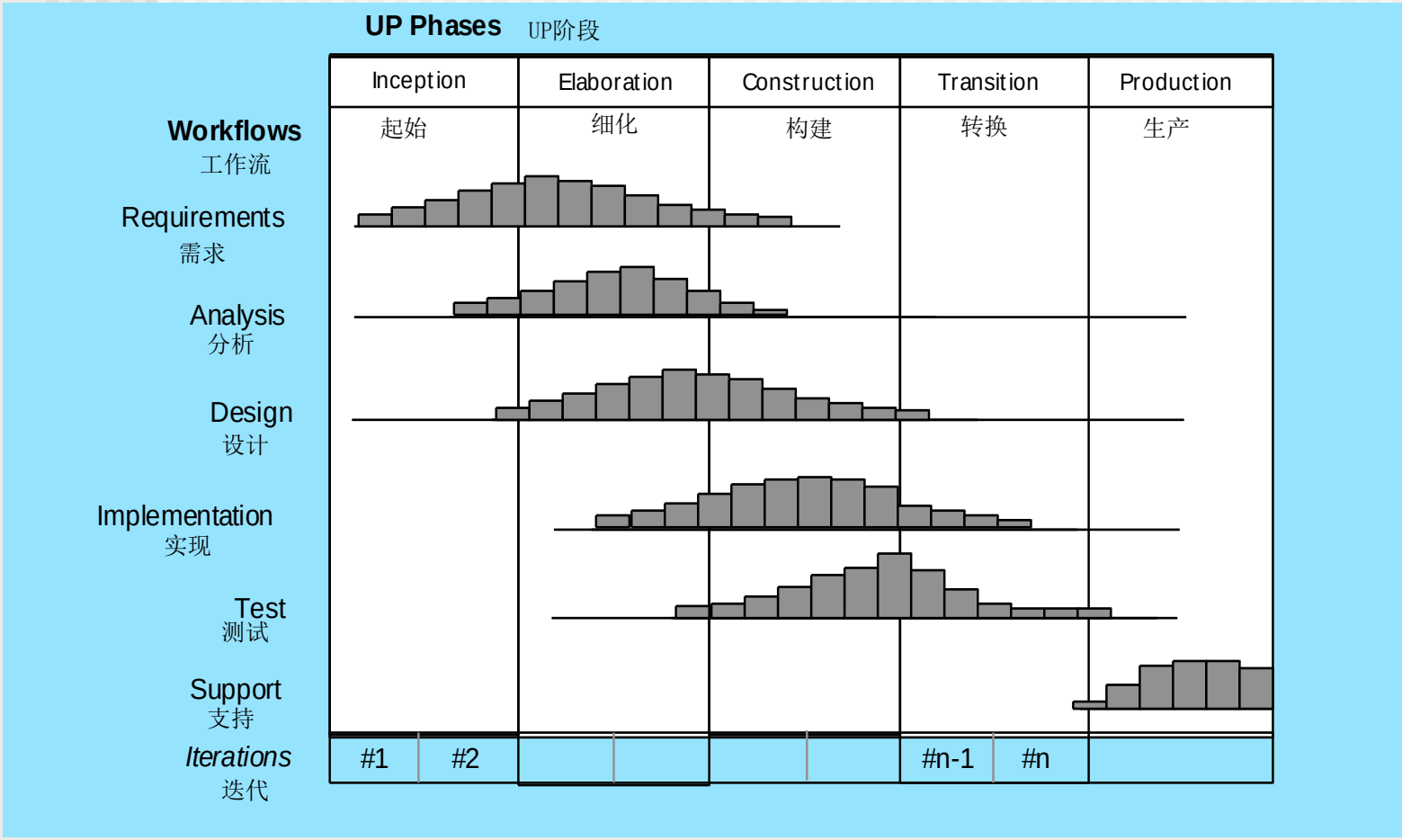
3 统一过程UP(Unified Process)

- 统一过程：一种“用例驱动，以架构为核心，迭代并且增量”的软件过程，与统一建模语言UML紧密结合。
- 20世界90年代早期，对象建模技术OMT（91）、应用的面向对象分析和设计（94）、面向对象的软件工程（92）的整合和统一。

统一过程(UP)



统一过程的阶段


UP Phases
UP阶段
Inception
Elaboration
Construction
Transition
Production
起始
细化
构建
转换
生产
Workflows
工作流
Requirements
需求
Analysis
分析
Design
设计
Implementation
实现
Test
测试
Support
支持
Iterations
迭代
#1
#2
#n-1
#n

UP工作产品

起始阶段

版本文档
初始用例模型
初始项目表
初始商业用例
初始风险评估
项目计划、阶段和迭代
商业模型（可选）
一个或多个原型

细化阶段

用例模型
补充需求（包含非功能性的分析模型、软件架构描述）
可执行的框架原型
初步设计模型
修正的风险清单
项目计划（包括迭代计划、适应性工作流、里程碑和技术性工作产物）
初步使用手册

构建阶段

设计模型
软件构件
集成软件增量
测试计划和步骤
测试用例
支持文档
用户手册
安装手册
当前版本描述

转换阶段

交付软件增量
Beta测试报告
用户反馈报告

4 个人过程模型和团队过程模型

- 个人过程模型和团队过程模型
- 强调成功软件过程的关键因素：测量、策划和自我管理

个人软件过程（PSP）

- **策划。**这个活动将需求活动分离出来，估算项目的规模和所需资源，并估算缺陷（工作中预测的缺陷数目）。所有的度量都用工作表或是模板记录。最后，识别开发任务，并建立项目进度计划。
- **高层设计。**搭建每个构件的的外部规格说明，并完成构件设计。如果有不确定的需求，则建立原型系统。所有问题都要记录和跟踪。
- **高层设计评审。**使用正式的验证方法（参见第21章）来发现设计中的错误。对所有的重要任务和工作结果都进行度量。
- **开发。**细化和评审构件级设计。完成编程，对代码进行评审，并进行编译和测试。对有的重要任务和工作结果都进行度量。
- **后验。**根据收集到的测量和度量结果（需要进行大量数据的统计分析），确定过程的有效性。度量和测量结果为提高过程的有效性提供指导。

团队软件过程（TSP）

- 建立自我管理团队来计划和跟踪他们的工作、确定目标、建立团队自己的过程和计划。团队既可以是纯粹的软件开发队伍，也可以是集成的产品队伍（IPT），可以由3~20名工程师组成。
- 指示管理人员如何指导和激励其团队，及如何帮助他们保持团队的最佳表现。
- 使CMM第5级的行为常规化并如预期一样，这样可加速软件过程改进。
  - 能力成熟模型(CMM)，一种衡量软件过程效率的技术，将在第30章中讨论。
- 为高成熟度的软件组织提供改进知道。
- 协助大学传授工业级团队技能。

5 过程技术

- 过程技术工具、过程建模工具
- 过程即软件（process is software）