目录：  
1.1 软件的概念 P2 2.1 软件工程发展史 P3-4  
1.2 软件产品的概念 P2 2.2 软件危机 P4  
1.3 软件产品的类型 P2 2.2.1 软件危机概述 P4  
1.3 应用软件的类型 P2-3 2.2.2 软件危机产生原因 P4  
 2.2.3 软件危机主要表现形式 P4

3.1软件工程标准化的概念 P7 2.2.4 软件危机的解决途径 P4  
3.2 软件工程标准化的意义 P7 2.3 软件工程的概念 P4-5  
3.3 软件工程标准 P7-8 2.4 软件工程学的内容、结构化层次和基本化概念 P5  
3.3.1 ISO 9000国际标准 P8 2.4.1软件工程过程 P6

3.3.2 中国的主要软件工程标准 P8 2.4.2软件开发的基本策略 P6

5.1 软件生存周期 P9 2.4.3软件工程的技术 P6

5.1.1 软件生存周期的概念 P9 2.4.4软件工程方法 P6-7

5.1.2 软件生存周期的任务 – 软件计划时期 P9-10 2.4.5软件工程工具 P7

5.2 软件过程 P10

5.2.1 软件过程的定义 P10 4.1 软件开发方法学 P8

5.2.2 软件过程的分类：项目管理过程、项目研发过程、项目支撑过程 P10

4.2 软件工具 P8-9

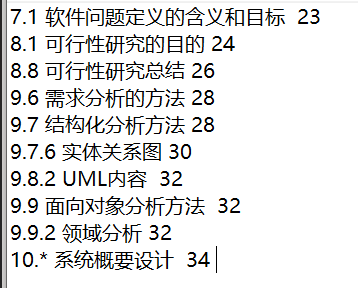
5.2.3软件过程举例 P11

5.3 软件过程模型 P11 6.1 软件过程改进的概念&目标意义 P20

5.3.1瀑布模型 P11-12 6.2 软件过程成熟度的概念 P20

5.3.2 快速原型模型 P12-13 6.3 不成熟过程与成熟过程的对比 P21-22

5.3.2.1 快速原型实现途径 P13-14 6.4 过程认同和过程制度化 P22

5.3.3 增量模型 P14

5.3.3.1 增量模型优缺点 P14

5.3.3.2 增量模型的增量方式 P14

5.3.3.3 增量模型示例 P14

5.3.3.4 增量模型示例 P15

5.3.4 螺旋模型 P15

5.3.4.1 螺旋模型六大步骤 P16

5.3.5 喷泉模型 P17

5.3.5.1 喷泉模型六大步骤 P18

5.3.6 迭代模型 P18-19

5.4 两个典型的软件过程模型 P19

5.4.1 统一过程RUP：Rational Unified Process P19

5.4.2 敏捷开发过程Agile P20

1.1 **软件的概念** **程序 ≠ 软件软件是指能够实现预定功能和性能的可执行的计算机程序，使程序正常运行所需要的数据，描述软件开发及其管理、程序的操作和使用的有关文档的完整集合。**

**计算机程序是能够完成预定功能和性能的可执行的指令序列；**

**数据是程序能适当处理的信息，具有适当的数据结构，并且可以储存；**

**软件文档（Software Documentation）是开发、使用和维护程序所需要的图文资料。**

1.2 **软件产品的概念**

软件产品：是指向用户提供的计算机软件、信息系统或设备中嵌入的软件，或在提供计算机信息系统集成、应用服务等技术服务时提供的计算机软件。它是通过人的脑力劳动，通过知识、技术、环境信息的高度整合，所产生的逻辑产品。

1.软件产品**以程序和文档的形式**保存在作为计算机存储器的磁盘和光盘介质上，软件的功能和作用**必须通过**操作计算机才能体现，**不能离开**硬件和支撑环境而单独存在。

2.软件产品的**生产主要是研制**。软件产品的**成本主要体现在软件的开发和研制**上，软件开发研制完成后，通过复制就可以产生大量的软件产品，不需要再花费人力和物力。

3.软件产品不会用坏，**不存在磨损、消耗**等问题。但是随着时间的推移，软件产品会由于不适应新的软件生态环境而**相对退化**。

4. 软件产品在测试阶段难以发现的**隐藏错误和缺陷**，会在投入使用后不断暴露，因此需要不断地维护，“补丁”越来越多，**维护成本**也就越来越高。

5. 软件产品的生产主要是靠脑力劳动，还**未完全摆脱手工开发方式**，大部分产品是“**定做的**”。在软件产品的开发过程中，由于软件开发者与用户的知识领域不同，需要**大量的交流与沟通**，并不断地**修改需求**，这会使软件开发费用不断增加。

**1.3 软件产品的类型**

**按软件使用环境划分** 1：嵌入式软件2：桌面级（PC）软件3：大型（巨型）机软件4：云计算软件

**按软件服务对象的范围划分**1：个人软件2：商业通用软件3：工程及科学应用软件4：企业定制软件5：人工智能软件

**按软件规模划分** 微型 参加人员数 1，研制期限 1 ～ 4周。小型 参加人员数 2 ～ 5 ，研制期限 1 ～ 12月。 中性 参加人员数 5 ～ 20 ，研制期限 1 ～ 2年。

大型 参加人员数＞ 20，研制期限＞ 2年。

**从软件工程学角度划分**

基础软件 保证计算机运行并支撑应用软件运行的基础环境，包括：操作系统、数据库、中间件。

应用软件 为满足用户不同领域、不同问题的应用需求而提供的那部分软件

支撑软件 支撑各种软件的开发与维护的软件，又称集成开发环境（Integrated Development Environment，IDE

**应用软件：桌面通用（个人）软件 企业级软件 嵌入式软件 云计算软件**

桌面通用（个人）软件是指安装在PC上可以独立运行的软件，包括：

办公室软件（如：Office，WPS）、

互联网软件（如：QQ，Foxmail，Google Chrome）、

多媒体软件（如：Photoshop，AutoCAD，Nero）等。

**企业级软件**

是指为商业组织、大型企业、行政事业部门**等特定领域创建并部署的软件**。

随着信息化时代的到来以及经济全球化的不断深入，企业级软件已经发生了从部门独立应用到**企业集成**、从少数人使用到面向**广大客户**、从企业内部使用到**企业间协作**的转变。

企业级软件具有**资源庞杂、事务密集、数据量大、用户众多、并发操作**等特点，构成了结构复杂、操作便捷、网际协调与分布、高安全需求的**应用集群**。例如，企业资源规划（Enterprise Resource Planning，ERP）、客户关系管理（Customer Relationship Management，CRM）、后台管理（Back Office）、供应链管理（Special Chemical Materials，SCM）、产品生存周期管理（Product Life-Cycle Management，PLM）等软件都属于企业级软件，此外还有会计类、电子商务类等专业软件。

**嵌入式软件**

是**基于嵌入式系统设计的软件**，它也是计算机软件的一种，同样由**程序**及其**文档**组成，也有自己的**基础软件**（嵌入式系统，如手机上的Android系统）、**支撑软件**（开发工具包、应用程序编程接口〔Application Programming Interface，API〕调用等）和**应用软件**。每种嵌入式软件都有自己**独特的应用环境和实用价值**。**Android**（以Linux为基础的半开源操作系统，主要用于移动设备）、**iOS**（iPhone Operating System，苹果公司为iPhone开发的操作系统）是最常见的**嵌入式系统**，在这些系统平台上运行的软件产品实质上都是**嵌入式系统的应用软件**。

**云计算软件**

云计算是一种**按使用量付费的模式**，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的**计算资源共享池**（资源包括网络、服务器、存储、应用软件和服务），这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。

云计算的**基本原理**：通过**使计算分布在大量的分布式计算机上**，而非本地计算机或远程服务器中，企业数据中心的运行将与互联网更相似。这使得企业能够将资源切换到需要的应用上，**根据需求访问计算机和存储系统**。

云计算的**应用方式**可分为基础设施即服务（Infrastructure as a Service，**IaaS**）、平台即服务（Platform as a Service，**PaaS**）、软件即服务（Software as a Service，**SaaS**）、数据即服务（Data as a Service，**DaaS**）等。

**2.1 软件工程发展史**

20世纪60年代中期以前 **程序设计阶段**。早期的程序规模小，**程序往往是个人设计、自己使用**。在进行程序设计时，通常要**注意**如何节省存储单元、提高运算速度。除了**程序清单**之外，没有其他任何文档资料。

