软件工程与实践

（第 3 版）

第1章 软件工程基础

教学目标

- 了解软件工程的发展和软件危机
- 掌握软件工程的概念、内容和原理 重点
- 熟悉软件生存周期及阶段任务 重点
- 掌握常用的软件开发模型
- 掌握软件开发准备及Visio应用实验

案例1-1

欧洲航天局首次发射Ariane-5火箭失败。1996年，欧洲航天局首次发射Ariane-5火箭，由于火箭控制系统的软件故障，导致首次发射Ariane-5火箭失败，直接经济损失5亿美元且严重影响了相关航空航天研究进展。

软件危机

软件危机（Software crisis）是指20世纪60年代计算机软件在研发、运行、维护和管理过程中，出现的一系列严重问题的现象。软件危机直接导致软件工程的产生。

*软件危机的教训主要包含两方面的问题：一是需要工程化方式研发软件且必须满足用户对软件日益增长的各种需求，二是强化管理和维护不断快速增长的现有软件。

1.1 软件工程的发展

1.1.1 软件危机

落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象。

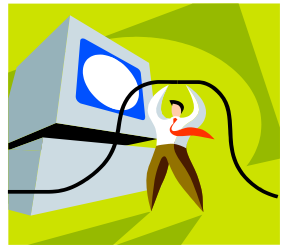
第一次软件危机 （60年代~70年代）

20 世纪60年代以前，计算机刚刚投入实际使用，这个时期主要的软件开发方式是使用机器语言或者汇编语言在特定的机器上进行软件的设计与编写。

从60年代中期开始，大容量、高速度计算机问世，使计算机的应用范围迅速扩大，软件开发急剧增长。高级语言开始出现；操作系统的发展引起了计算机应用方式的变化；大量数据处理导致第一代数据库管理系统的诞生。软件系统的规模越来越大，复杂程度越来越高，软件可靠性问题也越来越突出，程序设计的复杂度也随之增长。原来的个人设计、个人使用的方式不再能满足要求，迫切需要改变软件生产方式，提高软件生产率，软件危机开始爆发 。

1968 年北大西洋公约组织的计算机科学家在联邦德国召开国际会议，第一次讨论软件危机问题，并正式提出“软件工程”一词，从此一门新兴的工程学科——软件工程学——为研究和克服软件危机应运而生，“软件危机”的概念也是在那次会议上由F. L. Bauer提出的。

典型的代表莫过于C语言（1972年）：C语言让程序员能让程序员编写的代码在没有或只有较少机器相关性的同时又有不输于汇编语言的性能，而且丰富的语言特性也使得编程难度大大降低，成功的解决了“抽象性”和“可移植性”的问题。



1.1.1 软件危机

第一次软件危机 （60年代~70年代）

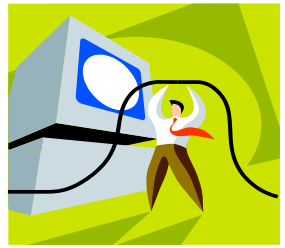
早期出现的软件危机主要表现在：

① 软件开发费用和进度失控。费用超支、进度拖延的情况屡屡发生。有时为了赶进度或压成本不得不采取一些权宜之计，这样又往往严重损害了软件产品的质量。

②软件的可靠性差。尽管耗费了大量的人力物力，而系统的正确性却越来越难以保证，出错率大大增加，由于软件错误而造成的损失十分惊人。

③生产出来的软件难以维护。很多程序缺乏相应的文档资料，程序中的错误难以定位，难以改正，有时改正了已有的错误又引入新的错误。随着软件的社会拥有量越来越大，维护占用了大量人力、物力和财力。

进入80年代以来，尽管软件工程研究与实践取得了可喜的成就，软件技术水平有了长足的进展，但是软件生产水平依然远远落后于硬件生产水平的发展速度



1.1.1 软件危机

第二次软件危机（80年代~90年代）

这次危机可以归因于软件复杂性的进一步增长。这个时候的大规模软件常常由数百万行代码组成，有数以百计的程序员参与其中，怎样高效、可靠的构造和维护这样规模的软件成为了一个新的难题。

著名的《人月神话》中提及，IBM公司开发的OS/360系统共有4000多个模块，约100万条指令，投入5000人年，耗资数亿美元，结果还是延期交付。在交付使用后的系统中仍发现大量（2000个以上）的错误。

1.1.1 软件危机

第二次软件危机（80年代~90年代）

软件危机不仅没有消失，还有加剧之势。主要表现在：

①软件成本在计算机系统总成本中所占的比例居高不下，且逐年上升。由于微电子学技术的进步和硬件生产自动化程度不断提高，硬件成本逐年下降，性能和产量迅速提高。然而软件开发需要大量人力，软件成本随着软件规模和数量的剧增而持续上升。从美、日两国的统计数字表明，1985年度软件成本大约占总成本的90%。

②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需要，软件产品供不应求的状况使得人类不能充分利用现代计算机硬件所能提供的巨大潜力。

1.1 软件工程的发展

1.1.1 软件危机

第二次软件危机（80年代~90年代）

这时候人们典型需求的是更好的“可组合性”(Composability)、“可延展性”(Malleability)以及“可维护性”(Maintainability)。

程序的性能已经不是一个大问题了，因为摩尔定律能帮你搞定它（70年代编写的C程序仍然能在现在的计算机上运行，而且它还更快！）。

为了解决这次危机，面向对象的编程语言（C++、C#、Java等）诞生了，更好的软件工程方法（设计模式、重构、测试、需求分析等等）诞生了，而程序员们也越来越不需要知道硬件是怎么工作的了。软件和硬件的界限越来越牢固，Java编写的代码能在任何JVM支持的平台上运行，程序员也非常乐于享受这样的便利。

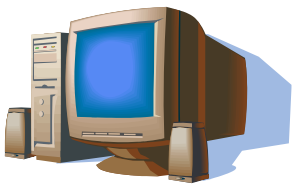
1.1 软件工程的发展

1.1.1 软件危机

第三次软件危机（2005年至今）

“免费的午餐已经结束了”。

摩尔定律在串行机器上宣告失效，多核时代正式来临！这个时候怎样在多核平台上仍然能保持性能的持续增长就成为了这一次软件危机的核心。并行编程给我们带来了许许多多新的技术难题，现阶段想要高效的利用这些多核平台以获得更好的性能，就必须对计算机的硬件有较深入的理解，而广大程序员却更喜欢能有一些更加便利的编程模型（也许是一门新的语言、也许是新的编程模型）来简单高效地进行并行编程。我们正处在这次危机的开端，前路满是荆棘。但是只要有问题，就会有机会。多核时代，你们的机会在哪里呢？



1.1 软件工程的发展

软件危机主要特征在7个方面：

(1) 软件运行经常出现功能、性能不满意或出现故障等现象。

(2) 软件产品的质量、可靠性和安全等方面时常达不到标准。软件产品质量难以保证，甚至在开发过程中就被迫中断。

(3) 软件开发管理差，对成本和进度难估计准确。

(4) 系统时常出现无法维护、升级或更新现象。

(5) 开发效率低，无法满足应用迅速发展与提高实际需要。

(6)研发成本难控制，在总成本中所占的比例不断大幅上升。

(7) 软件开发没有标准、完整、统一规范的文档资料。软件不仅只是程序，还应有一整套规范文档资料和售后服务。

1.1 软件工程的发展

2．软件危机产生的原因

产生软件危机的主要原因有：

(1)软件开发规模、复杂度和需求量不断增加及变化；

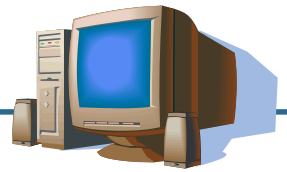
(2)软件需求分析与设计不完善有欠缺周，致使软件开发、维护和管理或文档出现问题；

(3)没有按照工程化方式运作，开发过程无统一的标准和准则、规范方法；

(4)研发人员与用户或研发人员之间互相的交流沟通不够或文档资料不完备；

(5)软件测试调试不规范不细致，提交的软件质量不达标；

(6)忽视软件运行过程中的正常维护和管理。



1.1 软件工程的发展

3．解决软件危机的措施

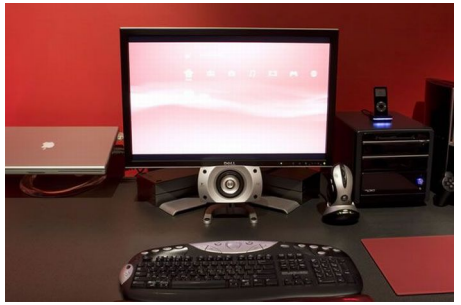
解决软件危机的主要措施有3个方面：

(1) 技术方法。运用软件工程的技术、方法和标准规范。

(2) 开发工具。选用先进高效的软件工具，同时采取切实可行的实施策略。

(3) 组织管理。研发机构需要组织高效、管理制度和标准严格规范、职责明确、质量保证、团结互助、齐心协力，注重文档及服务。

注意：为了避免和解决软件开发中再出现软件危机，不仅需要标准规范的技术措施，更要有强有力的组织管理保障。各方面密切配合、齐抓共管，切实以软件工程方式方法和规程进行运作，才能确保软件质量和信息化健康发展。



1.1.2 软件工程的发展过程

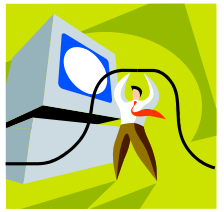
软件技术的发展经历了程序设计阶段、程序系统阶段、软件工程阶段和创新完善软件工程4个阶段，其典型技术如表1-1所示。

表1-1 软件技术各发展阶段的典型技术

阶段	程序设计阶段	程序系统阶段	软件工程阶段	创新完善软件工程阶段
软件典型技术	面向批处理 有限的分布 自定义软件	多用户 实时处理 数据库 软件产品	分布式系统 嵌入“智能” 低成本硬件 消费者的影响	强大桌面系统面向对象技术 专家系统、神经网络、并行计算、网格计算等高新技术

1.1 软件工程的发展

软件工程发展经历4个重要阶段：



1．传统软件工程

传统软件工程是指软件工程产生的初期，也称为第一代软件工程。

2．对象工程

对象工程也称为第二代软件工程。20世纪80年代中到90年代，以Smalltalk为代表的面向对象的程序设计语言相继推出，使面向对象的方法与技术得到快速发展。

3．过程工程

过程工程也称为第三代软件工程。随着网络等高新技术的出现及信息技术的广泛应用，软件规模和复杂度不断增大，开发时间相应持续增长，开发人员的增加，致使软件工程开发和管理的难度不断增强。

4．构件工程

构件工程也称为第四代软件工程，[illegible]取得重要进展，软件系统的开发可利用已有的可复用构件组装完成，而无需从头开始构建，可提高效率和质量、降低成本。

构件是指语义完整、语法正确和有可重用价值的单位模块，是软件重用过程中可以明确辨识的系统；结构上是语义描述、接口和实现代码的复合体。

1.1 软件工程的发展

计算机辅助软件工程简称CASE（Computer Aided Software ngineering）将软件工具和代码生成器进行集成，为很多软件系统提供了可靠的解决方案；专家系统和人工智能软件的应用更加广泛；人工神经网络软件开阔了信息处理的新途径；并行计算、网络技术、虚拟技术、多媒体技术和现代通信技术等新技术新方法改变了原有的工作方式。

