软件工程导论（第6版）

第1章　软件工程学概述

软件工程课程目标及考核要求

- 《软件工程》是软件工程专业教学计划中的一门必修课程，也是本专业的主干课程之一。

- 课程主要阐述软件生命周期中有关软件分析、设计、实现、测试与维护中涉及的模型、方法、工具及其应用等，以培养学生在软件生命周期中的软件工程原理、建模能力、软件应用开发能力为目标。

软件工程课程目标及考核要求

- 1：学会运用工程化思维进行软件开发。掌握软件工程的基本概念、软件生存期各阶段的特点和内容，掌握主流软件生命周期模型（瀑布、原型、RUP 等），敏捷开发的模型。
- 2：掌握可行性研究和评价方法，掌握需求分析阶段的任务及方法，熟悉各种图形工具的应用。
- 3：掌握软件工程中实用的开发方法和技术，掌握软件总体设计的基本原理和方法，掌握详细设计方法，掌握软件测试基本方法。
- 4：了解软件工程领域的发展动向。掌握软件生命周期遵循的标准、流程和管理规范，能够用所学知识进行规范化软件开发和管理，设计并编写各类技术资料，提高在软件开发过程中的工程素养。

软件工程课程目标及考核要求

- 考核要求：
  - 期末总评成绩=平时成绩×40%+期末考试成绩×60%
- 其中：
  - 平时成绩包括出勤、作业、实验、期中考试

实验课安排

- 以“凤凰商城”的真实案例进行敏捷项目的持续的规划与设计、开发与集成、测试与反馈、部署与发布。

- 实验环境为真实的华为公有云平台 http://www.huaweicloud.com/，实验手册中所有工具的操作及使用均在该平台上进行。

- 需要注册华为云账号，5人为一个开发小组。

注册华为云账号

第1章 软件工程学概述

迄今为止，计算机系统已经经历了4个不同的发展阶段，但是，人们仍然没有彻底摆脱“软件危机”的困扰，软件已经成为限制计算机系统发展的瓶颈。

为了更有效地开发与维护软件，软件工作者在20世纪60年代后期开始认真研究消除软件危机的途径，从而逐渐形成了一门新兴的工程学科——计算机软件工程学。

重点与难点

重点：

掌握软件工程的各个阶段。（提示：结合瀑布模型）

难点：

软件生命周期模型的内容理解及选择。

主要内容

1.1 软件危机

1.2 软件工程

1.3 软件生命周期

1.4 软件过程

主要内容

1.1 软件危机

1.2 软件工程

1.3 软件生命周期

1.4 软件过程

1.1 软件危机

1.1.1 软件危机的介绍

在计算机软件的开发和维护过程中所遇到的一系列严重问题。

1.1 软件危机

软件危机的典型表现

1、对软件开发成本和进度的估计常常很不准确

2、用户对“已完成的”软件系统不满意的现象经常发生

3、软件产品的质量往往靠不住。

4、软件常常是不可维护的。

1.1 软件危机

软件危机的典型表现

4、软件通常没有适当的文档资料。

5、软件成本在计算机系统总成本中所占的比例逐年上升。

6、软件开发生产率提高的速度，远远跟不上计算机应用迅速普及深入的趋势。

1.1 软件危机

1.1.2 产生软件危机的原因

与软件本身特点有关

1. 软件不同于硬件，管理和控制软件开发过程相当困难。
2. 软件在运行过程中不会因为使用时间过长而被“用坏”。如果运行中发现了错误，很可能是遇到了一个在开发时期引入的在测试阶段没能检测出来的错误。
3. 软件不同于一般程序，它的一个显著特点是规模庞大，而且程序复杂性将随着程序规模的增加而呈指数上升。

1.1 软件危机

与软件本身特点有关

4 事实上，对用户要求没有完整准确的认识就匆忙着手编写程序是许多软件开发工程失败的主要原因之一。

5 目前相当多的软件专业人员对软件开发和维护还有不少糊涂观念。在实践过程中或多或少地采用了错误的方法和技术，这可能是使软件问题发展成软件危机的主要原因。

6 错误的认识和做法主要表现为忽视软件需求分析的重要性，认为软件开发就是写程序并设法使之运行,轻视软件维护等

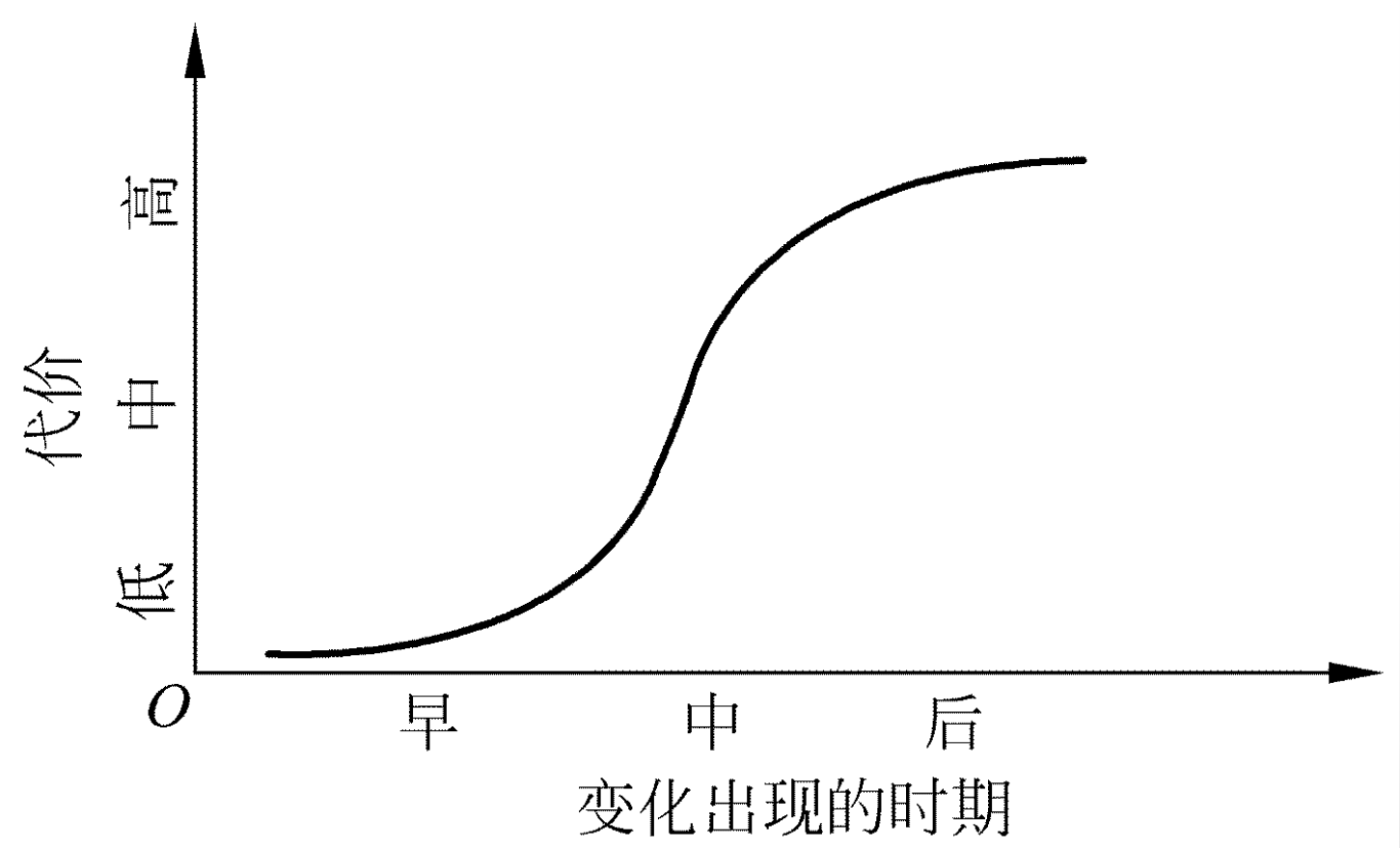
1.1 软件危机

软件开发与维护的方法不正确有关

1. 只重视程序而忽视软件配置其余成分的糊涂观念。
2. 软件开发人员在定义时期没有正确全面地理解用户需求，直到测试阶段或软件交付使用后才发现“已完成的”软件不完全符合用户的需要。
3. 严重的问题是在软件开发的不同阶段进行修改需要付出的代价是很不相同的，如下图所示。

1.1 软件危机

在软件开发的不同阶段进行修改需要付出的代价



1.1 软件危机

1.1.3 消除软件危机的途径

1. 首先应该对计算机软件有一个正确的认识。
2. 充分认识到软件开发不是某种个体劳动的神秘技巧，而应该是各类人员协同配合，共同完成的工程项目。
3. 推广使用在实践中总结出来的开发软件的成功的技术和方法，并且研究探索更好更有效的技术和方法。
4. 应该开发和使用更好的软件工具。

主要内容

1.1 软件危机

1.2 软件工程

1.3 软件生命周期

1.4 软件过程

1.2 软件工程

1.2.1　软件工程的介绍

软件工程概述

软件工程是指导计算机软件开发和维护的一门工程学科。采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，以经济地开发出高质量的软件并有效地维护它，这就是软件工程。

1.2 软件工程

1968年在第一届NATO会议上曾经给出了软件工程的一个早期定义：“软件工程就是为了经济地获得可靠的且能在实际机器上有效地运行的软件，而建立和使用完善的工程原理。”

1993年IEEE进一步给出了一个更全面更具体的定义：“软件工程是：①把系统的、规范的、可度量的途径应用于软件开发、运行和维护过程，也就是把工程应用于软件；②研究①中提到的途径。

1.2 软件工程

软件具有的本质特性

- 软件工程关注于大型程序的构造
- 软件工程的中心课题是控制复杂性
- 软件经常变化
- 开发软件的效率非常重要
- 和谐地合作是开发软件的关键
- 必须有效地支持它的用户
- 两种背景的人创造产品这个特性与前两个特性紧密相关