20世纪60年代中期～70年代中期 **“软件=程序+文档”阶段。**人们把软件视为**产品**，确定了软件生产的**各个阶段**必须完成的，有关计算机程序的功能、设计和使用的文字或图形资料，并把这些资料称为**文档**。而由于忽略软件维护问题，**软件危机**由此产生。

20世纪70年代中期～80年代中期 **软件工程阶段**。**软件工程学**运用工程学的基本原理和方法来组织和管理软件生产，以**保证软件产品的质量**和**提高软件产品生产率**。软件开发开始采用工程化开发**方法**、**标准**和**规范**，以及**面向对象技术**等。

20世纪80年代中期至今 **第四代技术阶段**。**面向对象技术**已在许多领域迅速取代了传统的软件开发方法，**并行计算、网络计算机、多媒体技术、云计算技术和现代通信技术**使人们开始采用和原来完全不同的方法进行工作。

**2.2 软件危机**

**2.2.1 软件危机概述**

**软件危机是指在计算机软件开发和维护过程中碰到问题的集合。**

**如何开发软件以满足社会对软件日益增长的需求？**

**如何维护数量不断增长的已有软件？**

**2.2.2 软件危机产生的原因**

1：软件是计算机系统中的**逻辑部件**，软件产品往往**规模庞大**，给软件的开发和维护带来客观的困难。

2：软件一般要使用5～10年，在这段时间里，很可能出现开发时没有预料到的问题。例如，当系统运行的硬件、软件环境发生了变化时，系统需求发生了变化时，需要及时地维护软件，使软件可以继续使用。

3：软件开发技术落后，生产方式和开发工具落后。

4:软件开发人员忽视软件需求分析的重要性，对软件可维护性不重视，也是造成软件危机的原因之一。

**2.2.3 软件危机的主要表现形式**

**1：软件的发展速度跟不上硬件的发展和用户的需求。**

**2：软件的成本和开发进度难以预测和控制，用户不满意。**

**3：软件产品质量差，可靠性不能保证。**

**4：软件产品可维护性差。**

**5: 软件没有合适的文档资料。**

**2.2.4 解决软件危机的途径**

**1: 使用好的软件开发技术和方法。**

**2: 开发软件时有良好的组织、严密的管理，各方面人员相互配合共同完成任务。**

**3: 使用好的软件开发工具，提高软件生产率。**

**2.3 软件工程**

什么是软件工程？软件工程（又称软件工程学）是一门旨在生产无故障的、及时交付的、在预算之内的，满足用户需求的软件的学科。实质上，软件工程就是采用工程的概念、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方法和最先进的软件开发技术结合，应用到软件开发、维护过程中。

软件工程所实现的内容，不是为目标用户开展业务而提供使用的工具产品，而是指导软件设计、开发人员进行项目实施的思想、方法和工具。

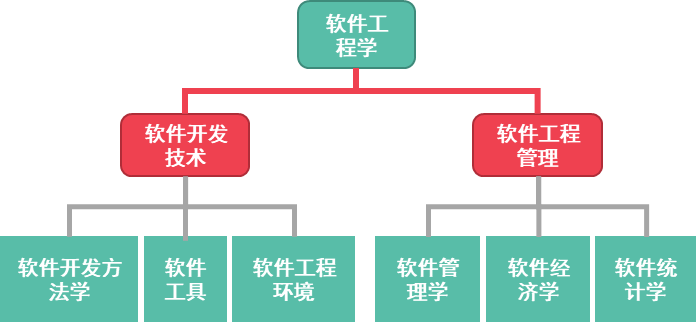
**软件工程是应用计算机科学理论和技术以及工程管理原则和方法，按预算和进度，实现满足用户要求的软件产品的定义、开发、发布和维护的工程或进行研究的学科。**

由于软件危机的发生，计算机科学家们开始研究解决软件危机的方法。逐渐形成了计算机科学技术领域中的一门新兴学科--**软件工程学**。**1993年，IEEE为软件工程下的定义：将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护过程，即将工程化应用于软件中的方法的研究。**

**2.4 软件工程学的内容**

**各项内容包含的具体信息**

**软件开发的目标是：优质高产，就应该从技术到管理都规约出相应的管理办法，在这个过程中就逐步形成了“软件工程学” 学科，其主要内容有：**

****

**2.4 软件工程的层次化结构**

软件工程的整体框架，可分为以下四个层次：

**质量保证层：质量保证层**是推动软件过程不断改进的动力，即全面的**质量管理**和**质量需求**，正是这种动力推动了软件工程方法向更加成熟的方向前进。

**过程、技术层：过程、技术层**定义了一组**关键过程与框架**，其目的是保证软件工程技术被有效地应用，使得软件能够被及时、高质量和科学合理地开发出来。

**方法层：方法层**提供了**软件开发的各种方法**，包括进行软件需求分析和设计、软件实现设计、测试和维护等。

**工具层：**工具层是指除了软件实现所应用的**语言开发环境之外**，为软件工程方法和过程提供的自动或半自动化的**支撑环境**。

**2.4 软件工程学的内容**

1.用分阶段的**生存周期计划**进行严格的管理**。**

**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.软件工程**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。

**一种层次化技术 Tools 工具 Methods 方法 Process model 过程 “quality” focus 质量关注点 Software Engineering 软件工程**

**2.4 软件工程的基本概念**

**2.4.1软件工程过程：**

问题定义→ 需求开发→ 软件设计→ 软件构造→ 软件测试

构想文档 分析模型 设计模型 源程序 测试规程

用户故事 软件需求规格说明 软件体系结构文档 目标代码 测试用例

软件详细设计文档 可执行构件 测试报告

**2.4.2软件开发的基本策略：**

**软件复用**：1.构建一个新的系统不必从零做起，直接复用已有的结构件进行组装。2.构件式经过反复使用验证的，由其组成的新系统具有较高的质量。

**分而治之**：1.将一个复杂的问题分解成若干个简单的问题，然后逐个解决。

2.来源于人们生活与工作的经验，完全适合技术领域。

**逐步演进**：1.软件开发是自底向上有序的生长过程。2.小步快跑：每做完一步在调成并为下一步确定方向，直到终点。

**优化折中：**1.优化：优化软件的各个质量特性。如运行速度、资源利用、用户体验。

2.折中:通过协调各个质量特性，实现整体质量的最优。

2.4.3 软件工程的技术

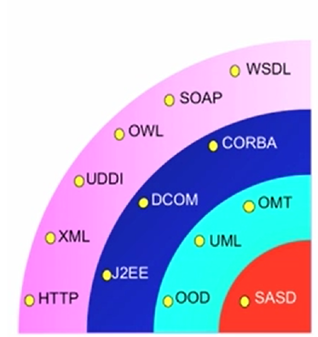
**抽象← →软件过程**

**软件建模方法← →软件复用**

**用户见面原型化← 软件工程的Wasserman规范 →度量**

**软件体系结构← →工具与集成环境**

2.4.4 软件工程方法



软件的复用程度逐渐提升，开发效率越来越高

**面向服务**：在应用表现层次上将软件构件化。即应用业务过程由服务组成，而服务由构件组装而成。

**面向构件**：需求比类的粒度更大的且易于复用的构件，期望实现软件的再工程。

面对对象：以类为基本程序单元，对象是类的实例化，对象之间以消息传递为基本手段。

**面向过程**：以算法作为基本构造单元，强调自顶向下的功能分解，将功能和数据进行一定程度的分离。

2.4.5 软件工程工具

需求开发 软件设计 软件构造 软件测试 软件维护 开发管理

软件构建工具 程序编辑器 单元测试工具 代码重构工具 需求管理工具

数据库设计工具 程序编译器 静态分析工具 逆向工程工具 项目管理工具

程序调试器 自动化测试工具 配置管理工具

集成开发环境 性能测试化工具 测试管理工具



**为软件开发方法提供了自动的或者半自动开发环境。**

**3.1 软件工程标准化的概念**

**软件工程的范围**从当初只是使用程序设计语言编写程序，扩展到**整个软件生存周期**，同时还有许多**技术管理工作**（如过程管理、产品管理和资源管理）以及**确认与验证工作**（如评审和审计、产品分析和测试等），这些都已成为软件工程标准化的内容。

**3.2 软件工程标准化的意义**

1. 减少开发人员间联系的差错和误解，提高软件的生产效率，缩短软件开发周期。
2. 有利于软件管理，降低开发和运行成本
3. 提高软件产品的质量（可靠性、可维护性和可移植性）
4. 可提高软件人员的技术水平