1.2.1 软件的概念特点和分类

程序是按照事先预定功能性能等要求设计和编写的指令序列；

1. 软件的概念

软件（Software）是计算机及手机等终端设备运行的程序、数据、文档和服务的集合，包括指令程序、数据、相关文档和完善的售后服务的完整集合。即；

软件=程序 + 数据 + 文档 + 服务。

其中，数据则是使程序正常处理信息的数据结构及信息表示；文档（Document）是与程序开发、维护和使用有关的技术资料。服务主要指对各种软件用户的服务，包括提供软件产品使用说明书、推销服务及售后技术支持等

软件分为系统软件、支撑软件（开发环境）和应用软件等。其中应用软件常称为信息系统主要是指具体的应用系统。

注意：程序与软件不同，程序只是软件的组成部分。“软件就是程序”的观点为误解，也严重影响了软件工程的正常进行和发展。文档必不可少，只有程序不能称为软件。

软件工程师是软件研发人员的统称，按照所处的领域不同可以分为软件策划/架构师、系统分析师、软件设计师、系统架构师、程序员、测试员、维护与管理员、售后技术服务等。

2．软件的主要特点

在实际研发、运行、维护、管理和使用中，软件主要特点：

(1) 智能性。软件是人类智能劳动的产物、代替和延伸。

(2) 抽象性。软件属于逻辑实体，无形性和智能性致使软件难以认识和理解。

(3) 人工方式。软件的开发、维护及设置管理等方面目前尚未完全脱离手工方式。

(4) 复杂性和系统性。逻辑处理和数据结构及构成等相对复杂

(5) 泛域性。软件应用很广泛，在信息化中可服务于各种领域、行业和层面。

(6) 复制性。软件成本相对比较昂贵，软件是人类创造性的可复制的特殊产品。

(7) 非损及更新性。软件不存在物理性磨损和老化问题，但可以退化需要更新升级。

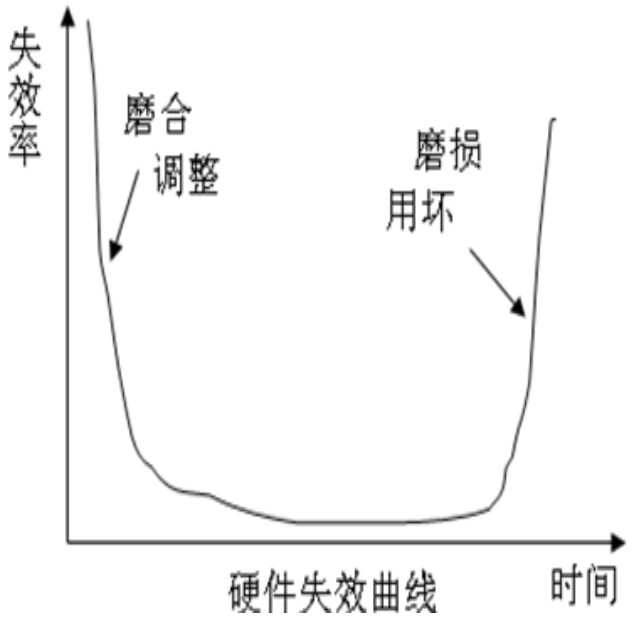


图1-1 硬件失效率曲线

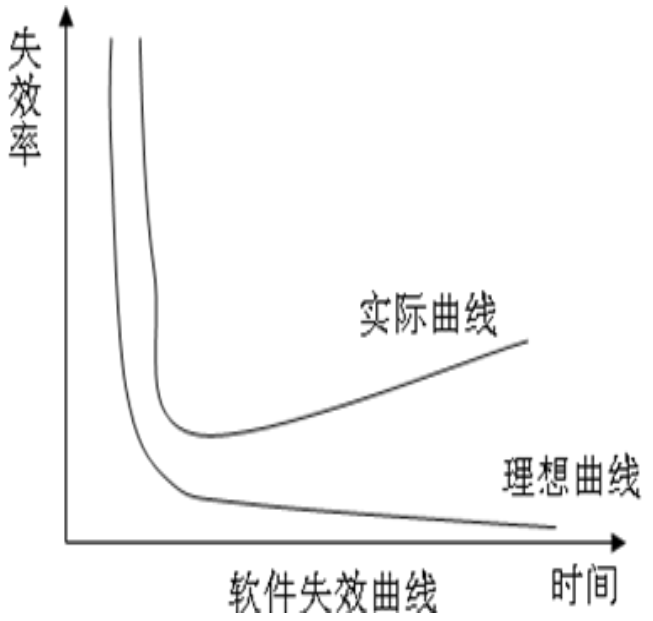


图1-2 软件失效率曲线

3.软件的分类

(1) 按照软件功能划分-系统软件、支撑软件、应用软件

(2) 按照软件规模划分

（微型、小型、中型、大型、超大型5种**见表1-2**）

(3)按照软件工作方式划分-实时处理软件、分时软件、交互式软件、批处理软件

(4)按照软件服务对象的范围划分-项目软件、产品软件。

(5)按照软件运行的终端设备划分。分为服务器软件、计算机软件、手机软件、机器人软件和其他嵌入设备(电子化设备)软件。

表1-2 软件规模分类

类　别	研发人数	研制期限	规模（代码行）
微　型	1 - 3	1 – 4 周	小于500行
小　型	3 - 5	1 – 8月	500-1万行
中　型	6 - 10	1 – 2 年	1万-5万行
大　型	11 - 20	2 – 3 年	5万-10万行
超大型	20人以上	3年以上	10万行以上

软件工程基本思想特别强调在软件开发过程中**利用工程化**标准准则、技术、方法和管理及文档与应用的**重要性**

1.2.2 软件工程的概念、特点和目标

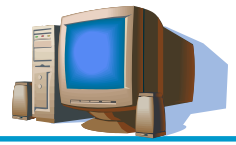
1. 软件工程的概念

按照中国国家标准GB/T 11457—1995《软件工程术语》的定义：**软件工程**（Software Engineering）是软件开发、运行、维护和引退的系统方法。

更新升级

《计算机科学技术百科全书》中对**软件工程**的定义是：应用计算机科学、数学及管理科学等原理，开发软件的过程。软件工程借鉴传统工程的原则、方法，以提高质量、降低成本。其中，**计算机科学和数学**用于构建模型与算法，**工程科学**用于制定规范、设计范型(paradigm)、评估成本及确定权衡，**管理科学**用于计划、资源、质量、成本等管理。

软件工程 = 工程原理 + 技术方法 + 管理技术



2．软件工程的特点和目标

软件工程学是软件工程化的思想、规范、过程、技术、环境和工具的集成，是将具体的技术和方法结合形成的一个完整体系。

软件工程学科主要特点:实践性和发展性,其问题来源并应用于实践,其特点体现为“3多”：一是多学科,不仅包含有关课题还涉及到计算机科学、工程科学、管理科学、数学等学科;二是多目标,不仅关心项目产品及其功能,还有注重质量、成本、进度、性能、可靠性、安全性、通用性、可维护性、有效性和界面等;三是多阶段,软件开发不只是编程,而是由可行性研究、计划立项、需求分析、总体设计、详细设计、编程（实现）、测试、运行、维护等过程。

软件工程的目的是在规定的时间和开发经费内，开发出满足用户需求的、高质量的软件产品(高效高质量低成本地研发软件产品)。其目标是实现软件研发与维护的优质高效和自动化。

两高一低

1.2.3 软件工程学及其内容和方法

1. 软件工程学概述

软件工程学是一个专门研究用工程化方法，构建和维护有效、实用和高质量的软件的学科，主要涉及软件系统的分析与设计方法、编程与实现技术、数据库及网络技术、软件开发工具、系统平台、标准、设计模式等方面。其主要内容包括软件开发技术和软件工程管理两个方面(如表1-3).软件开发技术包括软件工程方法、软件工具和软件开发环境；软件工程管理学包含软件工程经济学和软件管理学。

表1-3 软件工程学科的主要内容

软件工程原理	软件目标、原则、学科基础
软件工程过程	开发过程、运作过程、维护过程，如获取、供应、管理、开发、运作、维护、支持、剪裁
软件工程技术	开发技术、管理技术、度量技术、维护技术、应用技术
软件工程方法	开发方法、管理方法、度量方法、维护方法、应用方法、环境方法
软件工程模型	领域模型、需求模型、设计模型、实现模型、测试模型
软件工程管理	项目管理、质量管理、文档管理
软件工程度量	规模、复杂度、进度、费用、工作量
软件工程环境	硬件、网络、支撑软件
软件工程应用	应用软件工程基本原理、方法、技术解决特定领域问题

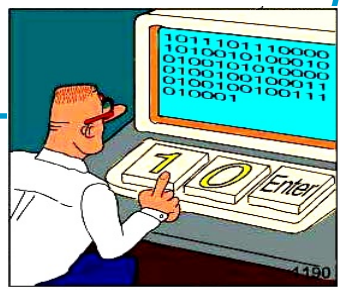
2. 常用的软件工程方法

软件工程方法学是研发、管理与维护软件的系统方法，确定软件开发阶段，规定各阶段的目标、任务、技术、方法、产品、验收等步骤和完成准则.具有方法、工具和过程三个要素,也称软件工程三要素：

（1）软件工程方法：包括软件开发“如何作”的技术和管理准则及文档等技术方法；

（2）软件工具：为方法的运用提供自动或半自动的软件支撑工具的集成环境；

（3）过程与管理：主要完成任务的工作阶段、工作内容、产品、验收的步骤和完成准则。也有将这一要素确定为“组织管理”，也有的称为“软件工程过程”。

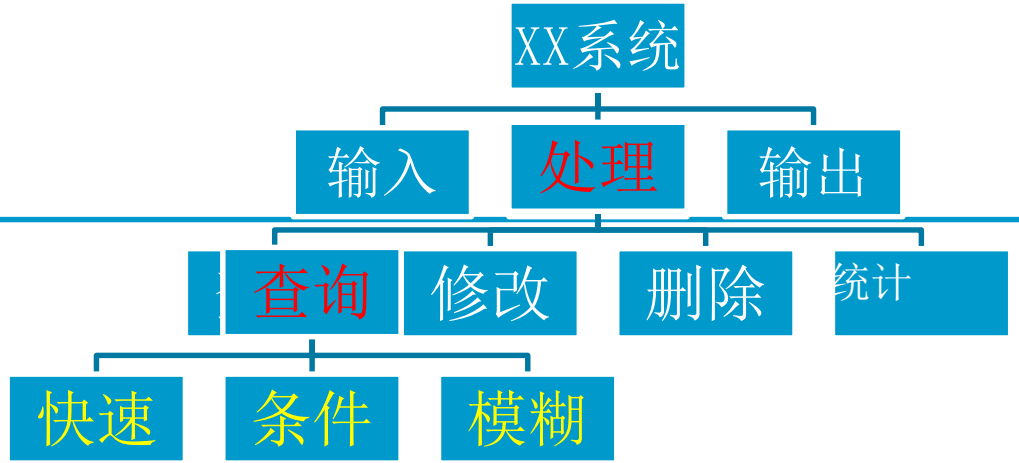


1.2 软件及软件工程概述

常用软件工程方法主要分为7种类型。

（1）面向功能方法。面向功能的软件开发方法也称为结构化方法，主要采用结构化技术，包括结构化分析、结构化设计和结构化实现，按照软件的开发过程、结构和顺序完成开发任务。