1.2 软件工程

1.2.2 软件工程的基本原理

1、用分阶段的生命周期计划严格管理

2、坚持进行阶段评审

3、实行严格的产品控制

4、采用现代程序设计技术

5、结果应能清楚地审查

6、开发小组的人员应该少而精

7、承认不断改进软件工程实践的必要性

1.2 软件工程

1.2.3 软件工程方法学

1、传统方法学

- 传统方法学也称为生命周期方法学或结构化范型。它采用结构化技术(结构化分析、结构化设计和结构化实现)来完成软件开发的各项任务，并使用适当的软件工具或软件工程环境来支持结构化技术的运用。

2、面向对象方法学

- 与传统方法相反，面向对象方法把数据和行为看成是同等重要的，它是一种以数据为主线，把数据和对数据的操作紧密地结合起来的方法。

1.2 软件工程

方法	工具	过程
• 完成软件开发的各项任务的技术方法，回答“怎样做”的问题	• 为运用方法而提供的自动的或半自动的软件工程支撑环境	• 为了获得高质量的软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤

1.2 软件工程

1. 传统方法学

概念：传统方法学也称为生命周期方法学或结构化范型。它采用结构化技术(结构化分析、结构化设计和结构化实现)来完成软件开发的各项任务，并使用适当的软件工具或软件工程环境来支持结构化技术的运用。

1.2 软件工程

传统方法学的特点：

传统方法学把软件生命周期的全过程依次划分为若干个阶段，然后顺序地完成每个阶段的任务。

每个阶段的开始和结束都有严格标准，对于任何两个相邻的阶段而言，前一阶段的结束标准就是后一阶段的开始标准。

1.2 软件工程

在每一个阶段结束之前都必须进行正式严格的技术审查和管理复审。

审查的一条主要标准就是每个阶段都应该交出“最新式的”(即和所开发的软件完全一致的)高质量的文档资料，从而保证在软件开发工程结束时有一个完整准确的软件配置交付使用。

1.2 软件工程

采用生命周期方法学可以大大提高软件开发的成功率，软件开发的生产率也能明显提高。

目前，传统方法学仍然是人们在开发软件时使用得十分广泛的软件工程方法学。

1.2 软件工程

面向对象方法学：

概念：与传统方法相反，面向对象方法把数据和行为看成是同等重要的，它是一种以数据为主线，把数据和对数据的操作紧密地结合起来的方法。

1.2 软件工程

四个要点

- 把对象(object)作为融合了数据及在数据上的操作行为的统一的软件构件。
- 把所有对象都划分成类(class)。
- 按照父类与子类的关系，把若干个相关类组成一个层次结构的系统。
- 对象彼此间仅能通过发送消息互相联系。

1.2 软件工程

面向对象方法学基本原则：

尽量模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界、解决问题的方法与过程，从而使描述问题的问题空间(也称为问题域)与实现解法的解空间(也称为求解域)在结构上尽可能一致。

1.2 软件工程

面向对象方法学：

优点：

降低了软件产品的复杂性，提高了软件的可理解性，简化了软件的开发和维护工作。

面向对象方法特有的继承性和多态性，进一步提高了面向对象软件的可重用性。

主要内容

1.1 软件危机

1.2 软件工程

1.3 软件生命周期

1.4 软件过程

1.3 软件生命周期

软件生命周期由软件定义、软件开发和运行维护(也称为软件维护)3个时期组成，每个时期又进一步划分成若干个阶段。

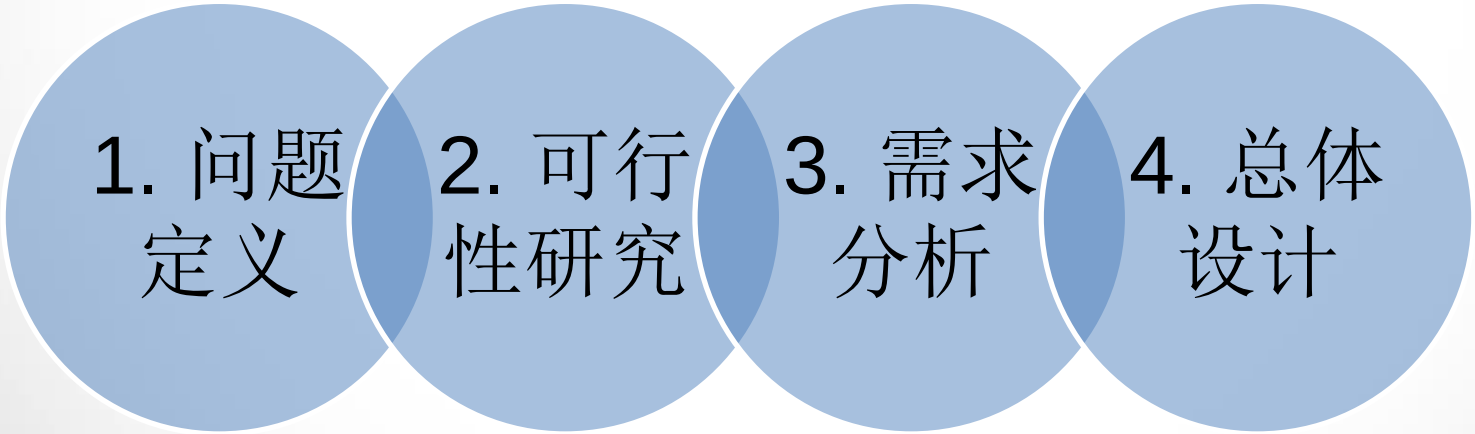
软件定义时期的任务是：确定软件开发工程必须完成的总目标；确定工程的可行性；导出实现工程目标应该采用的策略及系统必须完成的功能；估计完成该项工程需要的资源和成本，并且制定工程进度表。这个时期的工作通常又称为系统分析，由系统分析员负责完成。

1.3 软件生命周期

软件定义时期通常进一步划分成3个阶段，即问题定义、可行性研究和需求分析。开发时期具体设计和实现在前一个时期定义的软件，它通常由下述4个阶段组成：总体设计，详细设计，编码和单元测试，综合测试。其中前两个阶段又称为系统设计，后两个阶段又称为系统实现。维护时期的主要任务是使软件持久地满足用户的需要。