**3.3 软件工程标准**

3.3.1 ISO 9000国际标准

ISO 9000由5个相关标准组成，它们分别是：

（1）ISO 9000质量管理和质量保证标准：选择和使用的**导则**。

（2）ISO 9001质量体系：**设计/开发、生产、安装和服务**中的质量保证模式。

（3）ISO 9002质量体系：**生产和安装**中的质量保证模式。

（4）ISO 9003质量体系：**最终检验和测试**中的质量保证模式。

（5）ISO 9004质量管理和质量体系要素：**导则**。

3.3.2 中国的主要软件工程标准

分类 标准名称 标准号

**基础标准** 软件工程术语 GB/T 11457-1995

信息处理---数据流程图、程序流程图、系统流程 图、程序网络图和系统资源图的文件编辑符号，以及约定软件工程标准分类法 GB/T 15538-1995

信息处理---程序构造及其表示法的约定 GB 13502-1992

信息处理---判定表规范 GB/T 15535-1995

计算机系统配置图符号及其约定 GB/T 14085-1993

**开发标准** 软件开发规范 GB 8566-1988

计算机软件单元测试 GB/T 15532-1995

软件支持环境 GB/T 15853-1995

软件维护指南 GB/T 14079-1993

**文档标准** 软件文档管理指南 GB/T 16680-1996

计算机软件产品开发文件编制指南 GB 8567-1988

计算机软件需求说明编制指南 GB 9385-1988

计算机软件测试文件编制规范 GB 9386-1988

**管理标准** 计算机软件配置管理计划规范 GB/T 12505-1990

计算机软件可靠性和可维护性管理 GB/T 14394-1993

计算机软件质量保证计划规范 GB/T 12504-1990

**4.1 软件开发方法学**

**软件开发方法学**（SoftWare Methodology，**软件方法学**）是以**方法**为研究对象的软件学科。**主要涉及**指导软件设计的**原理和原则**，以及基于这些原理、原则的**方法**和**技术**。狭义的也指某种特定的软件设计指导原则和方法体系。不论何种含义，其关注的中心问题是**如何设计正确的软件和高效率地设计软件**。

软件方法学的**目的**是寻求科学方法的指导，使软件开发过程“**纪律化**”，即要寻找一些规范的“求解过程”，把软件开发活动置于坚实的理论基础之上。

软件开发方法学**包括**：

* **软件开发过程**：规定了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。
* **完成软件开发任务的技术方法**：结构化开发方法、面向对象开发方法、划分阶段、组件构件、软件复用技术等。

**4.2 软件工具**

**软件工具（Software Tools）是指为了支持计算机软件的开发和维护而研制的程序系统**。

使用软件工具的目的是提高软件设计的质量和生产效率，降低软件开发和维护的成本。

**软件工具可用于软件开发的整个过程。**软件开发人员在软件生产的各个阶段可根据不同的需要选用合适的工具。例如：

* **需求分析**工具用类生成需求说明；
* **设计阶段**需要使用编辑程序、编译程序、连接程序，有的软件能自动生成程序等；
* 在**测试阶段**可使用排错程序、跟踪程序、静态分析工具和监视工具等；
* **软件维护阶段**有版本管理、文档分析工具等；
* **软件管理**方面也有许多软件工具。

**5.1 软件生存周期**

**5.1.1 软件生存周期的概念**

**软件生命周期Software Life Cycle**

软件产品或软件系统**从产生直到淘汰**的全过程

**1.软件计划时期** 软件计划时期分为**问题定义**、**可行性研究和需求分析**3个阶段。

**2.软件开发时期** 软件开发时期可分为**软件设计**、**软件实现**和**综合测试**阶段。其中，

**软件设计阶段**：概要设计和软件详细设计阶段；

**软件实现阶段**：程序设计和软件单元测试；

**3.软件交付** 软件交付之前要进行**综合测试**等。

软件**交付**使用后，在软件运行过程中，需要不断地进行**维护**，才能使软件持久地满足用户的需要。

**5.1.2 软件生存周期的任务 – 软件计划时期**

**1.问题的定义**

**确定系统的目标、规模和基本任务。《项目计划报告》**

**2.可行性研究**

**从经济、技术、法律等方面分析确定系统是否值得开发，及时建议停止不值得开发的项目，避免人力、物力和时间的浪费。《可行性研究报告》**

**3.需求分析**

**确定软件系统应具备的具体功能。通常用数据流图、数据字典和简明算法描述表示系统的逻辑模型，以防止产生系统设计与用户的实际需求不相符的后果。《需求规格说明书》**

**4.概要设计**

**确定系统设计方案、软件的体系结构。确定软件由哪些模块组成以及这些模块之间的相互关系。《总体设计说明书》**

**5.详细设计**

**描述应该如何具体地实现系统。详细设计的每个模块应确定实现模块所需要的算法和数据结构。《详细设计说明书》**

**6.软件实现阶段**

**进行程序设计（编码）和模块测试。《源程序》《编码规范》**

**7.综合测试阶段**

**通过各种类型的测试，查出软件设计中的错误并改正，确保软件质量；**

**还要在用户的参与下进行验收，才可交付使用。《软件测试报告》**

**8.软件运行和维护**

**在软件运行期间，通过各种必要的维护，使系统改正错误或修改扩充功能以适应环境变化，从而延长软件的使用寿命，提高软件的效益。每次软件维护的要求及修改过程都应详细准确地记录下来，并作为文档保存。《软件维护说明》**

**5.2 软件过程**

**5.2.1 软件过程的定义**

**国际标准化组织定义：**把输入转化为输出的一组彼此相互关联的资源和活动。

**《信息技术 软件工程术语》定义：**软件过程是由组织或项目使用的，用以计划、管理、执行、监控、控制和改进其软件相关活动的过程或过程的集合。

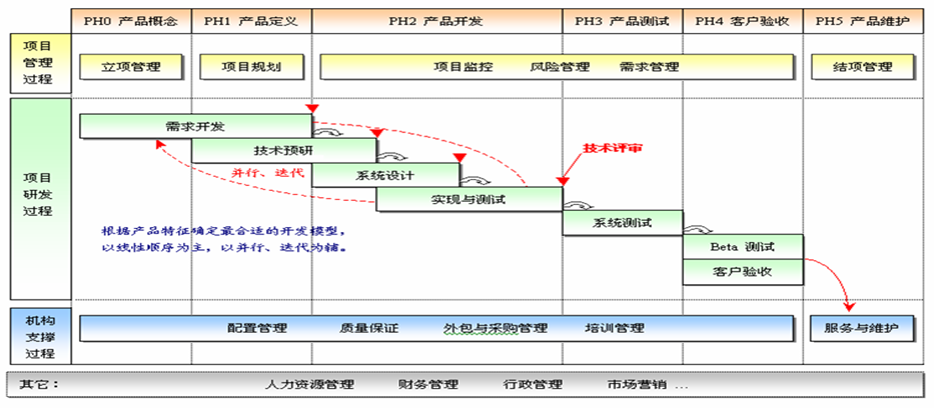
**软件过程** (software process) 或 **软件开发过程** (software development process)，是软件开发的**开发生命周期** (software development life cycle)，其各个阶段实现了软件的**需求定义与分析**、**设计**、**实现**、**测试**、**交付**和**维护**。

软件过程是在开发与构建系统时应遵循的步骤，是**软件开发的路线图**。

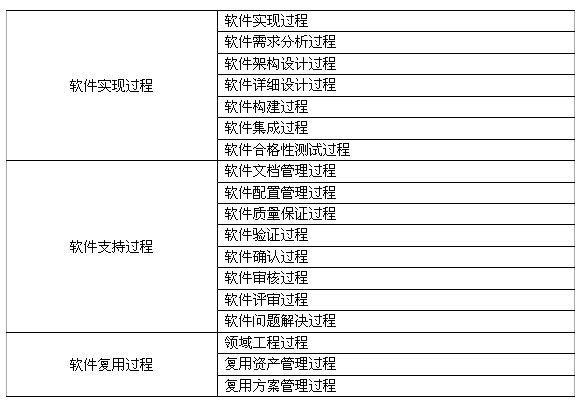
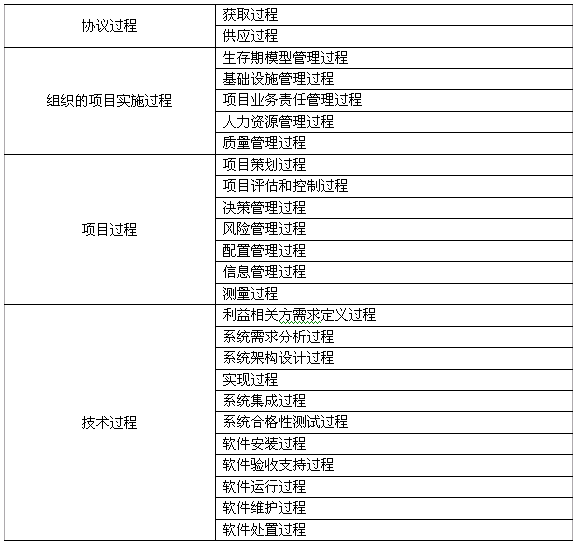
软件过程是为了获得高质量软件所需要完成的**一系列任务的框架**，它规定了完成各项任务的**工作步骤**；定义了运用方法的**顺序**、应该交付的**文档**、开发软件的**管理措施**和各阶段任务**完成的标志**。