（2）面向数据方法。从目标系统输入、输出数据的结构，导出程序框架结构，再补充其他细节，得到完整的程序结构图。此方法也可与其他方法结合,用于模块的详细设计和数据处理等。对输入输出数据结构明确的中小型系统很有效，如商用文件表格处理等。



1.2 软件及软件工程概述

（3）面向对象方法（Object-Oriented Method，OOM）是一种将面向对象的思想应用于软件开发过程中，指导开发活动的系统方法。将对象作为数据及其操作相结合的软件构件，用对象分解取代了传统方法的功能分解。基本思想是：对问题领域进行自然的分割，以更接近人类通常思维的方式建立问题领域的模型，以便对客观的信息实体（事物）进行结构和行为的模拟，从而使设计的软件更直接地表现问题的求解过程。OOM以对象作为最基本的元素，是分析和解决问题的核心。OOM要素是对象、类、继承以及消息通信。概括为：

面向对象 = 对象 + 类 + 继承 + 消息通信

实际上，按照这四个概念设计和实现的软件系统，都可认为是面向对象的。OOM由OOA（面向对象的分析）、OOD(面向对象设计)和OOP(面向对象的程序设计)三部分组成。

（4）面向问题方法.也称问题分析法（Problem Analysis Method, PAM）是80年代末由日立公司提出的，是在Yourdon方法、Jackson方法和自底向上的软件开发方法基础上的改进。其基本思想是：以输入输出数据结构指导系统的问题分解，经过系统分析逐步综合。

（5）面向方面的开发方法.(Aspect-Oriented Programming, AOP)是面向对象系统的扩展，在现有的AOP实现技术中，可通过创建Aspect库或专用Aspect语言实现面向方面的编程。

（6）基于构件的开发方法.基于构件的开发（Component-Based Development, CBD)或基于构件的软件工程(Component-Based Software Engineering, CBSE)方法是软件开发新范型。

软件复用（Software Reuse）或软件重用是指将已有的软件构件用于构造新的软件系统的过程。软件复用方法采用复用方式包括：

①复用分析：利用原有的需求分析结果，进一步深入分析比对查找异同及特性等。

②复用结构：主要复用系统模块的功能结构或数据结构等，并进行改进提高。

③复用设计：由于复用受环境影响小，设计结果比源程序的抽象级别高，因此可通过从现有系统中提取全部或不同粒度的设计构件，或独立于具体应用开发设计构件。

④复用程序：包括目标代码和源代码的复用,可通过连接(Link)、绑定(Binding)、包含(include)等功能，支持对象链接及嵌入(OLE)技术实现。

（7）可视化方法

3．软件研发工具

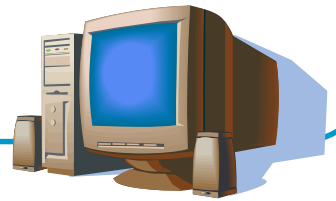
软件工具（Software tools）是指支持软件的开发、维护、管理而专门研发的程序系统。目的是提高软件开发的质量和效率，降低软件开发、维护和管理的成本, 支持特定的软件工程方法,减少手工方式管理的负担。软件工具构成:工具(主体)、工具接口和工具用户接口三部分。工具通过工具接口与其他工具、操作系统以及通信接口、环境信息库接口等进行相连交互。

软件工具种类繁多、涉及面广，可组成“工具箱”或“集成工具”，如编辑、编译、正文格式处理，静态分析、动态跟踪、需求分析、设计分析、测试、模拟和图形交互等。按照应用阶段分为：计划工具、分析工具、设计工具、测试工具等，按照功能分为：分析设计、Web开发、界面开发、项目管理、软件配置、质量保证、软件维护等。

4．软件开发环境

软件开发环境（Software Development Environment）也称为软件工程环境。是相关的一组软件工具集合，支持一定的软件开发方法或按照一定的软件开发模型组织而成。是包括方法、工具和管理等多种技术的综合系统。其设计目标是简化软件开发过程，提高软件开发质量和效率。

【案例1-2】基于Android的手机WebAPP是现代信息化时代极为常用的软件，其中手机通讯系统的“通讯录”，已经成为人们经常记录并使用的多种联系方式的不可缺少的手机软件，其软件开发环境为Java Jdk、Android Sdk和Eclipse。



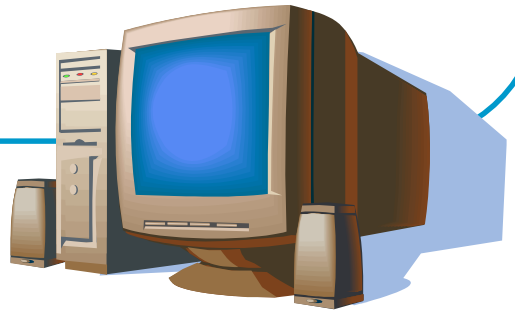
软件开发环境应具备4个特点：

(1) 适应性。适应各种用户的不同要求，环境中的工具可修改、增加、减少和更新；

(2) 坚定性。环境可自我保护，不受用户和系统影响，可进行非预见性的环境恢复；

(3) 紧密性。各种软件工具可密切配合工作,提高效率；

(4) 可移植性。指软件工具可以根据需要进行移植。



1.2 软件及软件工程概述

常用的软件工程环境具有三级结构，如图1-3所示：

(1)核心级。主要包括核心工具组、数据库、通讯工具、运行支持、功能和与硬件无关的移植接口等。

(2)基本级。一般包括环境的用户工具、编译、编辑程序和作业控制语言的解释程序等。

(3)应用级。通常指应用软件的开发工具。

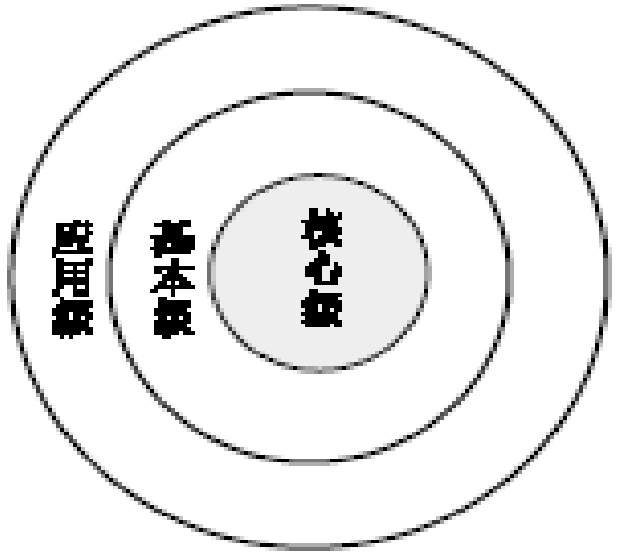


图1-3 典型的软件工程环境

5. 软件工程管理概述

软件工程管理学包括软件管理学、软件经济学和软件度量学。其目的是高效、高质量、低成本地研发出用户满意的软件产品。软件工程管理的任务是有效地组织人员，按照适当的技术、方法，利用好的软件工具“又好又快”地完成预定的软件项目。

软件工程管理的主要内容包括：

（1）组织人员。
（2）计划管理。
（3）费用管理。
（4）软件配置管理。

项目管理

1.2.4 软件过程及实际开发过程

软件过程（software process）ISO9000定义为：将输入转化为输出的一组彼此相关的资源和活动。

软件过程通常包括4类基本过程：

(1)软件规格说明（需求分析）：规定软件的功能、性能、可靠性及其运行环境等。

(2) 软件开发：研发满足规格说明的具体软件。

(3) 软件确认：确认软件能够完成客户提出的需求。

(4) 软件演进：为满足用户的变更要求，软件必须在使用过程中引进新技术新方法并根据新业务及时升级更新。

具体开发过程如图1-4所示。

1.2 软件及软件工程概述

软件过程特征包括可理解性、可见性（过程的进展和结果可见）、可靠性、可支持性（易使用CASE工具支持）、可维护性、可接受性（为软件工程师接受）、开发效率和健壮性（抵御外部意外错误的能力）等特性。