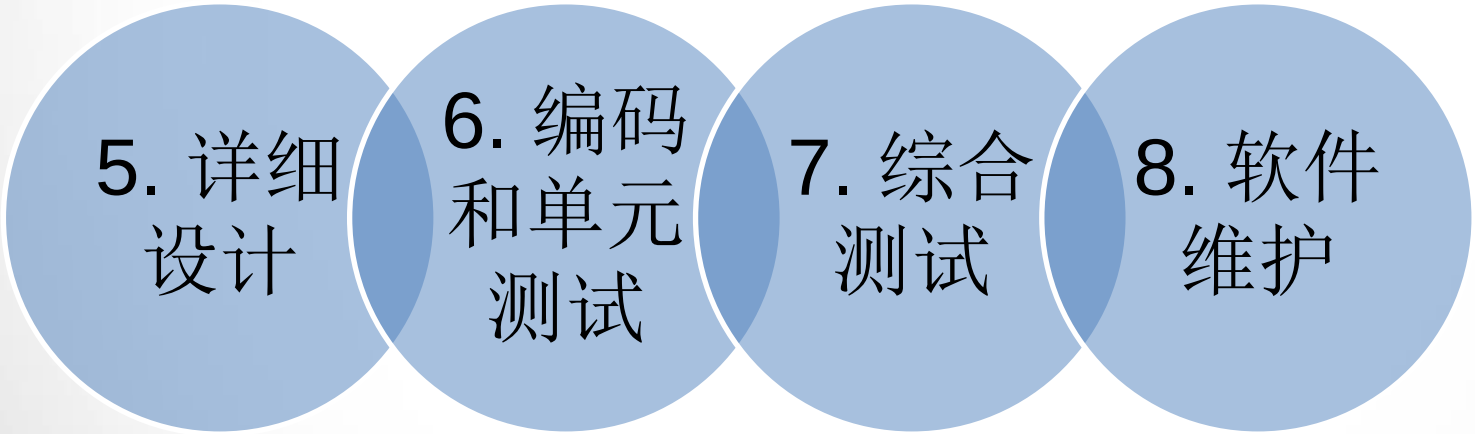
1.3 软件生命周期

下面简要介绍软件生命周期每个阶段的基本任务



1.3 软件生命周期

下面简要介绍软件生命周期每个阶段的基本任务



在实际从事软件开发工作时，软件规模、种类、开发环境及开发时使用的技术方法等因素，都影响阶段的划分。

主要内容

1.1 软件危机

1.2 软件工程

1.3 软件生命周期

1.4 软件过程

1.4 软件过程

软件过程是为了获得高质量软件所需要完成的一系列任务的框架，它规定了完成各项任务的工作步骤。

软件过程描述为了开发出客户需要的软件，什么人（who）、在什么时候（when）、做什么事（what）以及怎样（how）做这些事以实现某一个特定的具体目标。

1.4 软件过程

1.4.1 瀑布模型

瀑布模型一直是唯一被广泛采用的生命周期模型，现在它仍然是软件工程中应用得最广泛的过程模型。如下图所示为传统的瀑布模型

图1.2传统的瀑布模型如图1.2所示为传统的瀑布模型。

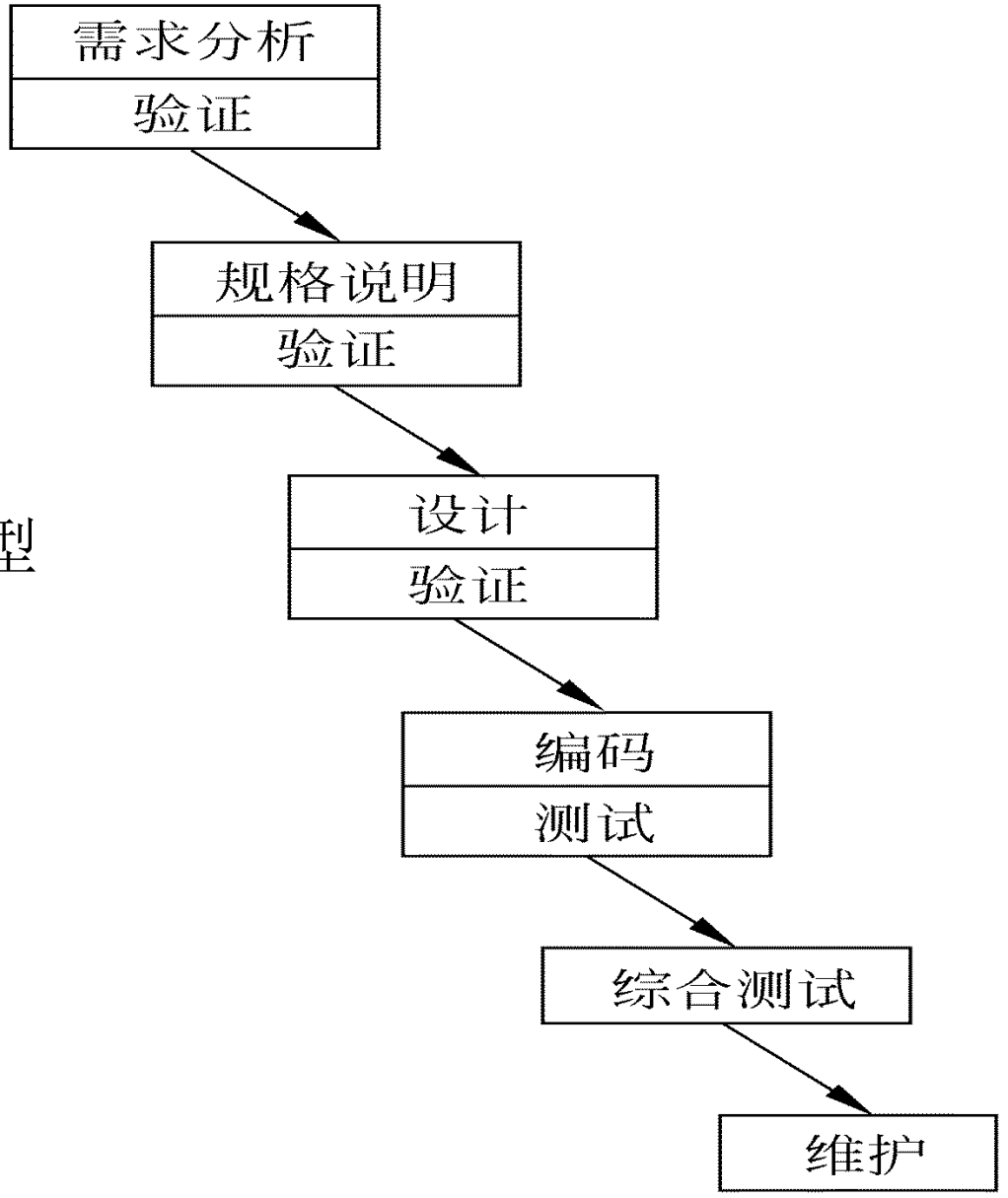


图1.2 传统的瀑布模型

1.4 软件过程

1.4.1 瀑布模型

按照传统的瀑布模型开发软件，有下述的几个特点。

a）阶段间具有顺序性和依赖性：

两重含义：①必须等前一阶段的工作完成之后，才能开始后一阶段的工作；②前一阶段的输出文档就是后一阶段的输入文档，因此，只有前一阶段的输出文档正确，后一阶段的工作才能获得正确的结果。