在近十几年的大量研究和实践之后，许多人逐渐认识到，从**软件过程的改进**来解决可能是有效的方法之一。

5.2.2 软件过程的分类：项目管理过程、项目研发过程、项目支撑过程



5.2.3软件过程举例



5.3 软件过程模型

**软件过程**是在工作产品构建过程中，所需完成的工作**活动**、**动作**和**任务的集合**。

**活动**主要实现宽泛的目标，与应用领 域、项目大小、结果复杂性或者实施软件工程的重要程度没有直接关系。

**动作**包含了主要工作产品生产过程中的一系列任务。

**任务**关注小而明确的目标，能够产生实际产品

**软件过程模型**也常称为软件开发模型、软件生存周期模型、软件工程范型

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。

5.3.1瀑布模型

**瀑布模型把软件生存周期划分为**计划时期**、**开发时期**和**运行维护时期**。这些时期又可细分为若干阶段：**

计划时期**可分为问题定义、可行性研究和需求分析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；**

运行维护时期**则需要不断进行运行维护，以延长软件的使用寿命。**

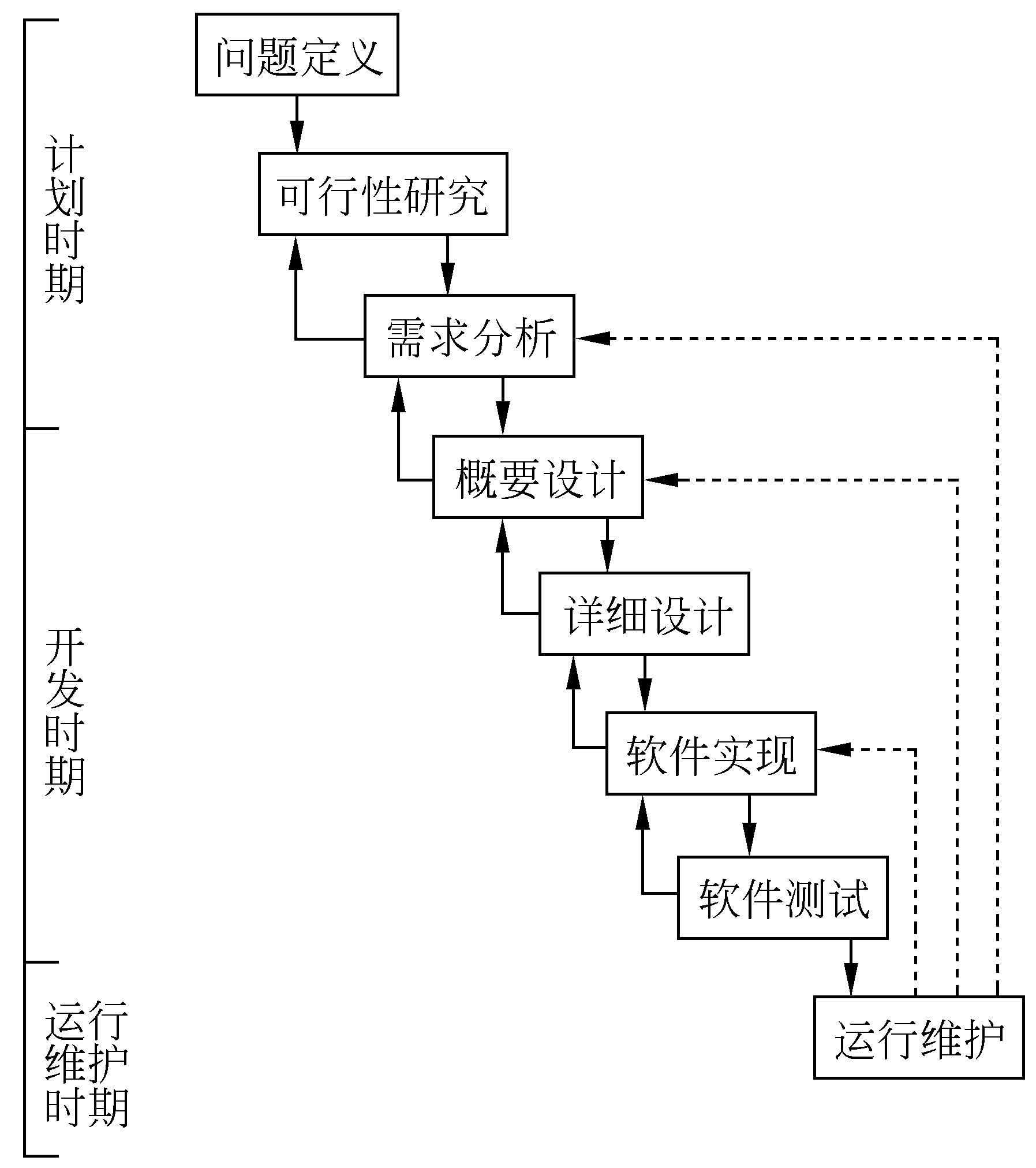
**瀑布模型：预见性开发、线性模型、文档驱动、过程控制为核心。**

**瀑布模型要求：**

**开发过程的**每个阶段结束时要进行复审**，复审通过了才能进入下一阶段，**复审通不过**要进行修改或回到前面的阶段进行返工。**

**软件维护时可能需要修改错误和排除故障；也可能是用户的需求改变了，或软件的运行环境改变了，需要修改软件的结构或功能，因而**维护工作可能要从修改需求分析或修改概要设计开始**，也可能从**修改软件编码**开始。**

**图中，**实线箭头**表示开发工作的流程，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.软件生存周期的顺序性**

**顺序性**是指只有前一阶段工作完成以后，后一阶段的工作才能开始；前一阶段的输出文档，就是后一阶段的输入文档。因而，瀑布模型的特点是由**文档驱动**。如果在生存周期的某一阶段出现了错误，往往要追溯到在它之前的一些阶段。

**2.尽可能推迟软件的编码**

实践表明，大、中型软件的编码阶段开始得越早，完成所需要的时间反而越长。

瀑布模型在编码之前安排了需求分析、概要设计、详细设计等阶段，从而把**逻辑设计**和**编码**清楚地划分开来，**尽可能推迟程序编码阶段**，以保证软件开发质量。