软件实际开发过程是最重要的软件过程，主要包括项目启动、需求调研分析、设计（概要设计及详细设计）、编码（实现）、测试、部署、测试和结束等过程。如图1-4所示。

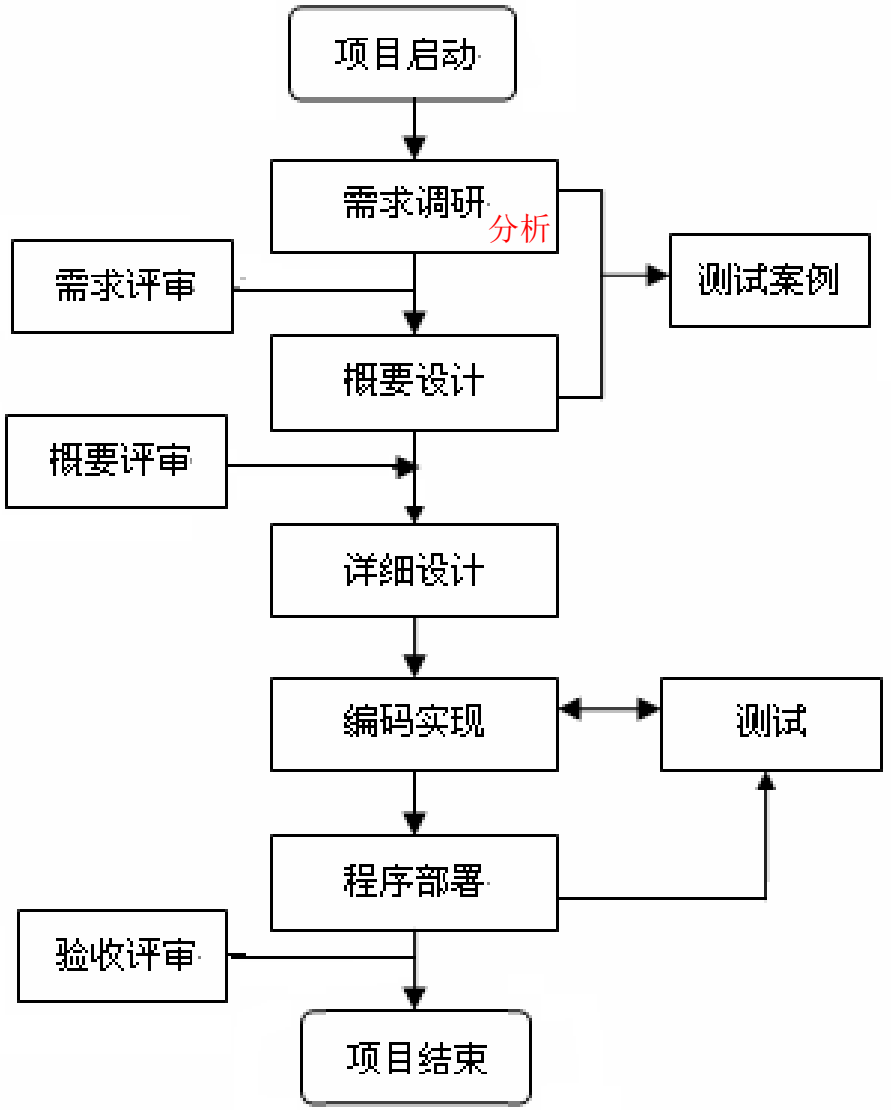


图1-4 系统开发过程

案例1-3

“企业人事管理信息系统”总体功能需求和目标要求。主要功能是用于支持企事业单位完成劳动人事管理工作，实现的主要目标包括：

(1) 支持企事业高效率完成劳动人事管理的日常业务，包括新职工调入时人事的管理，职员调出、辞职、退休等。

(2) 支持企事业进行劳动人事招聘及其相关的科学决策，如企事业单位领导根据现有的岗位员工需求情况决定招聘的岗位及人数等。

根据新系统总体功能需求等要求，通过调研、论证可以基本确定系统开发过程的总体框架。

1.2 软件及软件工程概述

软件开发过程的具体工作任务、参与人员及生成文档或程序，可以通过一个表具体详细地列出来，以便于清楚各阶段具体做什么工作（任务），如表1-4所示。

表1-4 软件开发工作任务、人员及输出

步 骤	任务及说明	参与者	生成文档或程序
可行性分析	对项目的技术、功能需求及市场，进行调研和初步分析，确定是否需要启动项目	部门主管 核心技术人员	可行性分析报告 技术调研报告
启动项目	正式启动项目，由部门主管制定合同，项目经理制定初步计划，初步计划包括设计和开发时间的初步估计	部门主管 核心技术人员	项目计划书 项目合同
需求分析	对项目详细需求分析，编写需求文档，对B/S 结构的系统应制作静态演示页面。需求分析文档和静态演示页面需要通过部门主管审批才能进行下一步骤	项目经理 项目小组核心成员	需求分析说明书 静态演示页面 项目计划修订版本
概要设计	根据需求分析进行概要设计。编写目的是说明对系统的设计考虑，包括程序系统流程、组织结构、模块划分、功能分配、接口设计。运行设计、数据结构设计和出错处理设计等，为详细设计提供基础。概要设计经过评审后，项目经理通过部门主管一起指定项目小组成员。	项目经理 项目小组核心成员	概要设计说明书
详细设计	详细设计编制目的是说明一个软件各个层次中的每一个程序（每个模块或子程序）的设计方案，如果一个软件系统比较简单，层次很少，可以不单独编写，有关内容可并入概要设计说明书。	项目经理 项目小组成员	详细设计文档 项目计划确定版本
编码实现	根据详细设计编程实现，同时有美工对操作界面进行美化	项目经理、程序设计员、美工	软件版本说明 软件产品规格说明
调试	项目经理提交测试申请，由测试部门对项目进行测试，项目小组配合测试部门修改软件中的错误	项目经理 程序开发人员 测试部门	测试申请 测试计划 测试报告
项目验收	项目验收归档	部门主管、项目经理	项目所有文档和程序

1.2.5 软件工程基本原理及原则

1. 软件工程的基本原理

B.Boehm软件工程7条基本原理。

（1）开发小组的人员应该优化组合且少而精。

（2）用分阶段的生存周期计划进行严格管理。

（3）坚持进行阶段评审。软件的质量保证工作不能等到编码阶段结束之后再进行。

（4）实行严格的软件产品控制。

（5）采用现代程序设计技术。

（6）软件工程结果应能清楚地审查。

（7）承认不断改进软件工程实践的必要性。

2. 软件工程的基本原则

(1) 选取适宜的开发模型。

(2) 采用合适的设计方法。

(3) 提供高质量的工程支撑。

(4) 重视软件工程的管理。

1.3.1 软件生存周期的有关概念

软件生存周期(Software life cycle)是从软件开始研发到停止使用的整个过程.是指软件产品从提出开发需求开始，经过开发、使用和维护,直到最后淘汰的整个周期,因此,也称为软件生命周期或软件生存期。

软件工程中的过程对应软件生存周期中的阶段(Phase) ,也是实现软件生产工程化的重要步骤，并赋予各阶段相对独立的任务。可以将一个软件的生存周期划分为市场调研、立项/策划、规划/计划、需求分析、概要设计、详细设计、编程、单元-集成测试、运行、维护这几个过程，前一过程的终点就是后一过程的起点。完成阶段性工作的标志称为里程碑(Milestone)，某些重要的里程碑又称为基线(Baseline)。

1.3.2 软件生存周期的阶段划分

软件生存周期划分阶段的方法有多种，可按软件规模、种类、开发方式、开发环境等划分。软件生存周期阶段划分的原则主要包括：

（1）各阶段的任务相对独立。便于分阶段计划、逐步完成。

（2）同一阶段的工作任务性质尽量相同。有利于软件开发和组织管理，明确开发人员的分工与职责，以便协同工作、保证质量。

1.3.3 软件生存周期各阶段的任务

软件生存周期组成包括软件策划、软件开发和运行维护三个时期。软件生存周期各阶段的主要任务。

GB 8567-2006将软件生存周期分为7个阶段如图1-5：

(1) 开发策划.主要完成问题定义、可行性论证、制定开发计划和项目申报立项,明确“要解决什么问题(开发什么软件)”.

(2) 需求分析。需求分析和定义阶段任务不是具体地解决问题，而是确定软件须具备的具体功能、性能等，即“必须做什么”及其他指标要求。

(3) 概要设计.主要设计软件的总体(外部)结构,结构组成模块,模块层次结构、调用关系及功能。并设计总体数据结构等。

主要怎么做

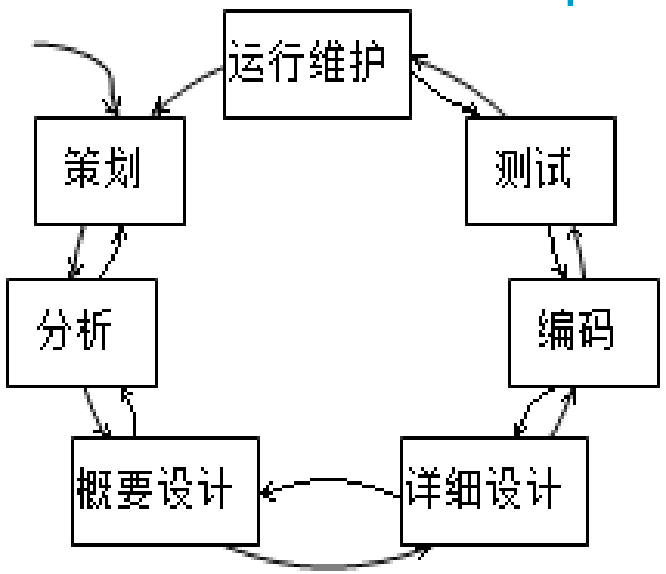


图1-5 软件生存周期阶段关系

1.3 软件生存周期及任务

具体怎么做

(4) 详细设计。对模块功能、性能、可靠性等进行具体技术描述，并转化为过程描述。

(5) 编写程序。又称编码（具体实现），将模块的控制结构转换成程序代码。

(6) 测试。为了保证软件需求和质量，在设计测试用例基础上对软件进行检测

(7) 运行维护。对交付并投入使用的软件进行各种维护，并记录保存文档。

1.4 常用软件开发模型

模型是对事物本质特征的一种抽象、模拟和描述,用于表示系统的重要特征描述，包括图表模型、数学模型、实物模型和描述模型。根据软件开发工程化及实际需要，软件生存周期的划分不同，形成不同的软件开发模型-软件生存周期模型或软件开发范型。

1.4.1 瀑布模型概述

瀑布模型（Waterfall model）将生存期的计划时期、开发时期和运行时期，又细分为若干个阶段：计划时期可分为问题定义、可行性研究、需求分析3个阶段，开发时期分为概要设计、详细设计、软件实现、软件测试等阶段，运行时期则需要不断进行运行维护，需要不断修改错误、排除故障，或以用户需求、运行环境改变进行改更调整。图1-6中的实线箭头表示开发流程，每个阶段顺序进行，有时会返工；虚线箭头表示维护工作的流程，根据不同情况返回到不同的阶段进行维护。

1.4 常用软件开发模型

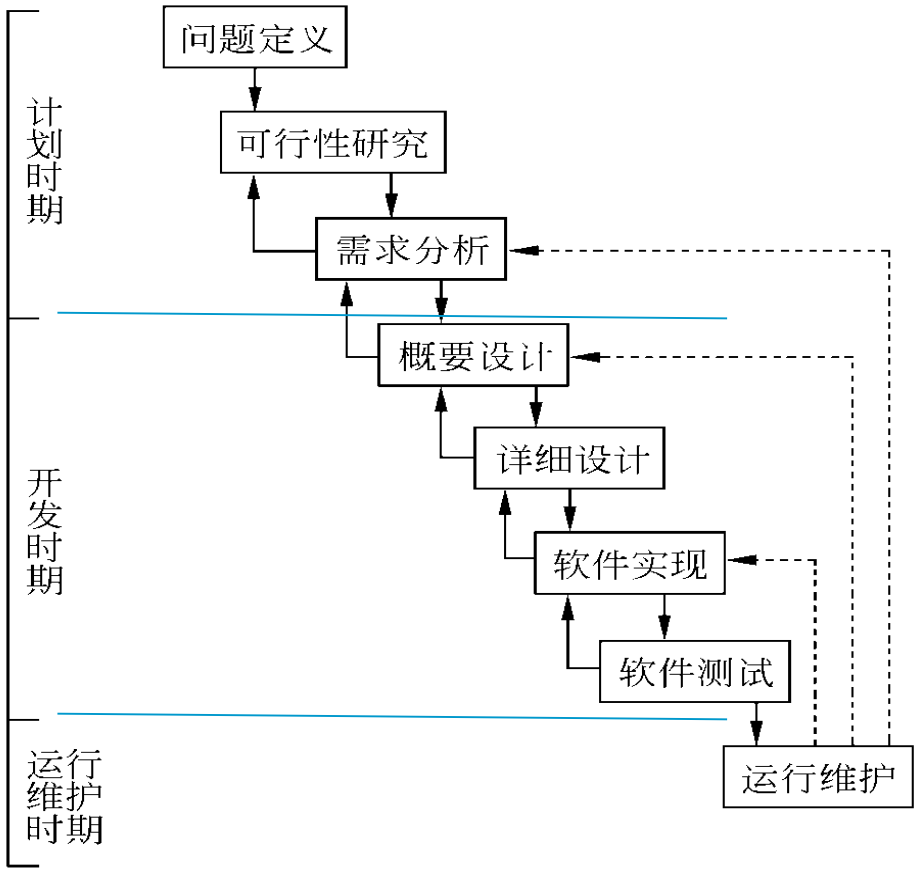


图1-6 瀑布模型

1.4 常用软件开发模型

瀑布模型开发软件特点

利用瀑布模型开发软件3个特点：

（1）开发过程的顺序性。

适用于软件需求明确,开发技术成熟,工程管理较严格情形。

（2）统筹兼顾不过早编程。

（3）严格要求保证质量。

为确保质量，应坚持做到：

① 必须各阶段都按照要求认真完成规定的文档。

② 各阶段须对完成文档复审，及时发现隐患并排除。

瀑布模型缺陷是将充满回溯且相互重叠的软件开发过程硬性地分为多个阶段，随着开发软件规模的增加，造成的问题大增。为了描述软件开发过程中可能回溯对瀑布模型进行了改进，开发各阶段可能循环重复。