1.4 软件过程

1.4.1　瀑布模型

b）推迟实现的观点

瀑布模型在编码之前设置了系统分析与系统设计的各个阶段，分析与设计阶段的基本任务规定，在这两个阶段主要考虑目标系统的逻辑模型，不涉及软件的物理实现。

1.4 软件过程

1.4.1 瀑布模型

c）质量保证的观点：

软件工程的基本目标是优质、高产。为了保证所开发的软件的质量，在瀑布模型的每个阶段都应坚持两个重要做法。

① 每个阶段都必须完成规定的文档，没有交出合格的文档就是没有完成该阶段的任务。

② 每个阶段结束前都要对所完成的文档进行评审，以便尽早发现问题，改正错误。

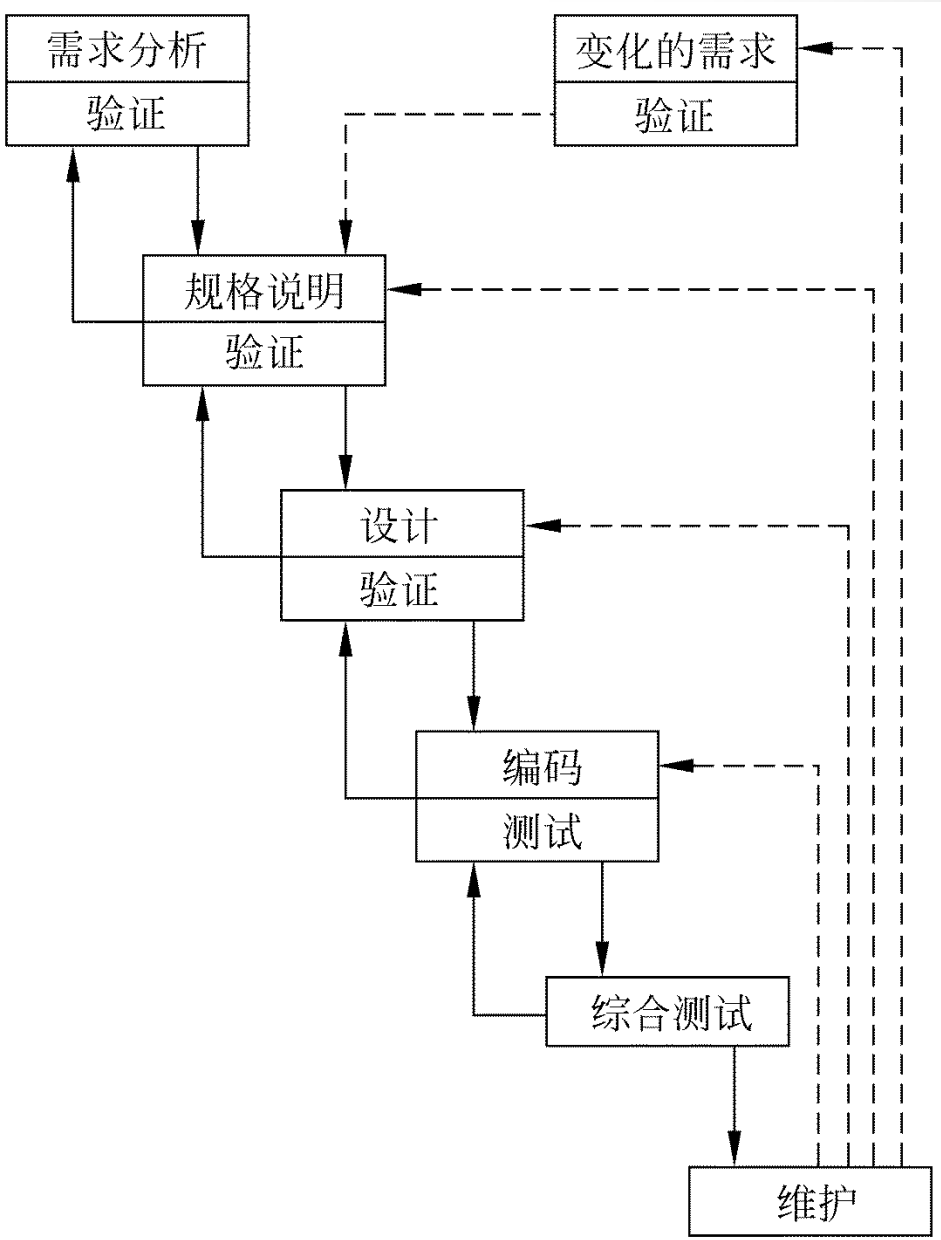
1.4 软件过程

1.4.1　瀑布模型

传统的瀑布模型过于理想化了，事实上，人在工作过程中不可能不犯错误。实际的瀑布模型是带“反馈环”的，如系统图1.3所示。

1、图中实线箭头表示开发过程，虚线箭头表示维护过程。、

2、实际的瀑布模型当在后面阶段发现前面阶段的错误时，需要沿图中左侧的反馈线返回前面的阶段，修正前面阶段的产品之后再回来继续完成后面阶段的任务。



1.4 软件过程

1.4.1 瀑布模型

瀑布模型有许多优点：

① 可强迫开发人员采用规范的方法（例如，结构化技术）；

② 严格地规定了每个阶段必须提交的文档；

③ 要求每个阶段交出的所有产品都必须经过质量保证小组的仔细验证。

1.4 软件过程

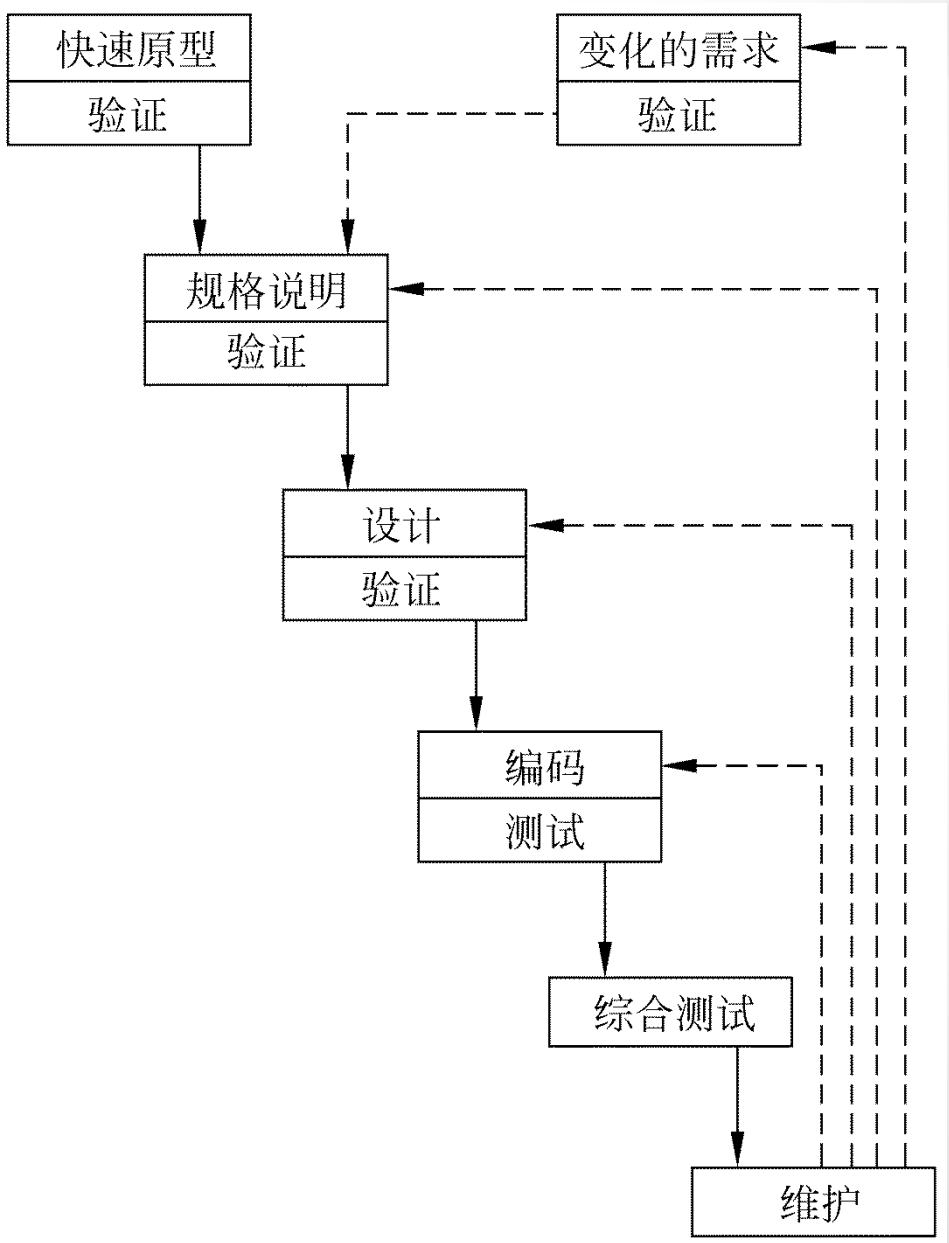
1.4.2. 快速原型模型

概念：

快速原型是快速建立起来的可以在计算机上运行的程序，它所能完成的功能往往是最终产品能完成的功能的一个子集。

如下图1.4所示：

图1.4 中实线箭头表示开发过程
虚线箭头表示维护过程



1.4 软件过程

1.4.2. 快速原型模型

快速原型模型是不带反馈环的，这正是这种过程模型的主要优点： 软件产品的开发基本上是线性顺序进行的。能基本上做到线性顺序开发的主要原因如下：

1.4 软件过程

1.4.2. 快速原型模型

（1）原型系统已经通过与用户交互而得到验证，据此产生的规格说明文档正确地描述了用户需求，因此，在开发过程的后续阶段不会因为发现了规格说明文档的错误而进行较大的返工。

（2）开发人员通过建立原型系统已经学到了许多东西，因此，在设计和编码阶段发生错误的可能性也比较小，这自然减少了在后续阶段需要改正前面阶段所犯错误的可能性。

1.4 软件过程

1.4.3. 增量模型

概念：

增量模型也称为渐增模型。使用增量模型开发软件时，把软件产品作为一系列的增量构件来设计、编码、集成和测试。每个构件由多个相互作用的模块构成，并且能够完成特定的功能。使用增量模型时，第一个增量构件往往实现软件的基本需求，提供最核心的功能。

增量模型如下图1.5所示：

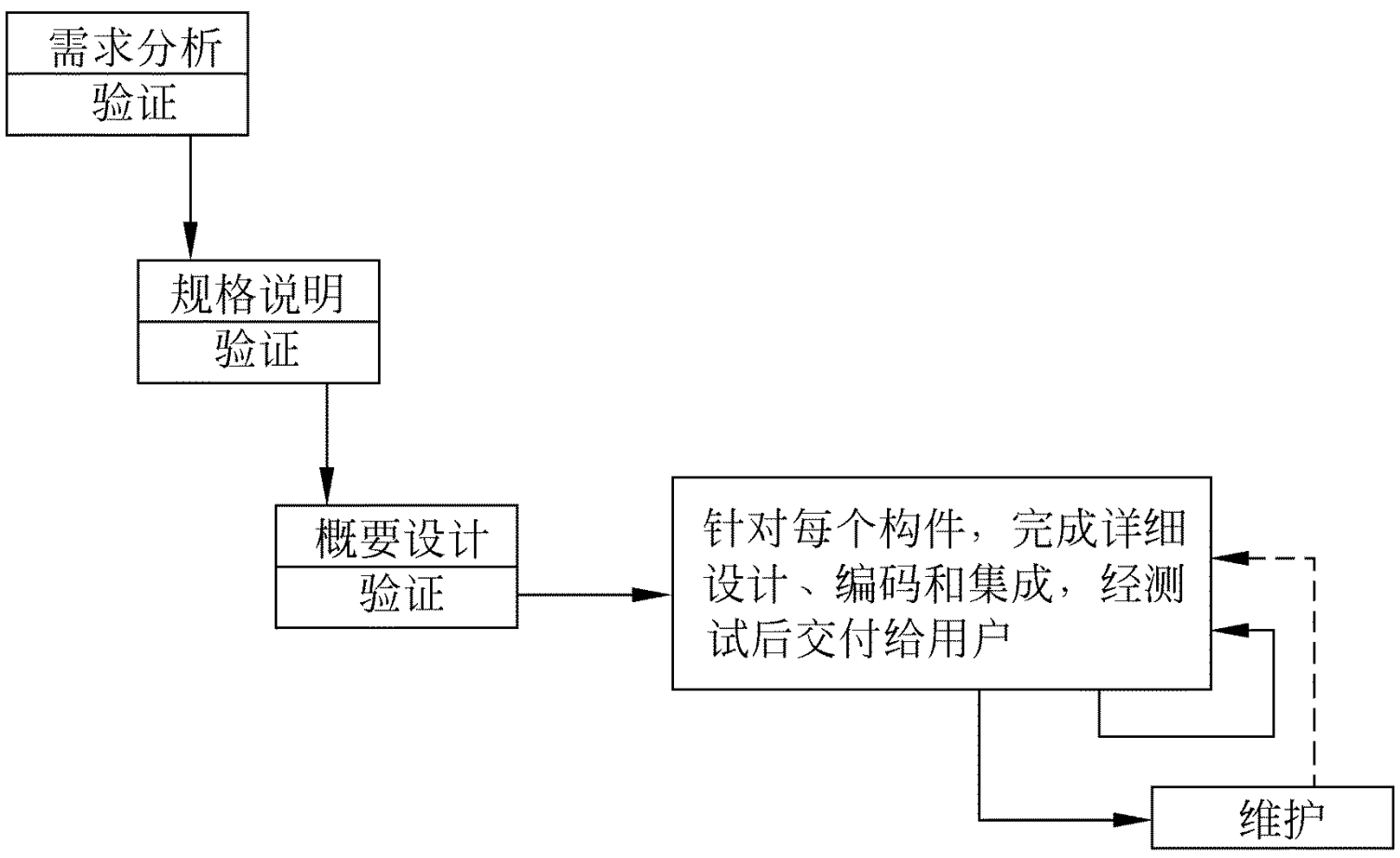


图1.5　增量模型

1.4 软件过程

1.4.3. 增量模型

优点：

能在较短时间内向用户提交可完成部分工作的产品。

逐步增加产品功能可以使用户有较充裕的时间学习和适应新产品，从而减少一个全新的软件可能给客户组织带来的冲击。

1.4 软件过程

1.4.3. 增量模型

使用增量模型的困难：

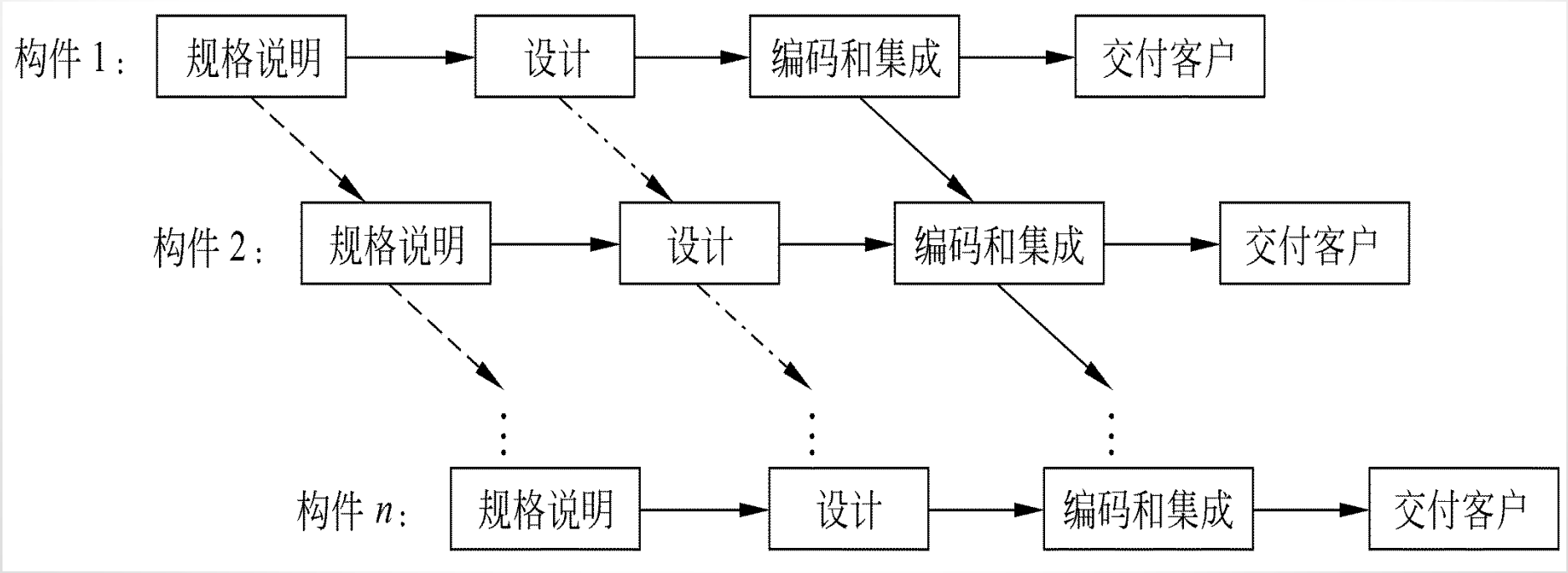
在把每个新的增量构件集成到现有软件体系结构中时，必须不破坏原来已经开发出的产品。