**3.保证质量**为了保证质量，瀑布模型**坚持两个重要做法**：

① 每个阶段都要完成规定的**文档**；

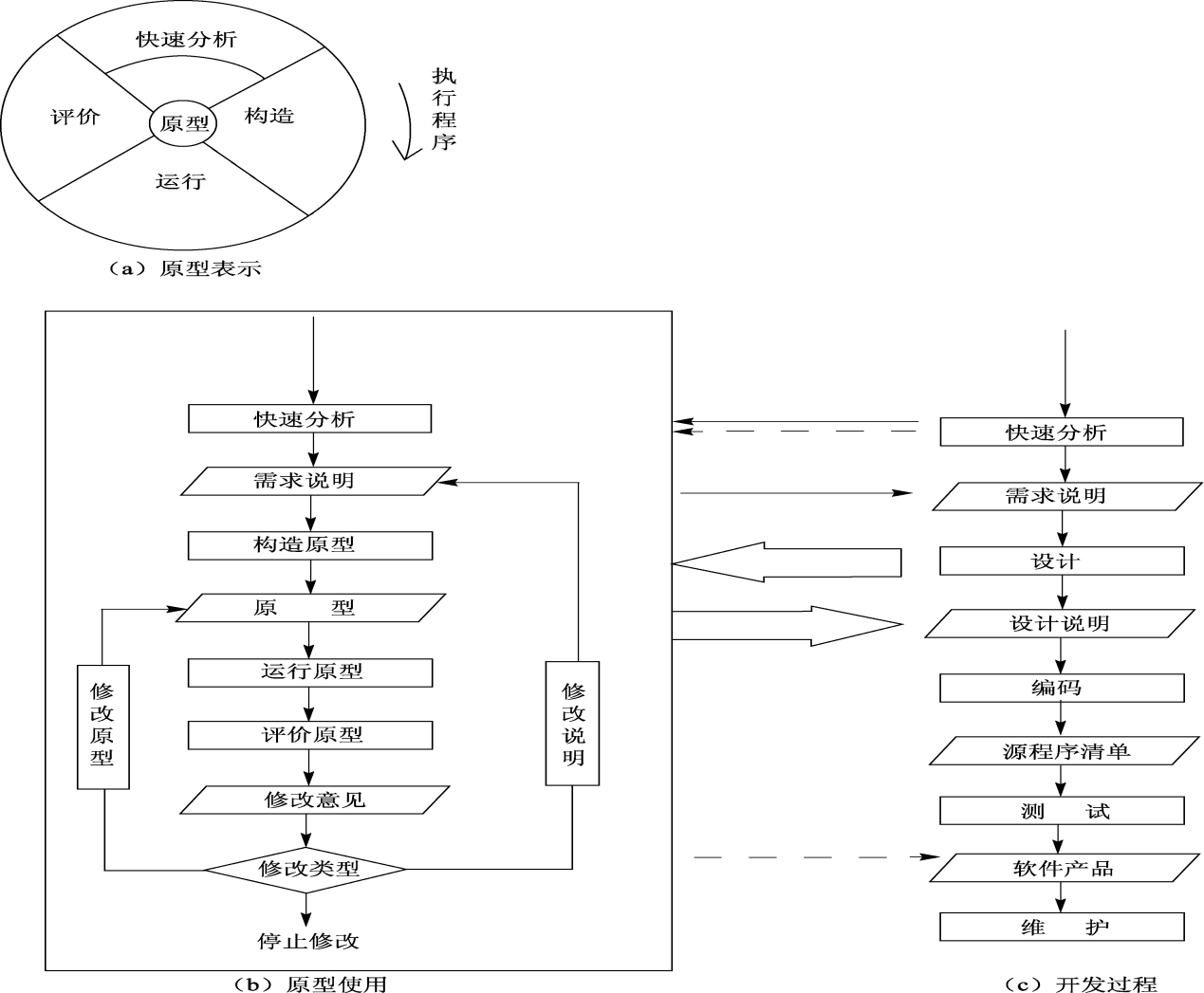
② 每个阶段都要对已完成的**文档进行复审**，以便及早发现隐患，排除故障。

5.3.2 快速原型模型

**快速原型模型（Rapid Prototype Model）**是快速开发一个**可以运行的原型系统**，该原型系统所能完成的**功能往往是最终产品能完成的功能的一个子集。**

请用户**试用原型系统**，以便能准确地认识到他们的**实际需求**是什么，然后编写软件系统的需求规格说明文档，根据这份文档开发出来的软件可以满足用户的真实需求。

符合软件开发的实际规律：**渐进明细**

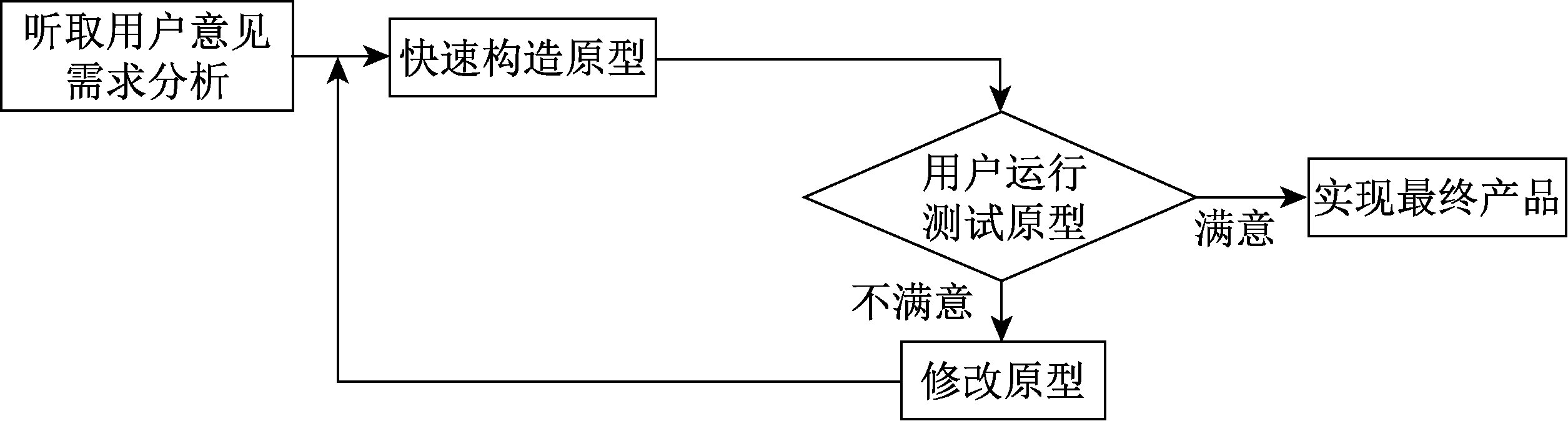


快速原型模型的**第一步所建的原型能完成的功能**，往往是**用户需求的主要功能**。

快速原型模型**鼓励用户参与开发过程**，用户参与原型的运行和评价，能充分地与开发者协调一致。开发期间，原型还可作为终端用户的**教学模型**。开发者一边进行软件开发，一边让用户学习使用软件。若用户发现软件功能不符合自己的实际要求，可**及时提出意见，开发者应立即进行修改**，如此反复进行，直到用户满意为止。

**开发人员听取用户意见，进行需求分析，尽快构造出原型，**原型的作用是获得用户的真正需求**。原型由用户运行、评价和测试，开发人员根据用户的意见修改原型，再次请用户试用，逐步使产品满足用户的需求。产品一旦交付给用户使用，维护便开始。**

**根据需要，维护工作可能返回到需求分析、设计或编码等不同的阶段。**



**5.3.2.1 快速原型实现途径**

**1.进化型的原型**

原型系统不断被开发和被修正，最终称为一个真正的系统。

**2.用于验证软件需求的原型**

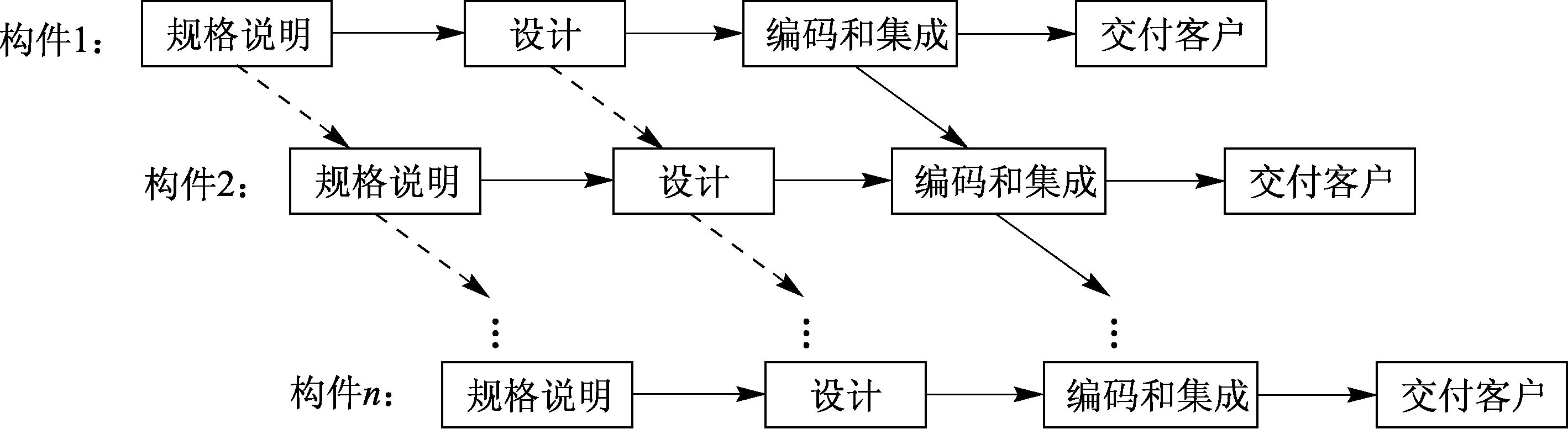
系统分析员在确定了软件需求之后，从中选出某些**需要验证的功能**，用适当的工具快速构造出可运行的原型系统，由用户试用和评价。这类原型往往用后就**丢弃**，因此构造它们的软件环境不必与目标系统的软件环境一致，通常使用简洁而易于修改的高级语言对原型进行编码。