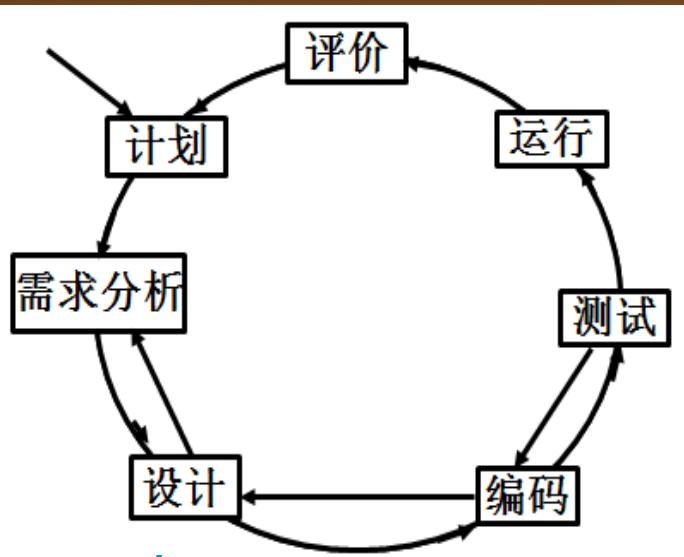
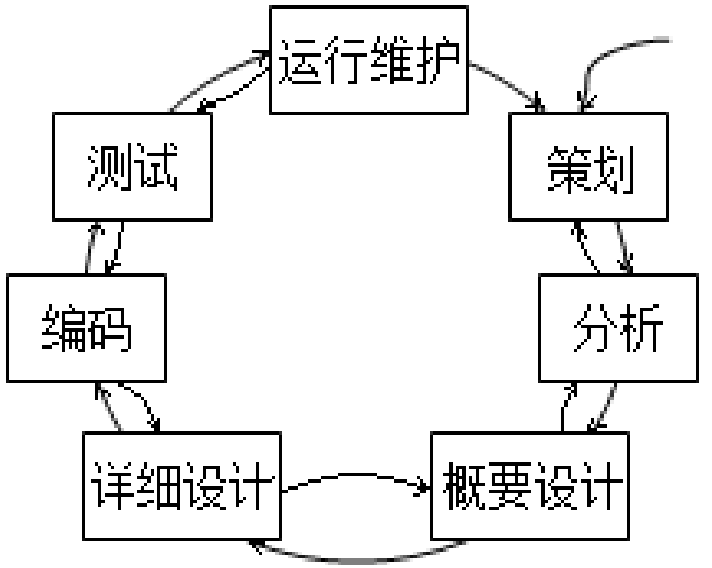


图1-7 循环模型

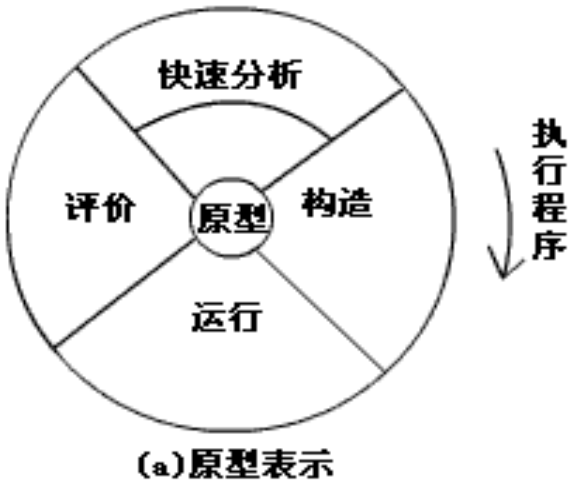


改进模型

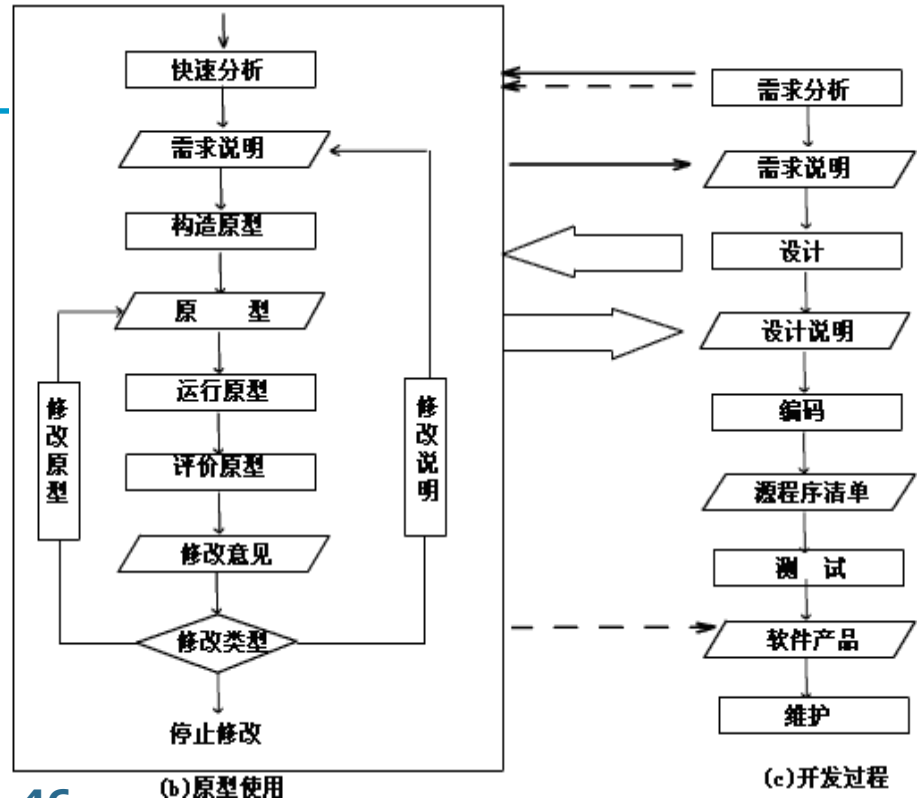
1.4 常用软件开发模型

1.4.2 快速原型模型

最适合于可以先尽快构建成一个原型的应用系统。需要先建造一个快速原型，如操作窗口及界面等，进行客户或潜在用户与系统间的试用交流，用户/客户可通过对原型的评价及改进意见,进一步细化待开发软件的需求，通过逐步调整原型达到客户要求,从中确定客户的具体需求；然后按照需求开发软件。如图1-8。



(a)原型表示



(b)原型使用

(c)开发过程

图1-18 快速原型模型

1.4.3 增量模型概述

增量模型灵活性很强，适用于软件需求不明确、设计方案有一定风险的软件项目。利用增量模型开发的软件被作为一系列的增量构件进行设计、实现、集成和测试，每个构件具有一定功能，并最终能组合成一个具有完整功能软件的模块。如图1-9。

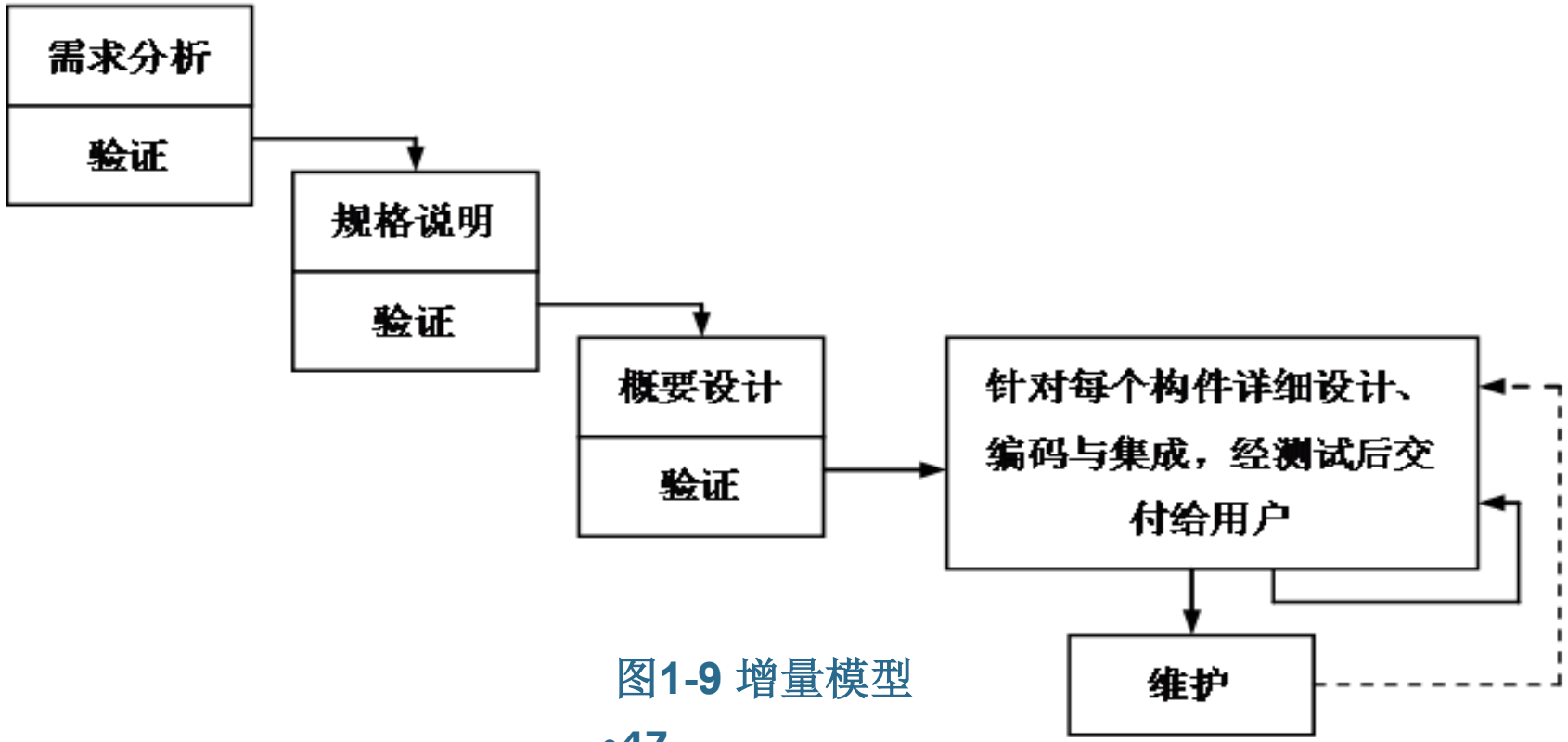


图1-9 增量模型

1.4 常用软件开发模型

同瀑布模型的本质区别:瀑布模型属于整体开发模型,规定在开始下一阶段工作之前，必须完成前一阶段的所有细节。而增量模型属于非整体开发模型，可推迟某些阶段或所有阶段中细节，较早地研发出软件。

增量模型的缺陷有两个方面：

（1）需要软件具备开放式的体系结构。主要因为各构件是逐渐并入已有的软件体系结构中的，所以加入构件不能破坏已构造好的系统部分。

（2）软件过程控制易失去整体性.软件在开发中难免需求变化,增量模型的灵活性可使其适应变化的能力优于瀑布模型和快速原型模型,但也容易退化为边做边改模型.