必须把软件的体系结构设计得便于按这种方式进行扩充，向现有产品中加入新构件的过程必须简单、方便，也就是说，软件体系结构必须是开放的。

1.4 软件过程

1.4.3. 增量模型

风险更大的增量模型：



1.4 软件过程

1.4.4 螺旋模型

概念：

螺旋模型的基本思想是，使用原型及其他方法来尽量降低风险。理解这种模型的一个简便方法，是把它看作在每个阶段之前都增加了风险分析过程的快速原型模型。

螺旋模型如下图所示：

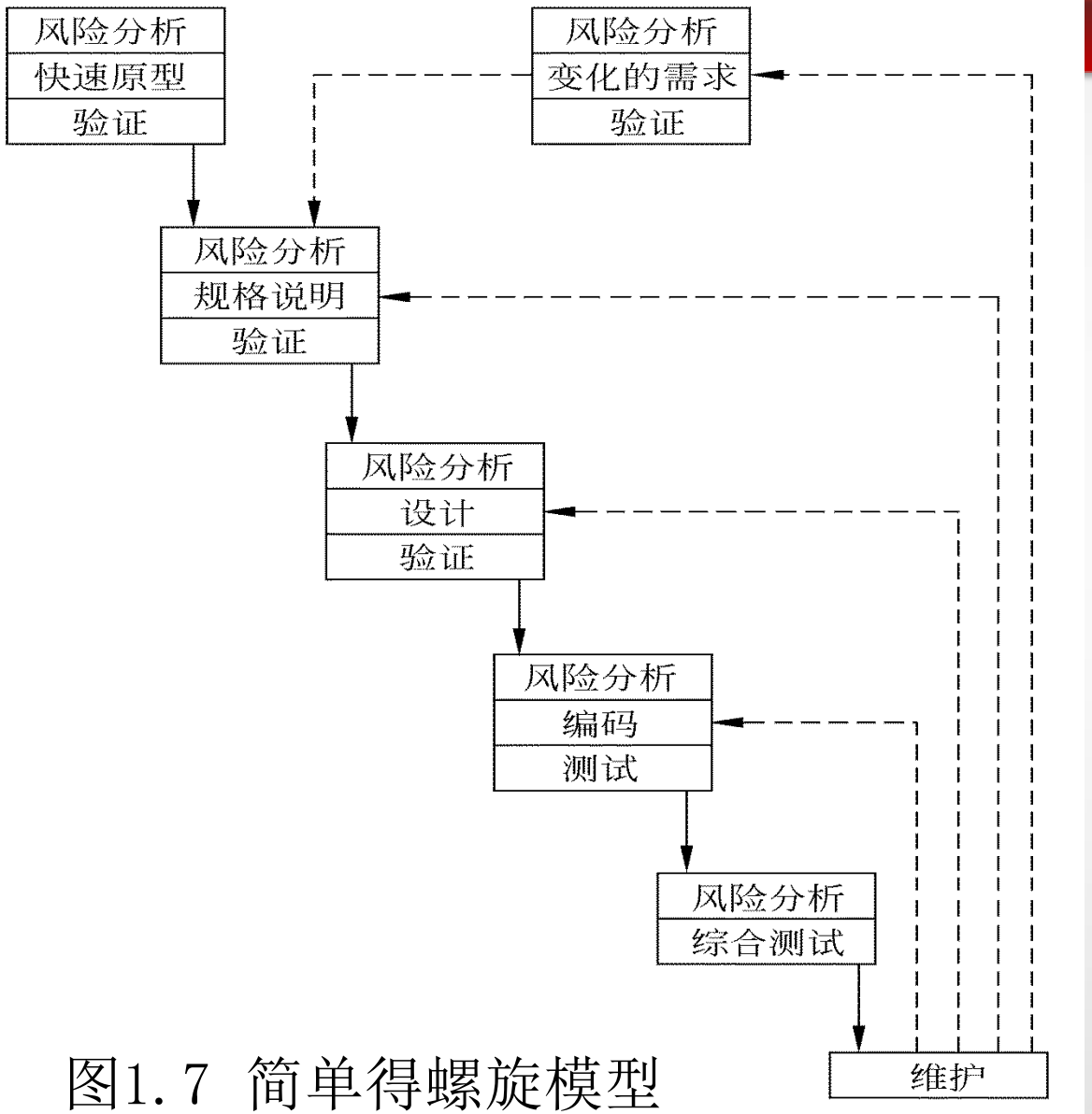

风险分析
快速原型
验证
风险分析
变化的需求
验证
风险分析
规格说明
验证
风险分析
设计
验证
风险分析
编码
测试
风险分析
综合测试
维护

图1.7 简单得螺旋模型

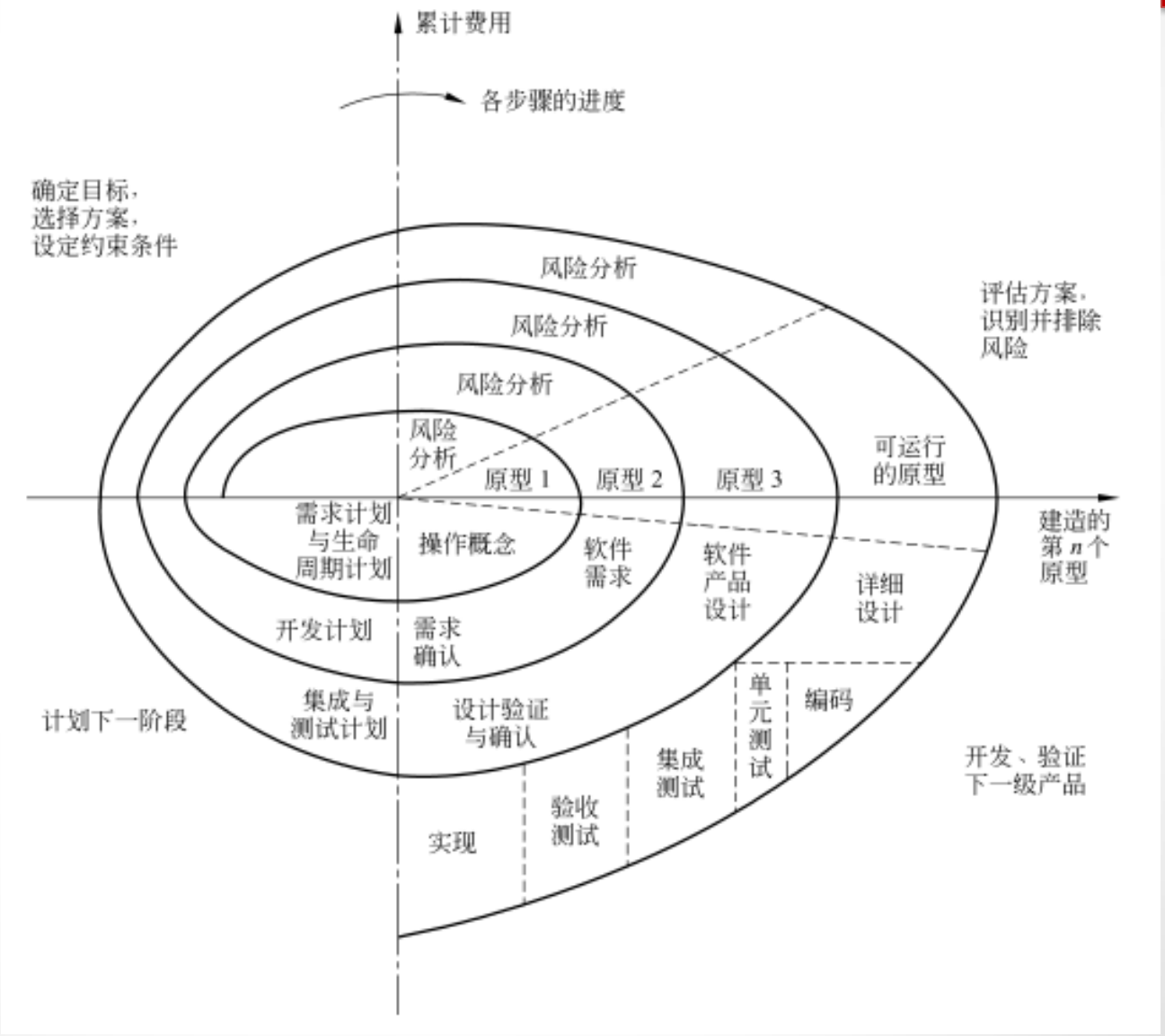

累计费用
各步骤的进度
确定目标，
选择方案，
设定约束条件
风险分析
风险分析
风险分析
风险
分析
评估方案，
识别并排除
风险
原型 1
原型 2
原型 3
可运行
的原型
需求计划
与生命
周期计划
操作概念
软件
需求
软件
产品
设计
详细
设计
建造的
第 n 个
原型
开发计划
需求
确认
集成与
测试计划
设计验证
与确认
单
元
测
试
编码
计划下一阶段
集成
测试
开发、验证
下一级产品
验收
测试
实现