**3.用于验证设计方案的原型**

原型可作为新颖设计思想的实现工具，对于新的设计思想，开发部分软件的原型，可提高风险开发的安全系数，**验证设计的可行性**。如果设计方案验证完成后就将原型**丢弃**，则构造原型的工具不必与设计目标系统的工具一致。如果想把原型作为最终产品的一部分，原型和目标系统可使用同样的软件设计工具。

**5.3.3 增量模型**

增量模型**也称**渐增模型**，**先选择一个或几个关键功能**，建立一个不完全的系统。这个系统**只包含目标系统的一部分功能，**或**对目标系统的功能**从某些方面**作简化**。通过运行这个系统可取得经验，加深对软件需求的理解，从而使系统逐步得到扩充和完善。如此反复进行，直到用户对所设计的软件系统满意为止。**



**5.3.3.1 增量模型优缺点**

**优点**

能在较短时间内向用户提交能完成一定功能的产品，并使用户有较充裕的时间学习和适应产品。

**缺点**

软件的体系结构设计必须是开放的，要便于向现有结构中加入新的构件。每次增量开发的产品都应当是可测试的、可扩充的。

从长远来看，具有开放结构的软件的可维护性明显好于封闭结构的软件。

5.3.3.2 增量模型的增量方式

**1.增量开发**

不是整体地开发软件，而是按一定的时间间隔开发部分软件

**2.增量提交**

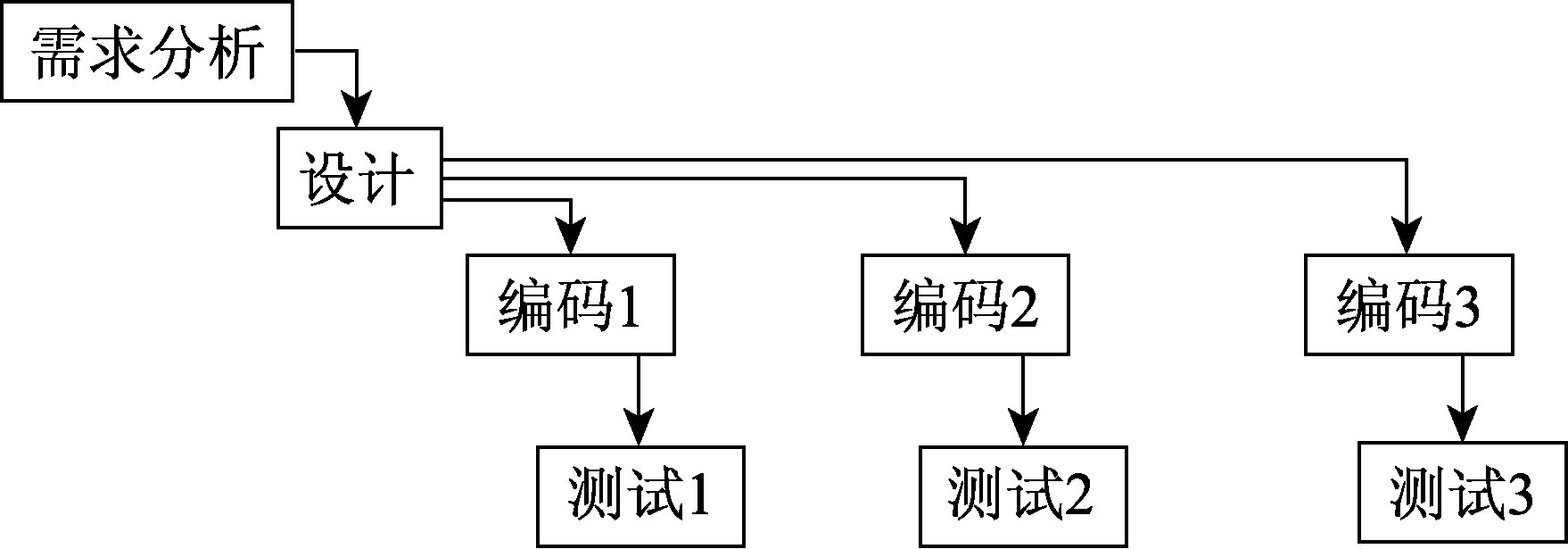
先提交部分软件给用户试用，听取用户意见；再提交另一部分软件，让用户试用；如此反复多次，直到全部提交为止。

**5.3.3.3 增量模型示例**

**增量模型一**

在软件需求分析和设计阶段采用整体开发方式，在编码和测试阶段采用增量模型开发方式**。**

**先对部分功能进行编码、测试，提交给用户试用，听取用户意见，及早发现问题、解决问题；再对另一部分功能进行编码、测试，提交用户试用。**



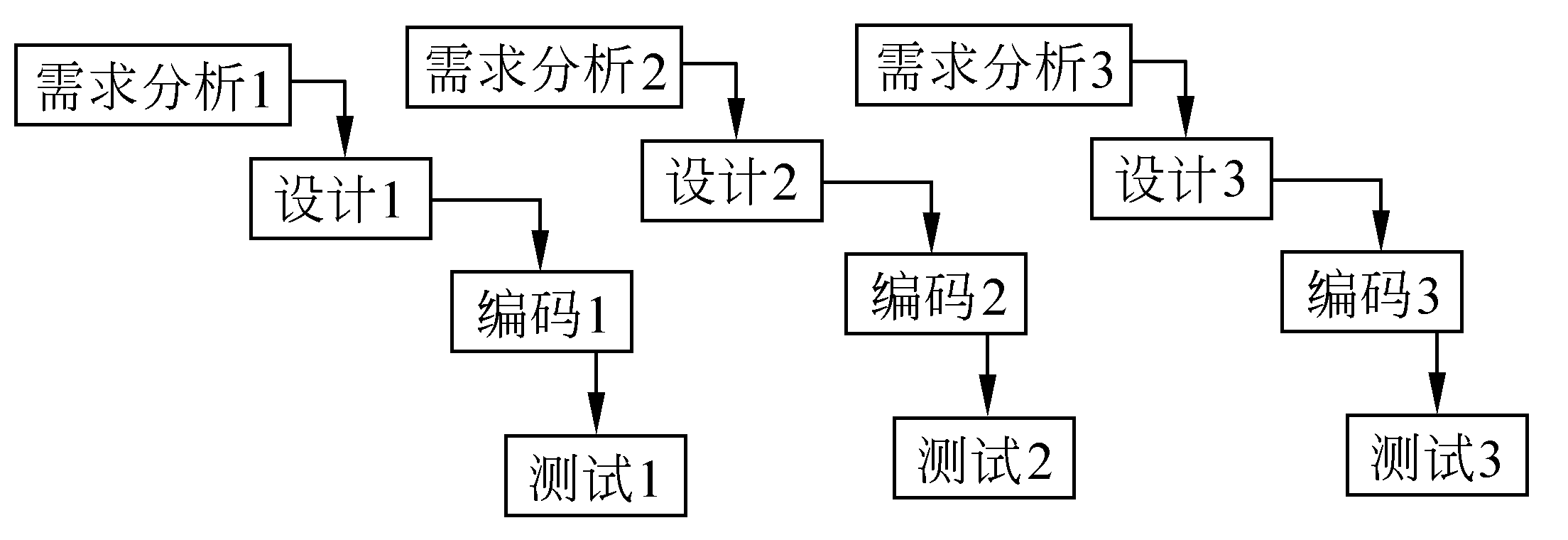
**5.3.3.4 增量模型示例**

**增量模型二**

所有阶段都采用增量模型开发方式。

**先对某部分功能进行需求分析、设计、编码和测试，提交给用户试用，充分听取用户意见；再对另一部分功能进行需求分析、设计、编码和测试，提交给用户试用，直至所有功能开发完毕。**

**用这种方式开发软件时，不同功能的软件构件**可以并行地构建**，因此有可能加快工程进度。但是也存在软件构件无法集成为一个整体的风险。**



**5.3.4 螺旋模型**

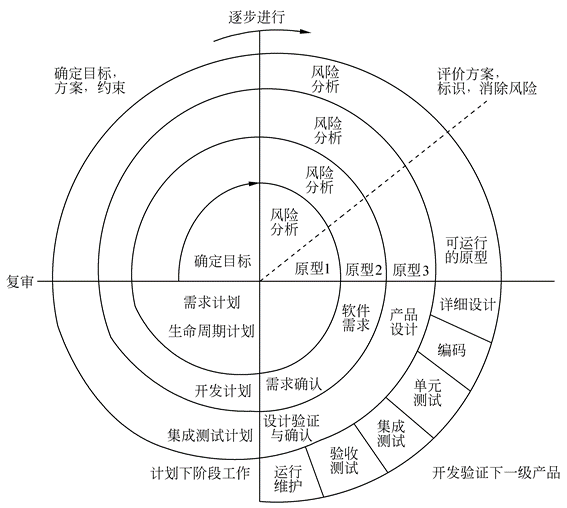
瀑布模型+快速原型+风险分析 迭代过程

瀑布模型要求在软件开发的初期就完全确定软件的需求，这在很多情况下往往是做不到的。螺旋模型试图克服瀑布模型的这一不足之处。

螺旋模型（Spiral Model，SM）是1986年由B．W．Boehm提出的。它把软件开发过程安排为逐步细化的螺旋周期序列，每经历一个周期，系统就细化和完善一些。