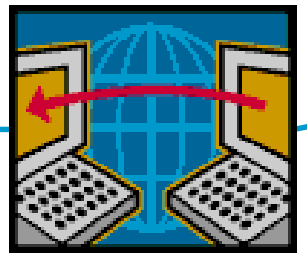
1.4 常用软件开发模型

1.4.4 螺旋模型概述

螺旋模型将瀑布模型和快速原型模型结合，强调了其他模型所忽视的风险分析，适合于大型复杂系统，吸收了“演化(推进)”概念，可使开发人员和客户了解对每个演化层的风险，继而做出反应。将开发过程划分为制定计划、风险分析、实施工程和客户评估四类活动. 将沿着螺旋线继续进行，自内向外逐步延伸，最终得到满意的软件产品。

螺旋模型沿着螺线进行多次迭代,其迭代过程如图1-10所示。

(1) 制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件；
(2) 风险分析：分析评估所选方案，考虑如何识别和消除风险；
(3) 实施工程：实施软件开发和验证；
(4) 客户评估：评价开发工作，提出修正建议，制定下一步计划。



1.4 常用软件开发模型

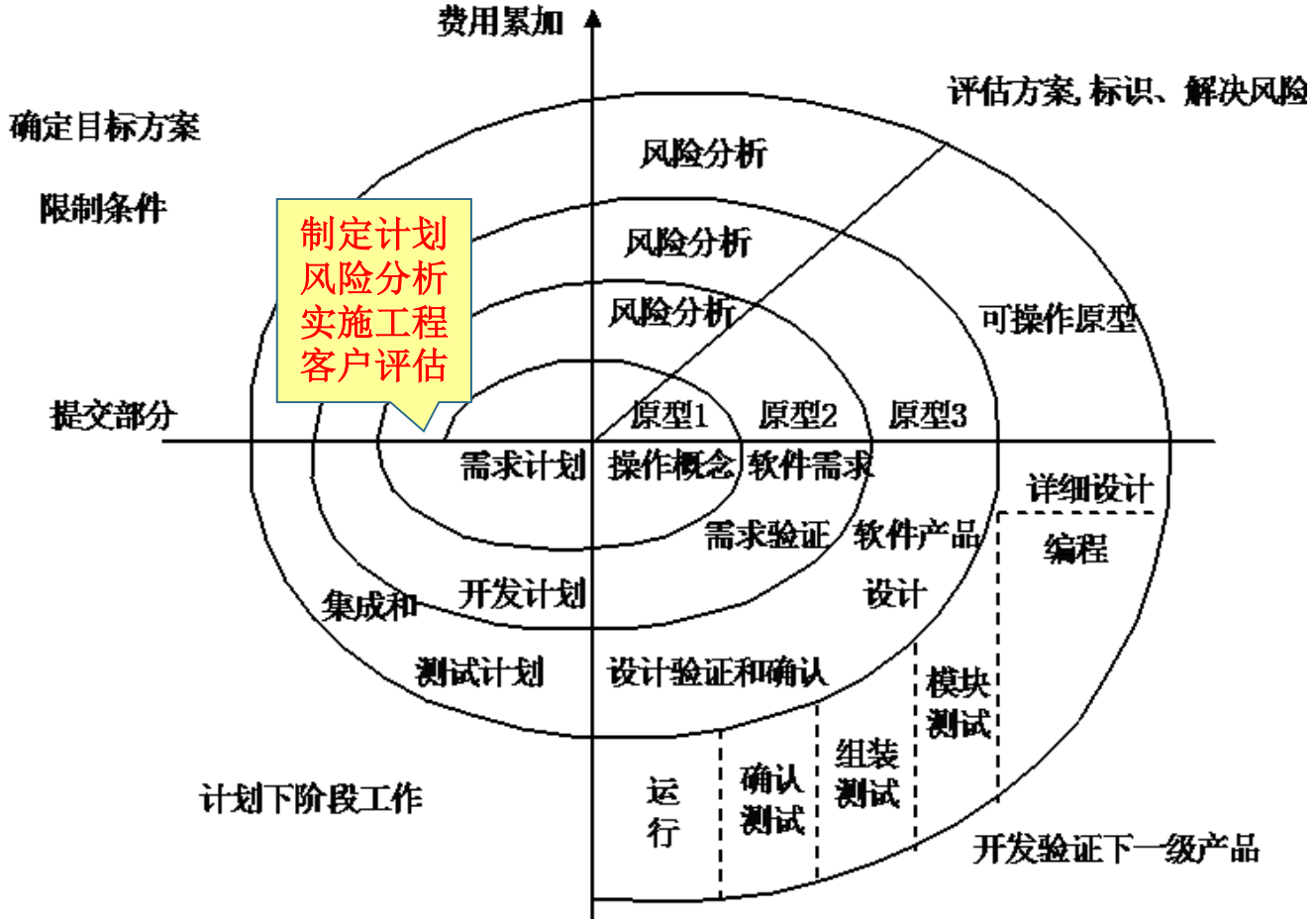


图1-10 螺旋模型

1.4 常用软件开发模型

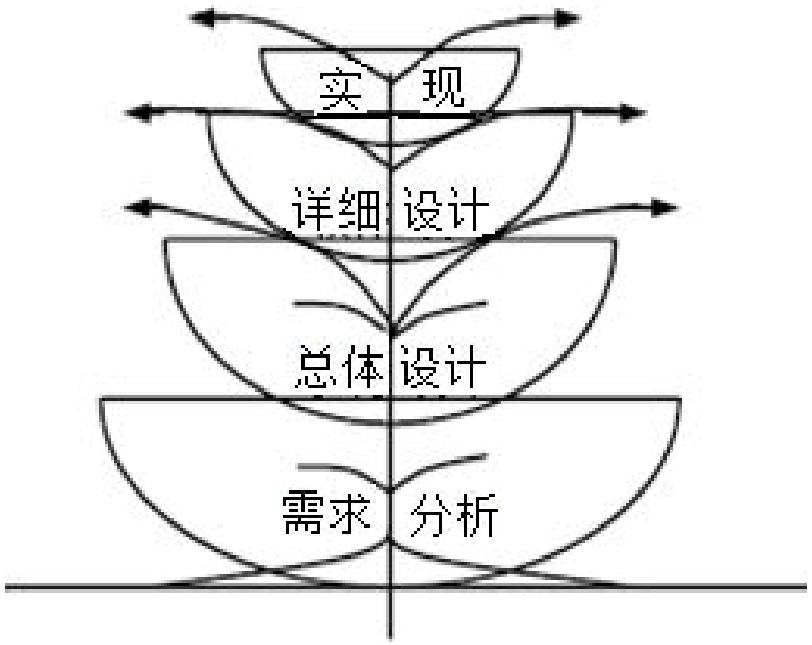
1.4.5 喷泉模型概述

喷泉模型主要适合于利用面向对象技术的软件开发项目。克服了瀑布模型不支持软件重用和多项开发活动集成的局限性。可使开发过程具有迭代性和无间隙性。

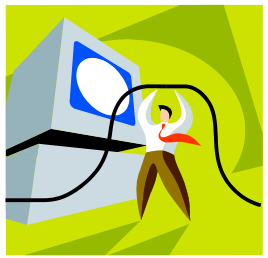
喷泉模型是以面向对象的开发方法为基础，以用户需求为源泉。从如图1-11的喷泉模型中可以看出7个特点：

（1）规定软件开发过程有4个阶段：需求分析、总体设计、详细设计和实现，还可分成多个开发步骤。

（2）各阶段相互重叠，反映了软件过程并行性的特点。



1-11 喷泉模型



1.4 常用软件开发模型

（3）以分析为基础，资源消耗成塔形，在分析阶段消耗的资源最多。

（4）反映软件过程迭代性的自然特性，从高层返回低层无资源消耗。

（5）强调增量开发，依据分析一点设计一点的原则，并不要求一个阶段的彻底完成，整个过程是一个迭代的逐步提炼的过程。

（6）是对象驱动过程，对象是活动作用的实体，也是项目管理的基本内容。

（7）实现中由于活动不同，可分为系统实现和对象实现，这既反映了全系统的开发过程，也反映了对象族的开发和重用过程。

1.4.6 基于面向对象的模型

面向对象技术优点很多应用非常广泛，构件重用就是其重要技术之一。强调了类的创建与封装，一个类创建与封装成功后，便可在不同的应用系统中被重用。面向对象技术为基于构件的软件过程模型提供了强大的技术框架。基于面向对象的模型，综合了面向对象和原型方法及重用技术。该模型如图1-12所示。