图1.8完整的螺旋模型

1.4 软件过程

1.4.5. 喷泉模型

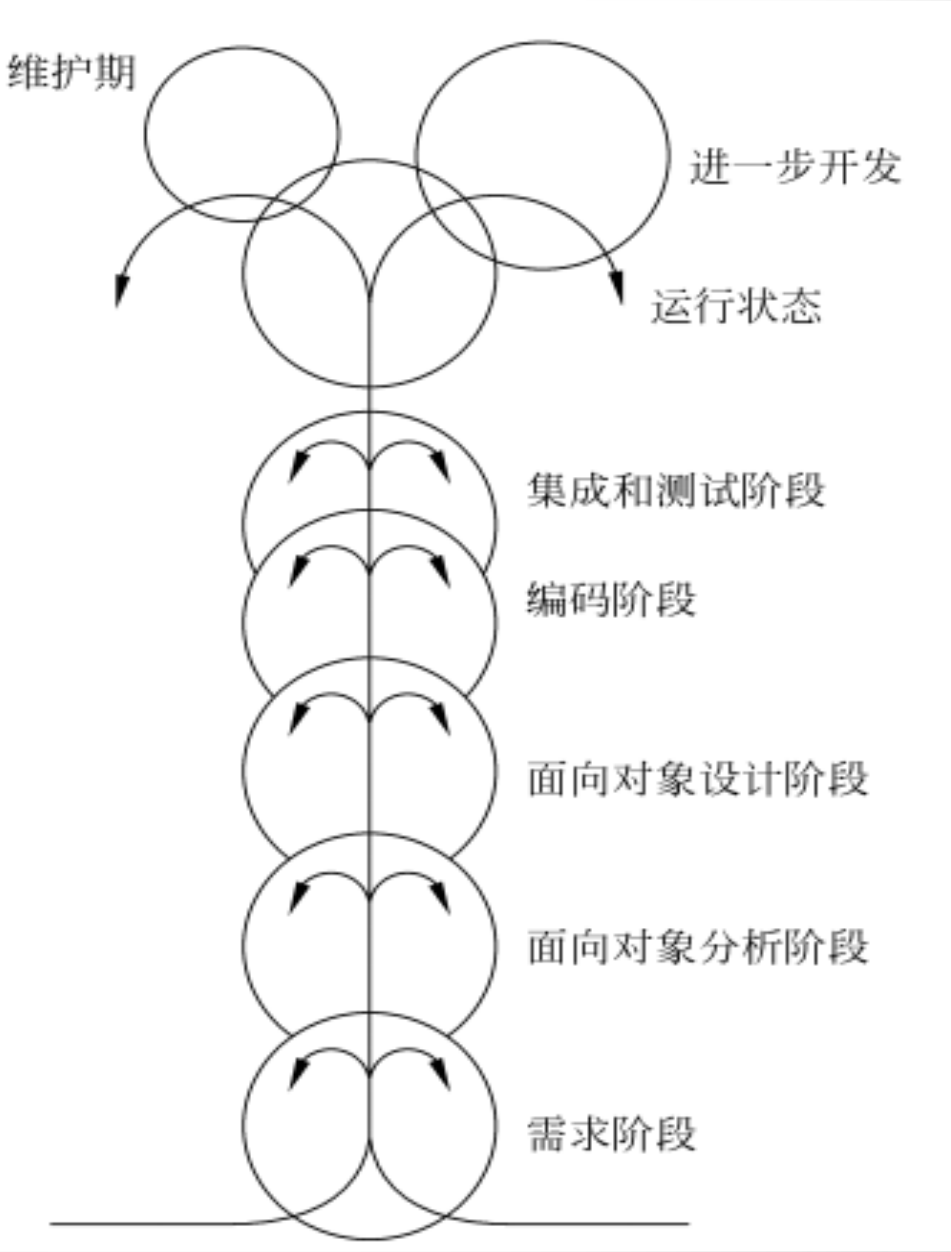
概念：

“喷泉”这个词体现了面向对象软件开发过程迭代和无缝的特性。迭代是软件开发过程中普遍存在的一种内在属性。用面向对象方法学开发软件时，工作重点应该放在生命周期中的分析阶段。

喷泉模型图如下图1.9所示：

图中代表不同阶段的圆圈相互重叠，这明确表示两个活动之间存在交迭；

图中在一个阶段内的向下箭头代表该阶段内的迭代（或求精）。图中较小的圆圈代表维护，圆圈较小象征着采用了面向对象范型之后维护时间缩短了。



1.4 软件过程

1.4.6. Rational统一过程

概念：

Rational统一过程（Rational Unified Process,**RUP**）是由Rational软件公司推出的一种完整而且完美的软件过程。

RUP总结了经过多年商业化验证的6条最有效的软件开发经验，这些经验被称为“最佳实践”。

1.4 软件过程

1.4.6. Rational统一过程

a）**最佳实践**

① 迭代式开发

迭代式开发允许在每次迭代过程中需求都可以有变化，这种开发方法通过一系列细化来加深对问题的理解，因此能更容易地容纳需求的变更。

② 管理需求

RUP描述了如何提取、组织系统的功能性需求和约束条件并把它们文档化。

1.4 软件过程

1.4.6. Rational统一过程

a) **最佳实践**

③ 使用基于构件的体系结构

UP提供了使用现有的或新开发的构件定义体系结构的系统化方法，从而有助于降低软件开发的复杂性，提高软件重用率。

④ 可视化建模

可视化建模语言UML紧密地联系在一起，在开发过程中建立起软件系统的可视化模型，可以帮助人们提高管理软件复杂性的能力。

1.4 软件过程

1.4.6. Rational统一过程

a) **最佳实践**

⑤ 验证软件质量

软件质量评估不再是事后型的或由单独小组进行的孤立活动，而是内建在贯穿于整个开发过程的、由全体成员参与的所有活动中。

⑥ 控制软件变更

RUP描述了如何控制、跟踪和监控修改，以确保迭代开发的成功。

1.4 软件过程

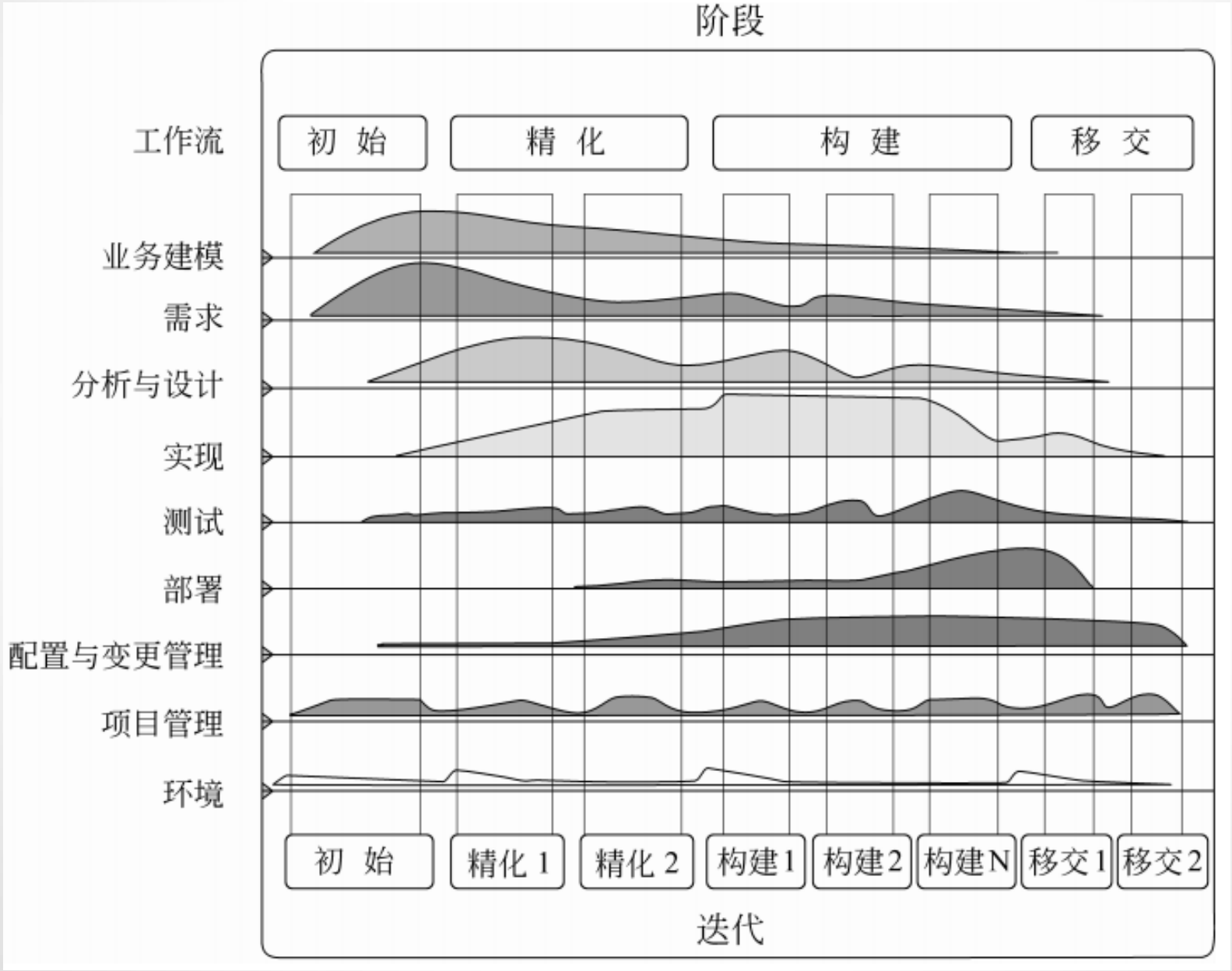
1.4.6. Rational统一过程

b) **RUP软件开发生命周期**

RUP软件开发生命周期是一个二维的生命周期模型，如下图1.10所示。图中纵轴代表核心工作流，横轴代表时间。

① **核心工作流**

RUP中有9个核心工作流，其中前6个为核心过程工作流程，后3个为核心支持工作流程。


阶段
工作流
初 始
精 化
构 建
移 交
业务建模
需求
分析与设计
实现
测试
部署
配置与变更管理
项目管理
环境
初 始
精化 1
精化 2
构建1
构建2
构建N
移交1
移交2
迭代