每一螺旋周期由6个步骤组成。

**确定计划 风险分析**



**客户评估 实施工程**

**5.3.4.1 螺旋模型六大步骤**

**1.确定任务目标**根据初始需求分析项目计划，确定任务目标、可选方案和限制。

**2.选择对象**对各种软硬件设备、开发方法、技术、开发工具、人员、开发管理等对象进行选择，并决定软件是进行研制、购买还是利用现有的。

**3.分析约束对象**分析软件开发的时间、经费等限制条件。

**4.风险分析**评估目标、对象、约束条件三者之间的联系，列出可能出现的问题及问题的严重程度等，把最重要的问题作为尚未解决的关键问题，称为风险。

**5.制定消除风险的方法**应有详尽的说明和周密的计划，并估计可能产生的后果。依此来开发软件，为制定下一周期的计划打下基础。

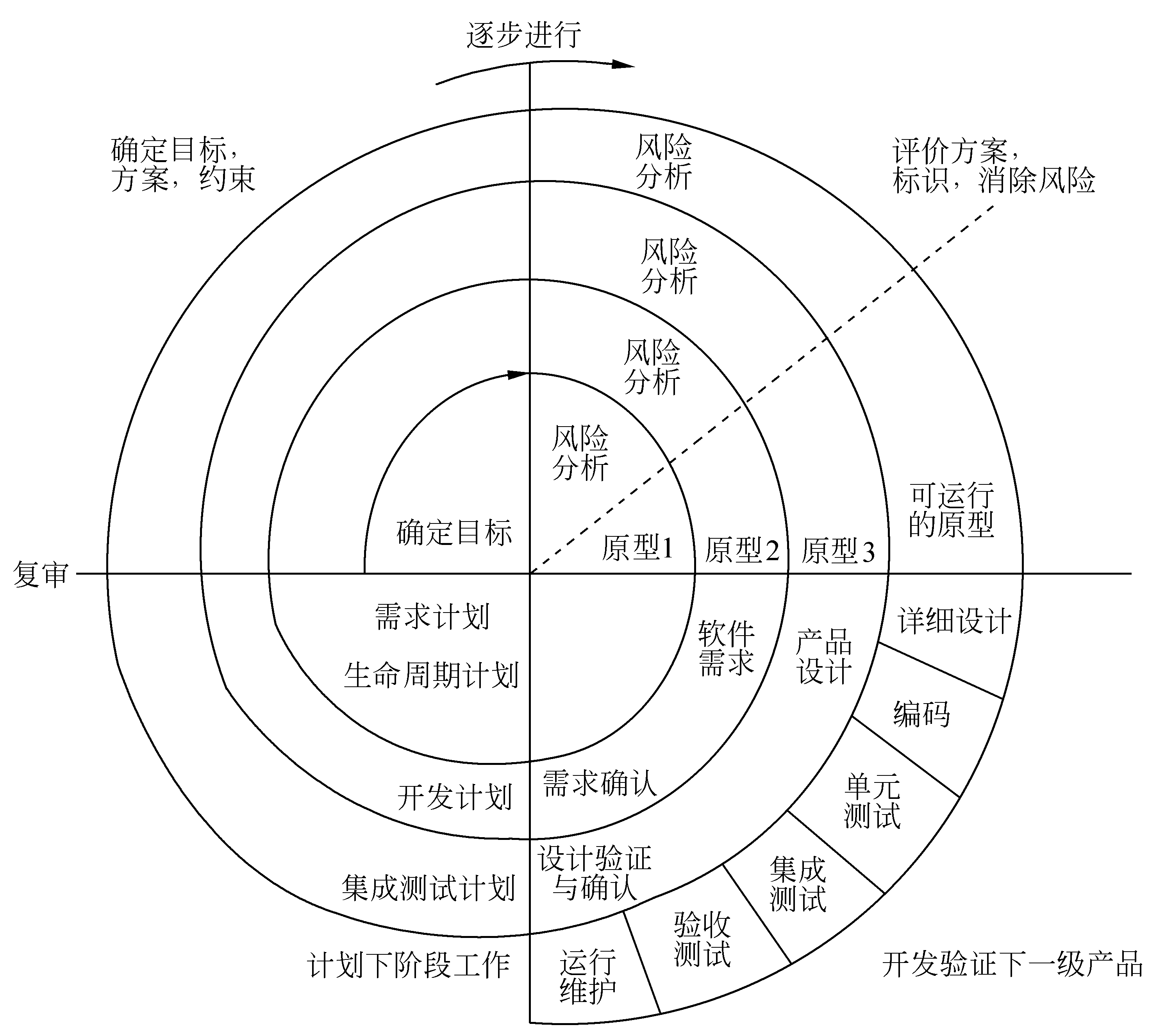
**制定下一周期的工作计划**

在**第一个螺旋周期**，确定目标、选择对象、分析约束，通过风险  
分析制定消除风险的方法，初步开发**原型1**，制定**系统开发计划**。

在**第二个螺旋周期**，进一步明确系统的目标、开发方案及约束条件，  
通过风险分析制定消除风险的方法，在原型1的基础上开发**原型2**。进一步**明确软件需求**，进行需求确认，修改开发计划。

在**第三个螺旋周期**，再进一步确认系统目标、开发方案及约束条件，进行风险分析，制定进一步消除风险的方法，在原型2的基础上开发**原型3**。此时可进行**产品设计**，再对设计进行验证和确认，制定**集成测试计划**。

在**第四个螺旋周期**，软件开发方案、系统目标和约束条件得到确定，在风险分析的基础上，开发出具有实用价值的**可操作原型**，此时可对产品进行**详细设计**，进入**编码**、**单元测试**、**集成测试阶段**，最后进入**验收测试**，验收合格后**交付**用户使用，进入**运行、维护**阶段。

****

**5.3.5 喷泉模型**

按传统的瀑布模型开发和维护软件，需要有**两个前提**：

① 用户能清楚地提供系统的需求；

② 开发者能完整地理解这些需求，软件生存周期各阶段能明确地划分，每个阶段结束时要复审，复审通过了后一阶段才能开始。

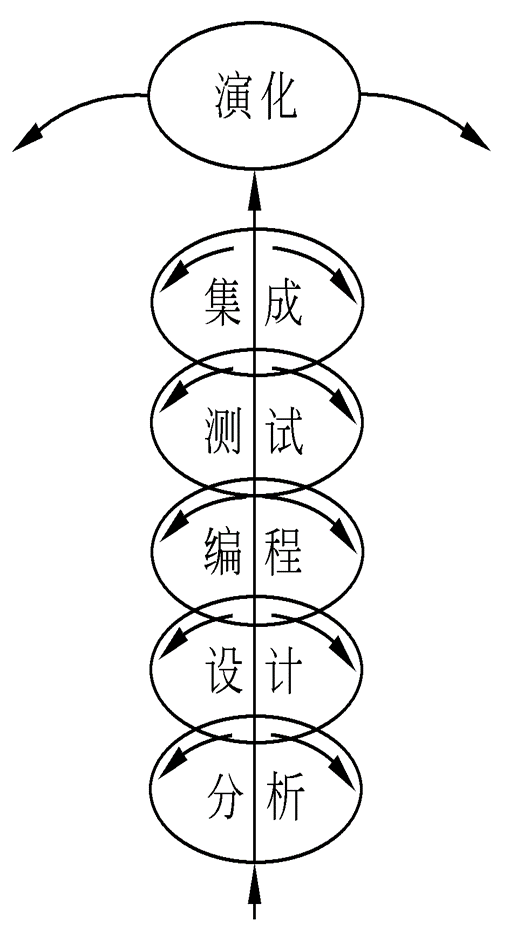
然而，在实际开发软件时，用户往往事先难以说清系统需求，开发者也由于主客观的原因，缺乏与用户交流的机会，其结果是系统开发完成后，修改和维护的开销及难度过大。