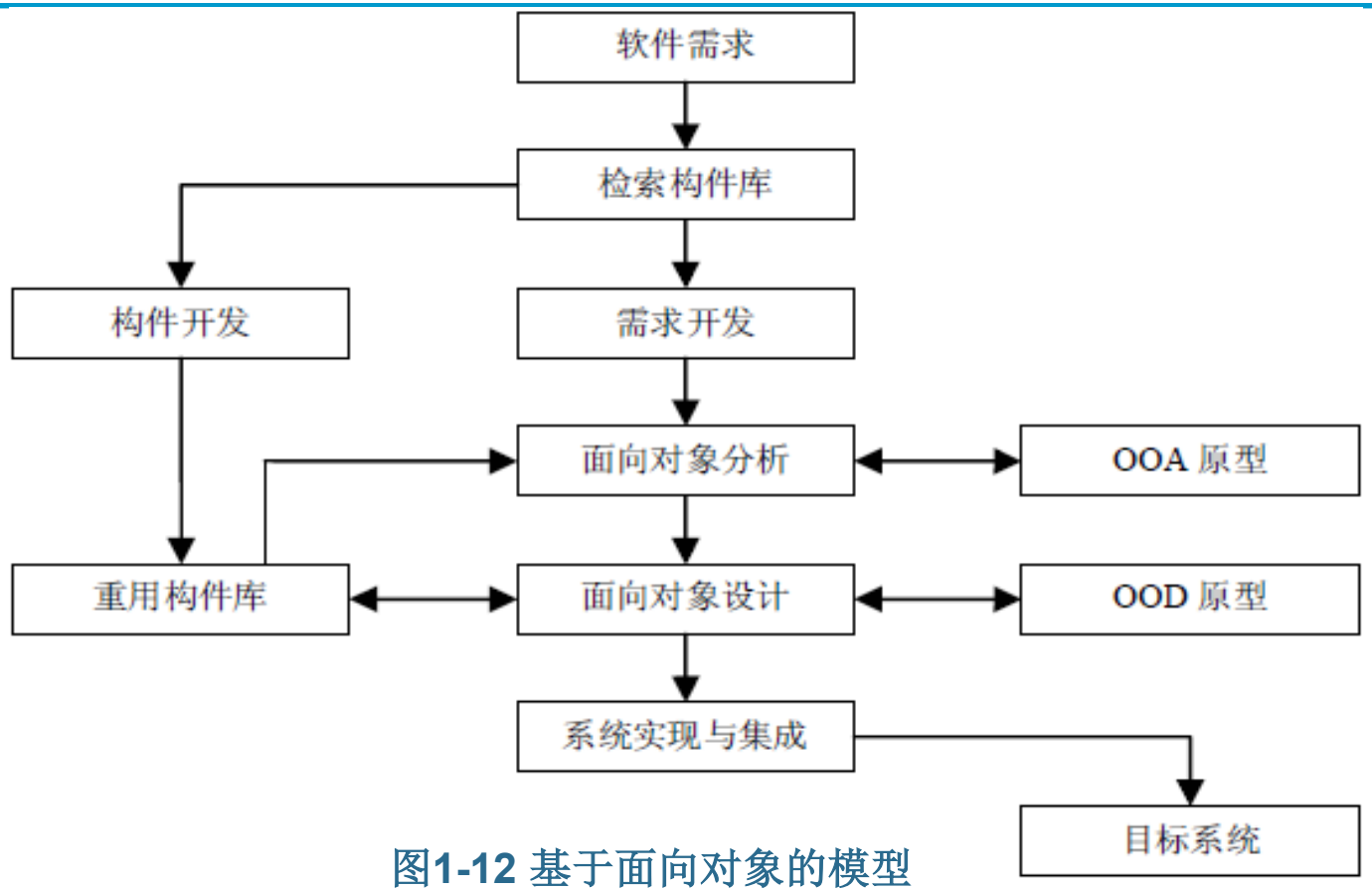


图1-12 基于面向对象的模型

1.4 常用软件开发模型

*其他软件开发模型

1. 智能模型

智能模型也称为基于知识的软件开发模型，是知识工程与软件工程在开发模型上的结合，以瀑布模型与专家系统的综合应用为基础。如图1-13所示可见与其他模型不同，其维护并不在程序一级上进行，可将问题的复杂性极大降低。

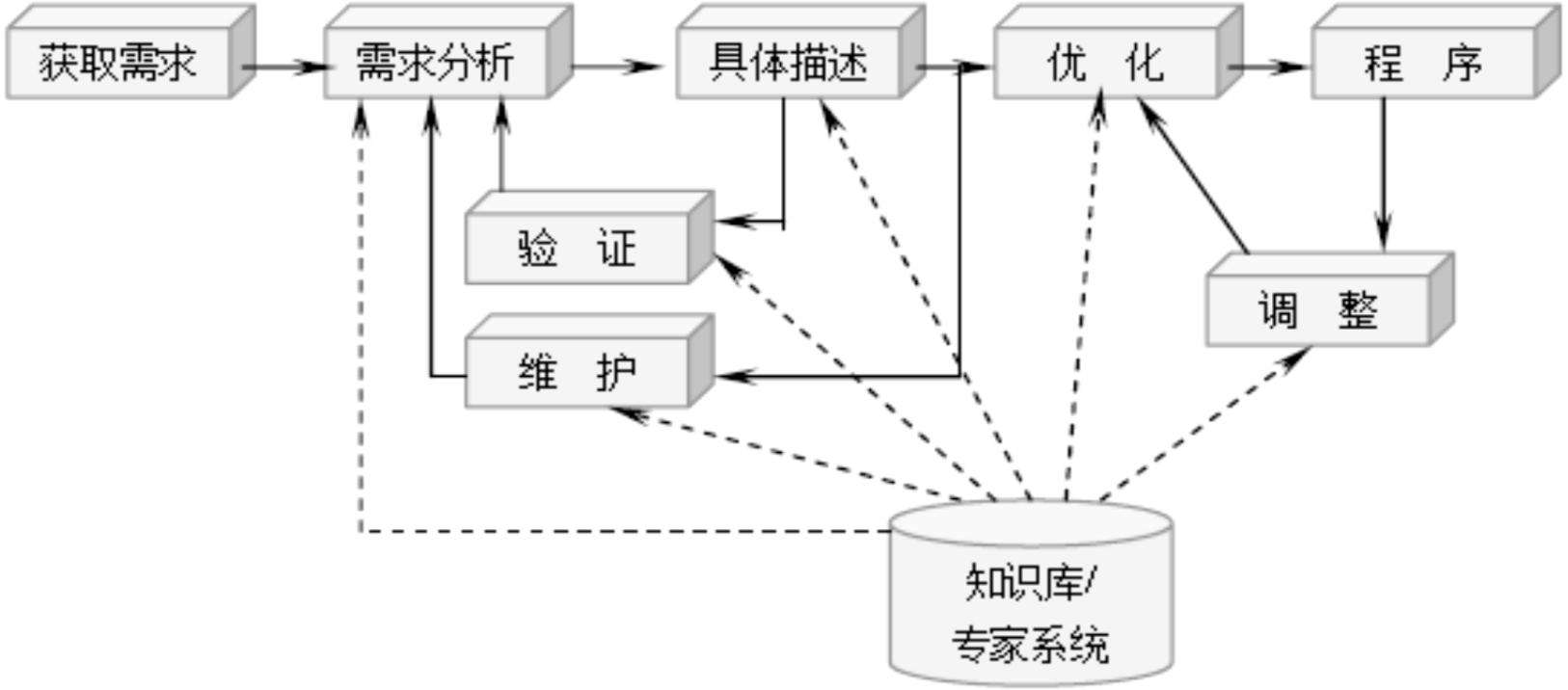


图1-13 智能模型

智能模型的主要优点为：

（1）利用领域专家系统，可使需求说明更完整、准确和无二义性。

（2）借助软件工程专家系统，提供一个设计库支持，在开发过程中成为设计者的助手。

（3）通过软件工程知识和特定应用领域的知识及规则的应用，对开发提供帮助。

1.4 常用软件开发模型

2. 统一过程模型

统一开发过程RUP(Rational Unified Process)模型在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发活动为开发成员提供了必要的准则、模板和工具指导,并确保共享相同的知识基础。建立了简洁和清晰的过程结构,为开发过程提供较多的通用性。

(1)RUP的二维开发模型及其核心工作流(Core Workflows)。主要包括：商业建模、需求、分析与设计、实现、测试、核心支持工作流、部署、配置和变更管理、项目管理和环境。如图1-14所示。