1.4 软件过程

1.4.6. Rational统一过程

b) RUP软件开发生命周期

工作阶段

RUP把软件生命周期划分成4个连续的阶段。每个阶段都有明确的目标，并且定义了用来评估是否达到这些目标的里程碑。每个阶段的目标通过一次或多次迭代来完成。

下面简述4个阶段的工作目标。

1.4 软件过程

1.4.6. Rational统一过程

初始阶段： 建立业务模型，定义最终产品视图，并且确定项目的范围。

精化阶段： 设计并确定系统的体系结构，制定项目计划，确定资源需求。

构建阶段： 开发出所有构件和应用程序，把它们集成为客户需要的产品，并且详尽地测试所有功能。

移交阶段： 把开发出的产品提交给用户使用。

1.4 软件过程

1.4.6. Rational统一过程

b) RUP软件开发生命周期

RUP迭代式开发

RUP重复一系列组成软件生命周期的循环。每次循环都经历一个完整的生命周期，每次循环结束都向用户交付产品的一个可运行的版本。

每个阶段又进一步细分为一次或多次迭代过程。

1.4 软件过程

1.4.7. 敏捷过程与极限编程

概念：

敏捷过程为了使软件开发团队具有高效工作和快速响应变化的能力，17位著名的软件专家于2001年2月联合起草了敏捷软件开发宣言。敏捷软件开发宣言由下述4个简单的价值观声明组成。

1.4 软件过程

1.4.7.敏捷过程与极限编程

- 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

1.4 软件过程

1.4.7.敏捷过程与极限编程

极限编程：

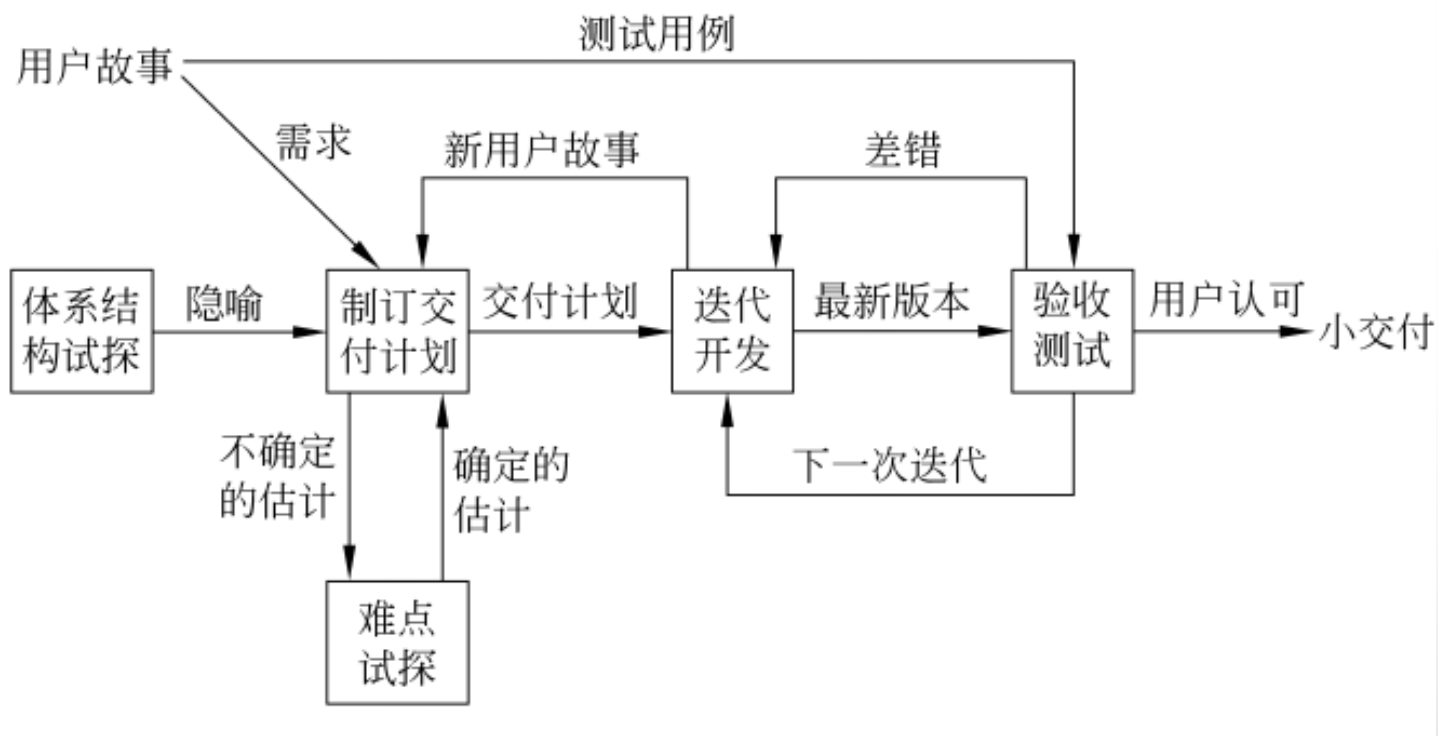
极限编程（eXtreme Programming,XP）是敏捷过程中最富盛名的一个，其名称中“极限”二字的含义是指把好的开发实践运用到极致。

目前，极限编程已经成为一种典型的开发方法，广泛应用于需求模糊且经常改变的场合。

1.4 软件过程

1.4.7. 敏捷过程与极限编程

极限编程的整体开发过程



1.4 软件过程

1.4.7.敏捷过程与极限编程

极限编程的迭代过程

图1.11描述了极限编程的整体开发过程。首先，项目组针对客户代表提出的“用户故事” 进行讨论，提出隐喻，在此项活动中可能需要对体系结构进行“试探” 。然后，项目组在隐喻和用户故事的基础上，根据客户设定的优先级制订交付计划。接下来开始多个迭代过程（通常每个迭代历时1~3周），在迭代期内产生的新用户故事不在本次迭代内解决，以保证本次开发过程不受干扰。开发出的新版本软件通过验收测试之后交付用户使用。

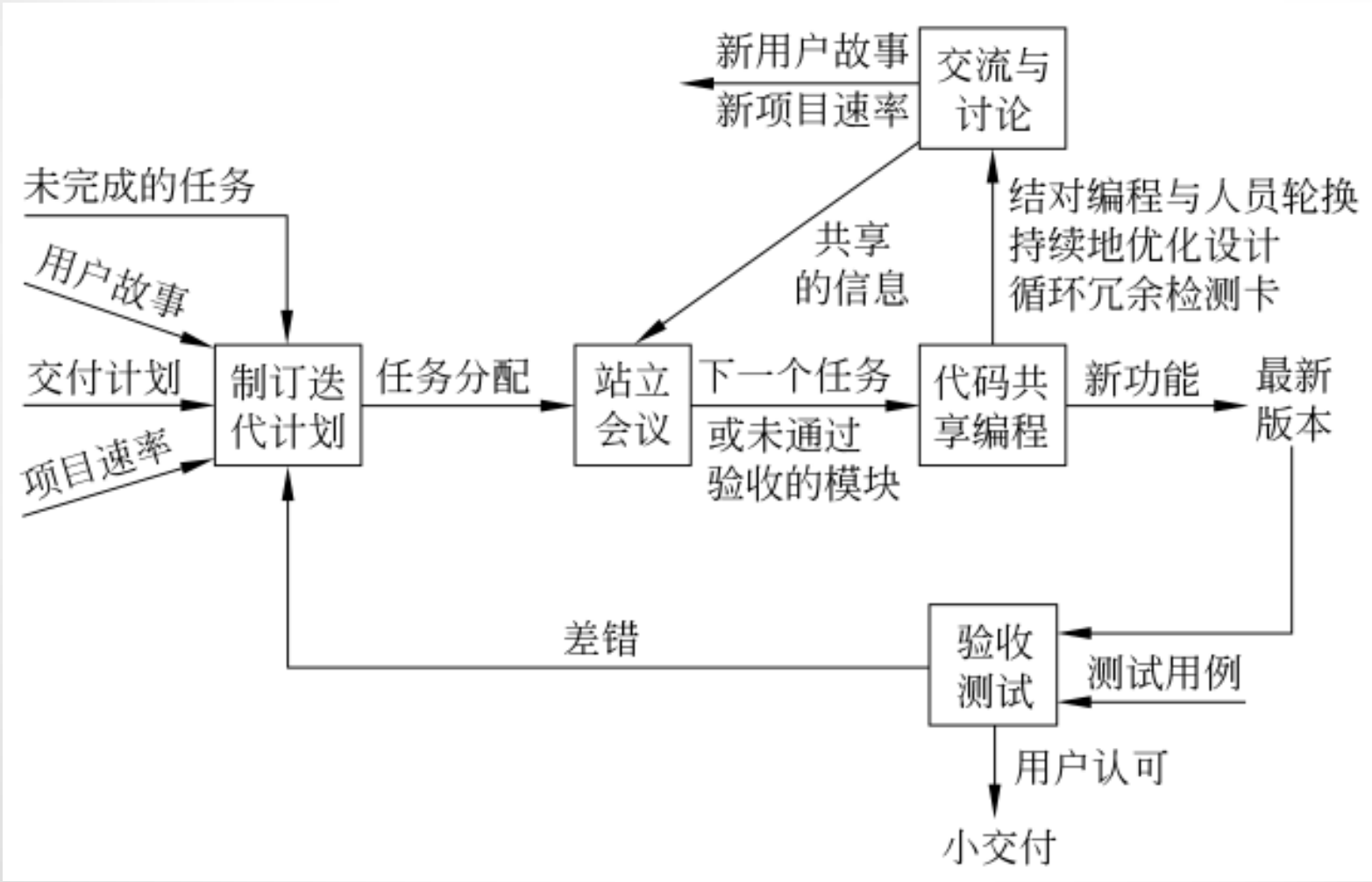

新用户故事
新项目速率
交流与讨论
未完成的任务
用户故事
交付计划
项目速率
制订迭代计划
任务分配
站立会议
共享的信息
下一个任务
或未通过验收的模块
结对编程与人员轮换
持续地优化设计
循环冗余检测卡
代码共享编程
新功能
最新版本
差错
验收测试
测试用例
用户认可
小交付