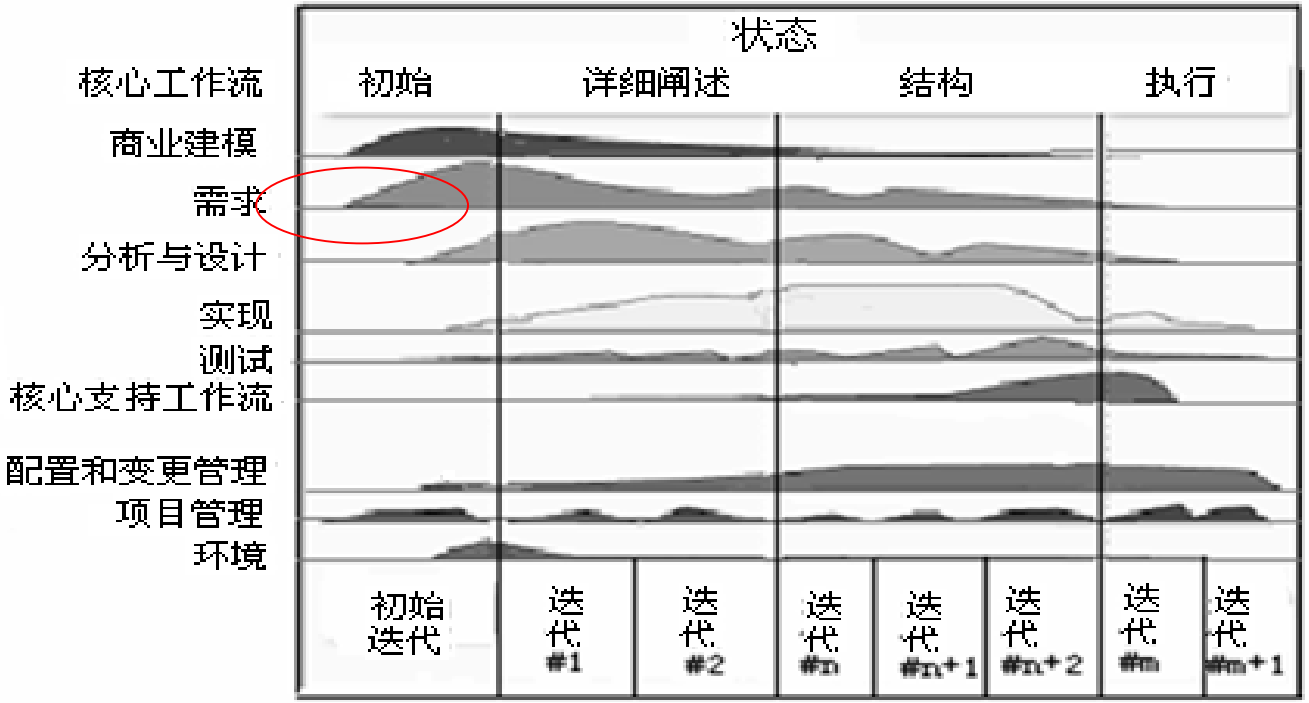


图1-14 RUP的二维开发模型及其核心工作流

(2)开发中各阶段和里程碑，主要包括。初始阶段、细化阶段、构造阶段、交付阶段。

(3)RUP的迭代开发模式。如图1-15所示。

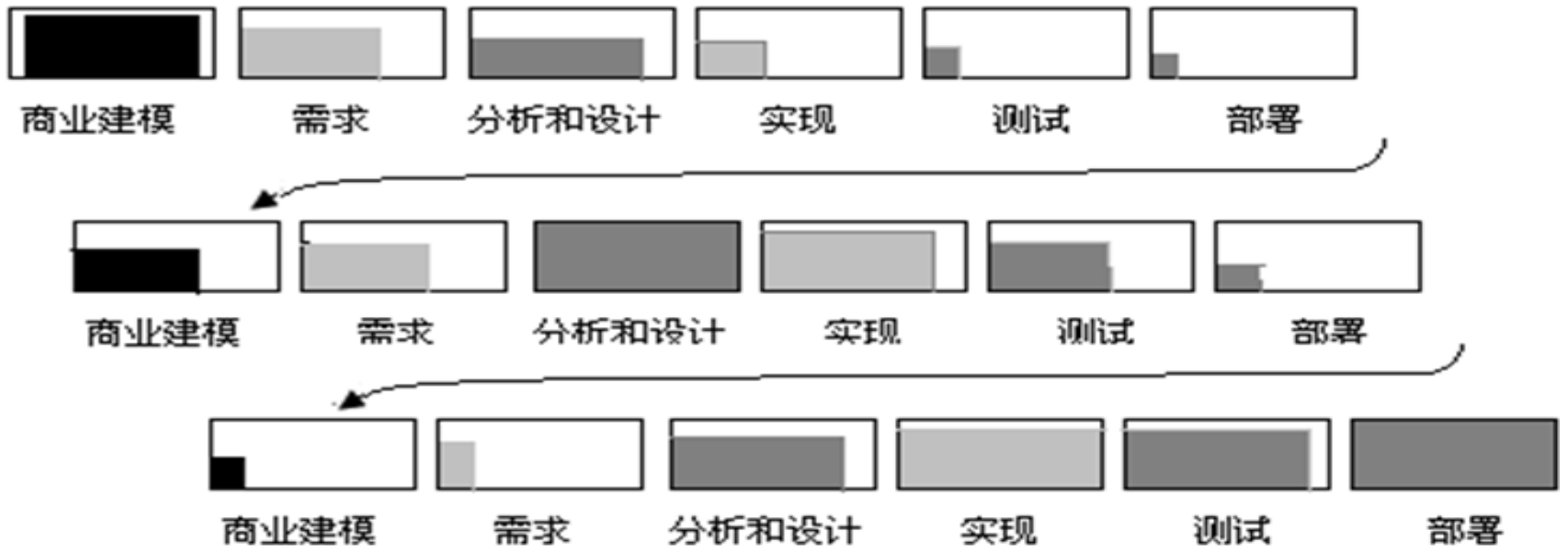


图1-15 RUP的迭代开发模式

1.4 常用软件开发模型

除了上述模型以外，还有其他一些的类似的模型，如形式化方法模型等。
新型技术模型，即第四代技术模型如图1-16所示。

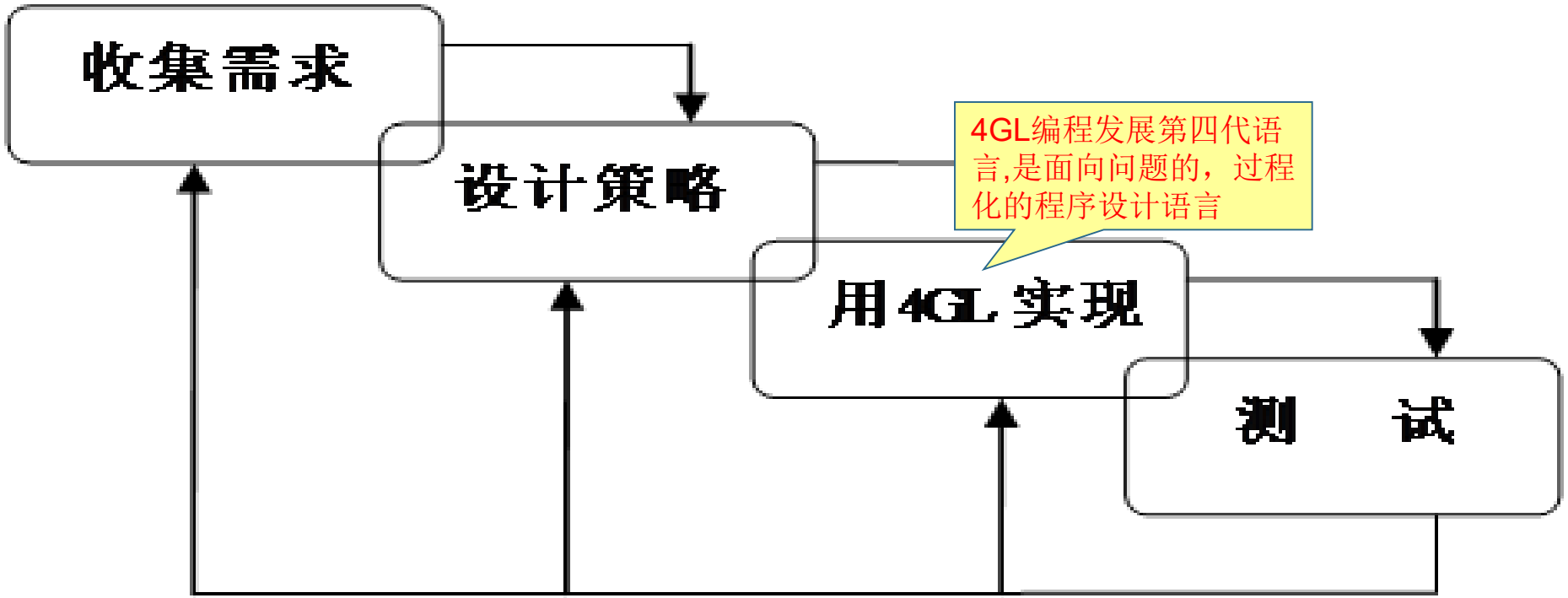
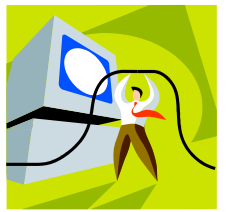


图1-16 第四代技术模型

1.4.7 软件开发模型的选定

1. 开发模型与开发方法及工具的关系

应用软件的开发过程主要包括:生存周期的系统规划、需求分析、软件设计、实现四个阶段。软件的开发方法多种多样，结构化方法和面向对象的方法是常用的最基本的开发方法。当采用不同的开发方法时，软件的生存周期过程将表现为不同的过程模型。为解决开发工程中大量复杂的手工劳动，提高软件的开发效率，还要采用计算机辅助软件工程CASE开发工具来支持整个开发过程。软件的开发模型（生存周期过程模型）与开发方法、开发工具之间的关系如图1-17所示。



1.4 常用软件开发模型

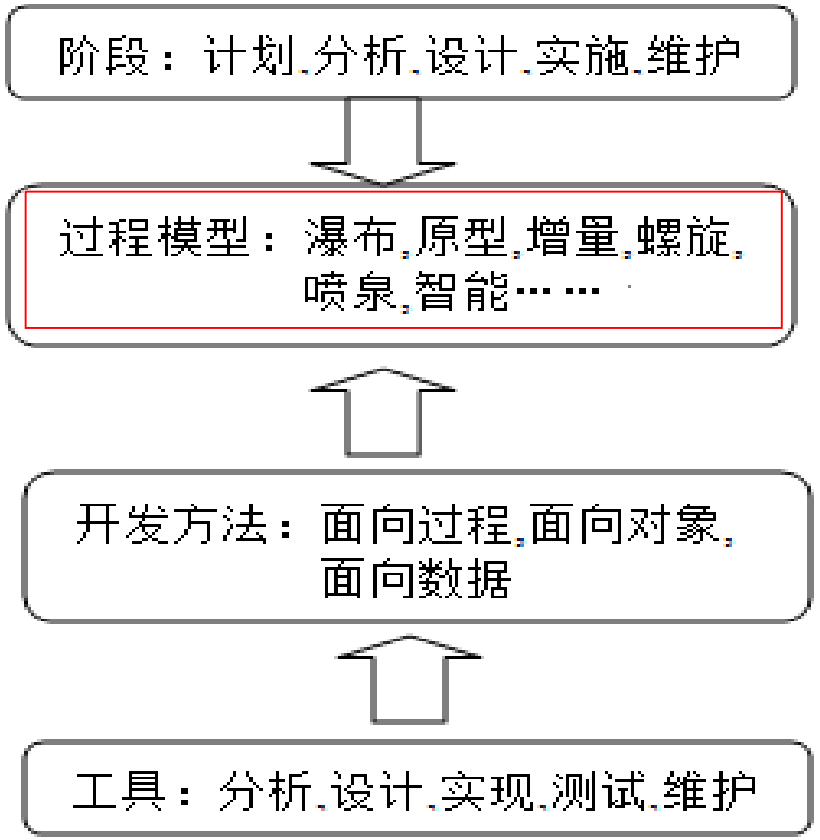


图1-17 开发模型、方法和工具之间的关系

2. 软件开发模型选取

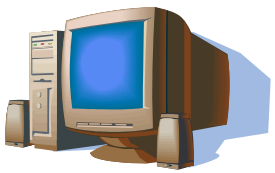
最常用的是瀑布模型和原型模型，其次是增量模型，由于迭代模型比较难以掌握使用较少。各种模型各有其特点和优缺点。在具体选择模型时需要综合考虑以下6点：

（1）符合软件本身的性质，包括规模、复杂性等；
（2）满足软件应用系统整体开发进度要求；
（3）尽可能控制并消除软件开发风险；
（4）具有计算机辅助工具快速的支持，如快速原型工具；
（5）与用户和软件开发人员的知识和技能匹配；
（6）有利于软件开发的管理与控制。

注意：通常情况下，面向过程方法可使用瀑布模型、增量模型和螺旋模型进行开发；面向对象方法可采用快速原型、增量模型、喷泉模型和统一过程进行开发；面向数据方法一股采用瀑布模型和增量模型进行开发。

3．软件开发模型的修定

在实际软件开发过程中，开发模型的选定并非直接照抄照搬、一成不变，有时还需要根据实际开发目标要求进行裁剪、修改、确定和综合运用。

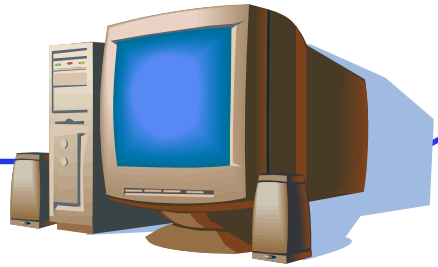


1.5 本章小结

软件是计算机程序及其有关的数据和文档的结合。软件危机是指在计算机软件开发和维护时所遇到的一系列问题。软件危机主要问题：一是如何开发软件以满足对软件日益增长的需求；二是如何维护数量不断增长的已有软件。

软件工程是软件开发、运行、维护和引退的系统方法。软件工程是指导计算机软件开发和维护的工程学科。软件工程采用工程的概念、原理、技术和方法来开发与维护软件。软件工程的目标是实现软件的优质高产。其主要内容是软件开发技术和软件工程管理。

软件开发方法学是编制软件的系统方法，它确定软件开发的各个阶段，规定每一阶段 的活动、产品、验收的步骤和完成准则。常用的软件开发方法有结构化方法、面向数据结构方法和面向对象方法等。



1.5 本章小结

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，规定了完成各项任务的工作步骤。ISO 9000把**软件过程定义**为：“把输入转化为输出的一组彼此相关的资源和活动”。软件过程定义了运用方法的顺序、应该交付的文档、开发软件的管理措施、各阶段任务完成的标志。软件过程必须科学、合理，才能获得高质量的软件产品。

软件产品从问题定义开始，经过开发、使用和维护，直到最后被淘汰的整个过程称为软件生存周期。根据软件开发工程化的需要，生存周期的划分有所不同，从而形成了不同的软件生存周期模型，或称软件开发模型。**软件开发模型**包括：瀑布模型、快速原型模型、增量模型、喷泉模型、螺旋模型、智能模型、构件组装模型、统一过程模型等。

软件开发时可把各种模型的特点结合起来，充分利用优点、减少缺点。软件开发的各个阶段必须完成的各种规格书、说明书、用户手册等文档。