图1.11

1.4 软件过程

1.4.8. 微软过程

a）微软过程准则：

项目计划应该兼顾未来的不确定因素。
用有效的风险管理来减少不确定因素的影响。
经常生成并快速地测试软件的过渡版本，从而提高产品的稳定性和可预测性。
采用快速循环、递进的开发过程。
用创造性的工作来平衡产品特性和产品成本。
项目进度表应该具有较高稳定性和权威性。
使用小型项目组并发地完成开发工作。
在项目早期把软件配置项基线化，项目后期则冻结产品。
使用原型验证概念，对项目进行早期论证。
把零缺陷作为追求的目标。
里程碑评审会的目的是改进工作，切忌相互指责。

1.4 软件过程

1.4.8. 微软过程

b）**微软软件生命周期：**如下图1.13所示

- 规划阶段
- 设计阶段
- 开发阶段
- 稳定阶段
- 发布阶段

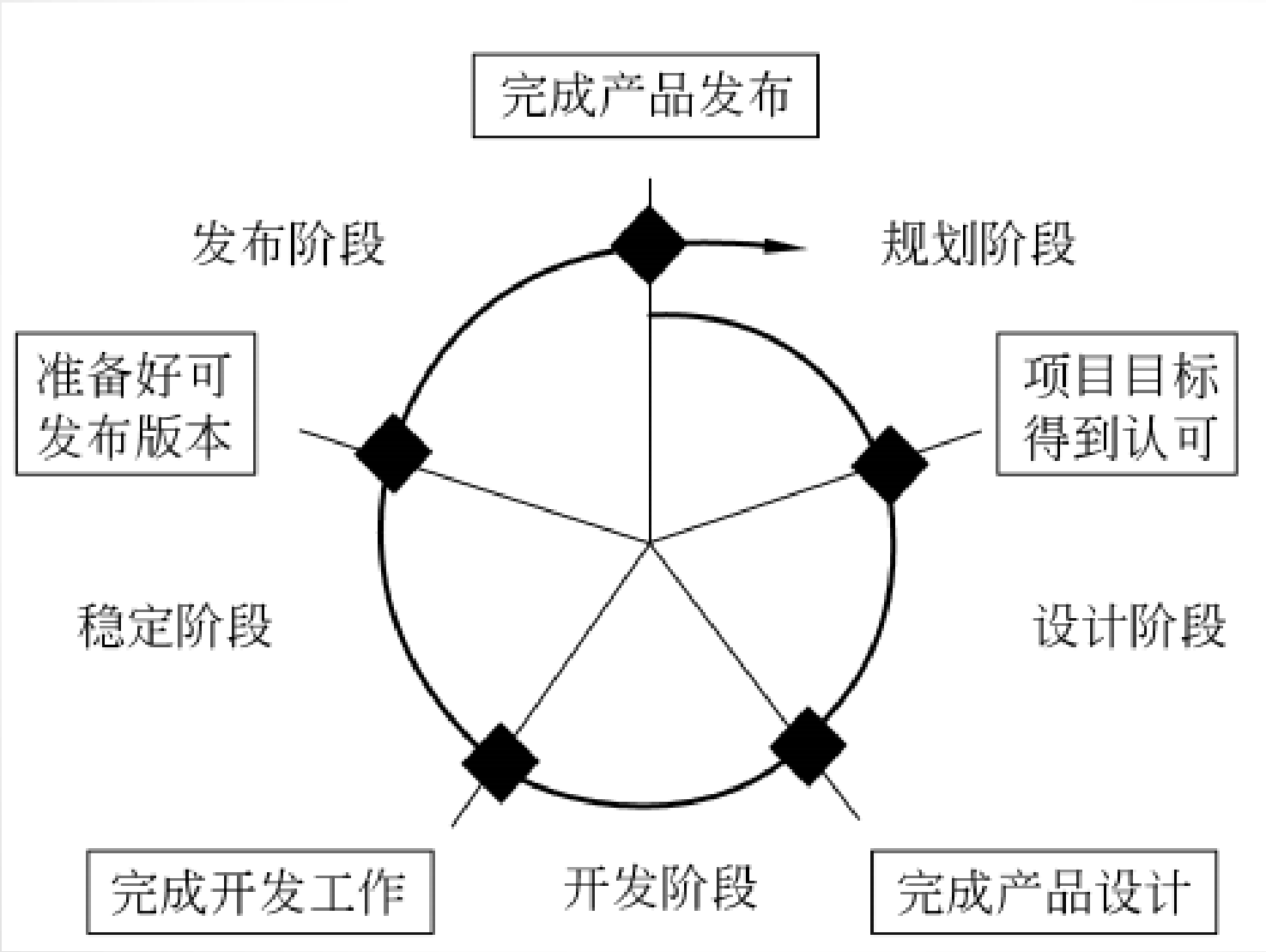

完成产品发布
发布阶段
规划阶段
准备好可
发布版本
项目目标
得到认可
稳定阶段
设计阶段
完成开发工作
开发阶段
完成产品设计

图1.13

1.4 软件过程

1.4.8. 微软过程

c）微软过程模型：

微软过程的每一个生命周期发布一个递进的软件版本，各个生命周期持续、快速地迭代循环。如下图1.14所示

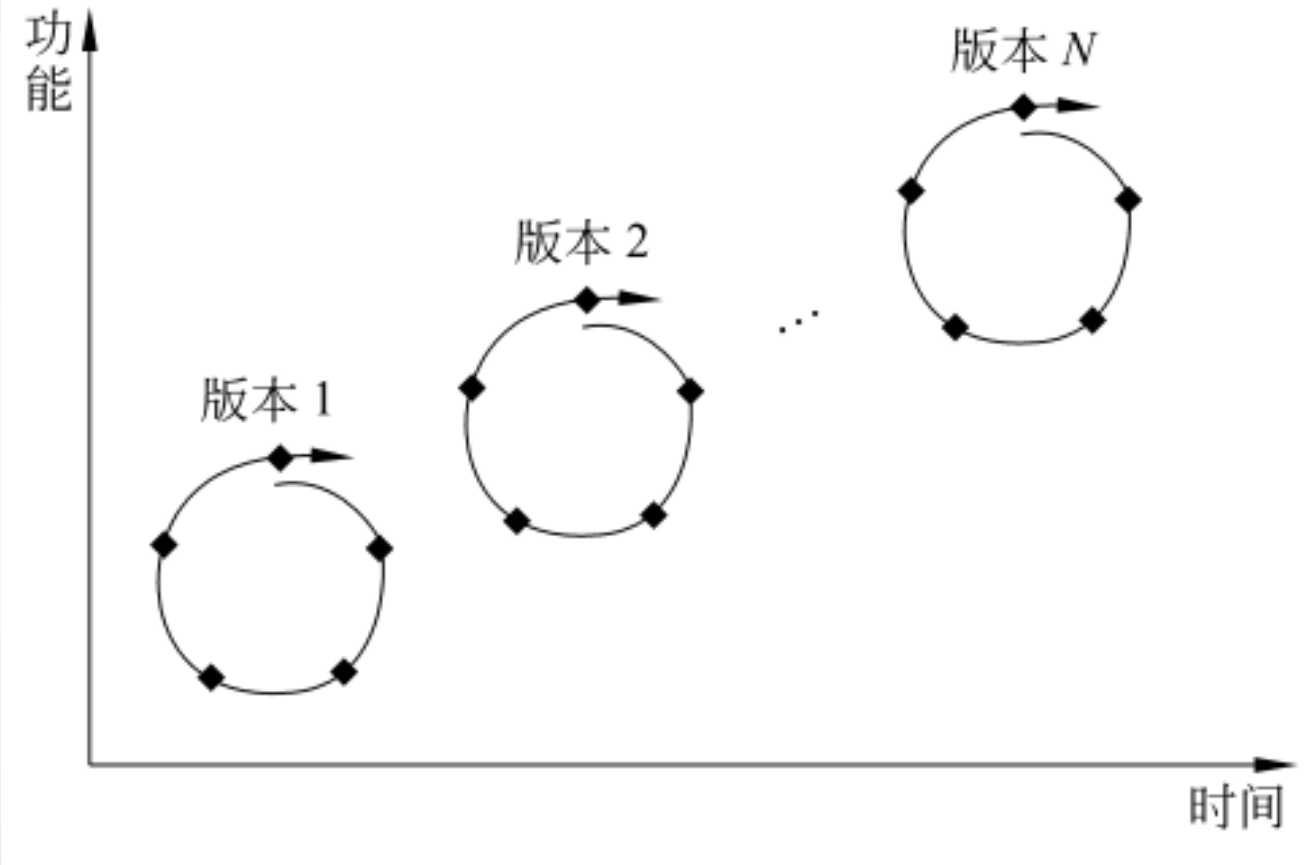


图1.14

本章小结

1. 本章对计算机软件工程学作一个简短的概述，回顾计算机系统发展简史。

2. 本章介绍 软件工程的基本原理和方法有概括的本质的认识，详细讲解生命周期相关知识。

3. 详细讲解8种典型的软件过程模型。

本章结束