喷泉模型是一种**以用户需求为动力，以对象作为驱动**的模型。它克服了瀑布模型不**支持软件重用和多项开发活动集成**的局限性，开发过程具有**迭代性**和**无间隙性**。

* 无缝：各阶段都使用统一的概念和表示符号——对象
* 迭代：容易实现各步骤的多次反复迭代。

喷泉模型（Fountain Model）是典型的**面向对象软件开发模型**，着重强调不同阶段之间的重叠，认为面向对象的软件开发过程不需要或不应该严格区分不同的开发阶段。基于喷泉模型，Hodge等人提出将软件开发过程划分为**系统分析、系统设计、对象设计与实现（编程）、测试和系统组装集成**等阶段。

* 圆圈重叠：不同阶段的重叠，各活动存在交迭 — 无缝
* 中轴线：总目标，避免开发无序
* 向下的箭头：该阶段内的迭代

****

“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**。

**图中代表不同阶段的圆圈相互重叠，这表示两个活动之间存在交迭**；而面向对象方法的概念和表示方法上的一致性，保证了在各项开发活动之间的无缝过渡，事实上，用面向对象方法开发软件时，在分析、设计和编码等项开发活动之间并不存在明显的边界。

使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**。由于在整个开发过程中都使用统一的软件概念“对象”，所有其他概念（例如功能、关系、事件等）都是围绕对象组成的，因此对生命周期各阶段的区分自然就不重要、不明显。

软件过程各个阶段之间的迭代或一个阶段内各个工作步骤之间的迭代，在面向对象范型中比在结构化范型中更常见。每次反复都会增加或明确一些目标系统的性质，但却不是对先前工作结果的本质改动，这样就减少了不一致性，减低了出错的可能性。

5.3.5.1 喷泉模型六大步骤

**1.系统分析**

在系统分析阶段**建立对象模型和过程模型**。系统模型中的对象是对现实世界中客观对象的抽象，应当结构清晰、易于理解、易于描述和规范。

**2.系统设计**

给出对象模型和过程模型的**规范描述。**

**3.对象设计与实现（编程）**

在软件开发时，先检索对象库，若是对象库中**已经存在的对象**，则可不必设计，只要重复使用或加以修改后使用；否则，应**定义新的对象**，进行设计和实现。

**4.测试**

测试所有的**对象**及对象相互之间的**关系**是否符合要求。

**5.系统组装集成**

面向对象软件的特点之一是**软件重用和组装技术**。对象是数据和操作的封装载体，组装在一起才构成完整的系统。**模块组装**也称**模块集成**、**系统集成**。软件设计是将对象模块集成、构造生成所需的系统。

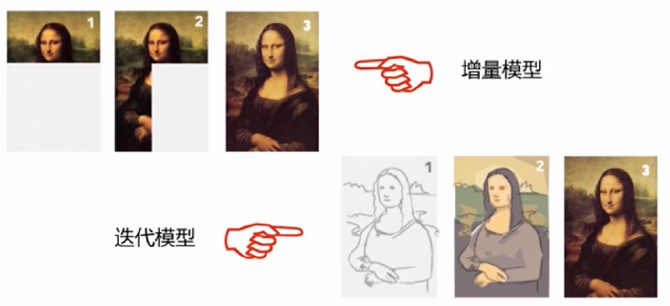
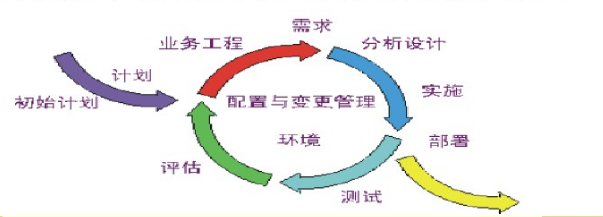
**6.演化**

由于喷泉模型主张**分析和设计过程的重叠**、不严格加以区分，模块集成过程要反复经过分析、设计、测试、集成这几个阶段，每次集成都使系统功能在原有基础上得到扩展，因而称为**系统演化**。

**5.3.6 迭代模型**

**迭代模型**将整个项目的开发目标**划分成为一些更易于完成和达到的阶段性小目标**，这些小目标都有一个定义**明确的阶段性评估标准**。

**迭代**就是为了完成一定的阶段性目标而所从事的一系列开发活动，在每个迭代开始前都要根据项目当前的状态和所要达到的阶段性目标制定迭代计划，**整个迭代过程包含**了需求、设计、实施（编码）、部署、测试等各种类型的开发活动，迭代完成之后需要对迭代完成的结果进行评估，并以此为依据来制定下一次迭代的目标。



5.3.6.1迭代模型的特点

**1.允许变更需求**

通过向用户演示迭代所产生的部分系统功能，我们可以尽早地**收集用户对于系统的反馈**，**及时改正**对于用户需求的**理解偏差**，从而保证开发出来的系统真正地解决客户的问题。

**2.逐步集成元素**

在迭代式方法中，集成可以说是连续不断的，每一次迭代都会增量式集成一些新的系统功能，要集成的元素都比过去少得多，所以**工作量和难度都是比较低**的。

**3.尽早降低风险**

迭代化开发的**主要指导原则就是以架构为中心**，在早期的迭代中所要解决的主要问题就是尽快确定系统架构，通过几次迭代来尽快地设计出能够满足核心需求的系统架构，这样可以迅速降低整个项目的风险。等到系统架构稳定之后，项目的风险就比较低了，这个时候再去实现系统中尚未完成的功能，进而完成整个项目。

**4.有助于提高团队的士气**

开发人员通过每次迭代都可以**在短期内看到自己的工作成果**，从而有助于他们**增强信心**，更好地完成开发任务。

**5.生成更高质量的产品**

**每次迭代都会产生一个可运行的系统**，通过对这个可运行系统**进行测试**，我们在早期的迭代中就可以**及时发现缺陷并改正**，性能上的瓶颈也可以尽早发现并处理。因为在每次迭代中总是不断地纠正错误，我们可以得到更高质量的产品。

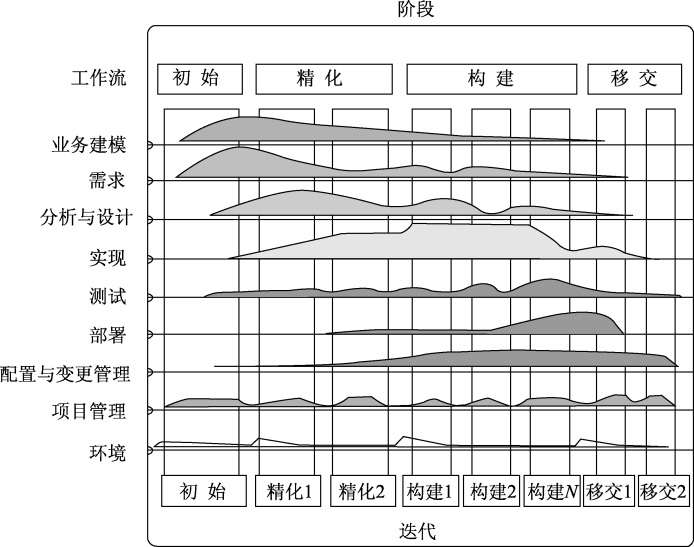
**6.保证项目开发进度**

每次**迭代结束时都会进行评估**，来**判断该次迭代有没有达到预定的目标**。项目经理可以很清楚地知道有哪些需求已经实现了，并且比较准确地估计项目的状态，对项目的开发进度进行必要的调整，保证项目按时完成。

5.4 两个典型的软件过程模型

**5.4.1 统一过程RUP：Rational Unified Process**

RUP对软件过程模式中的四大要素（生命周期、人员、方法、产品）及相互关系进行了详尽的论述。在“生命周期”方面，**RUP构架了一个迭代与增量的二维生命周期结构。**

****

**一个活动在每次迭代中都有**

**每个阶段又细化为若干迭代1个迭代，贯穿工作流**

* **横轴为生命周期的四个阶段**：初始、精化、构建和移交，各阶段结束于一个项目里程碑。
* **纵轴为九个核心工作流程**：业务建模、需求、分析设计、实现、测试、部署、配置与变更管理、项目管理、环境。

**5.4.2 敏捷开发过程Agile**

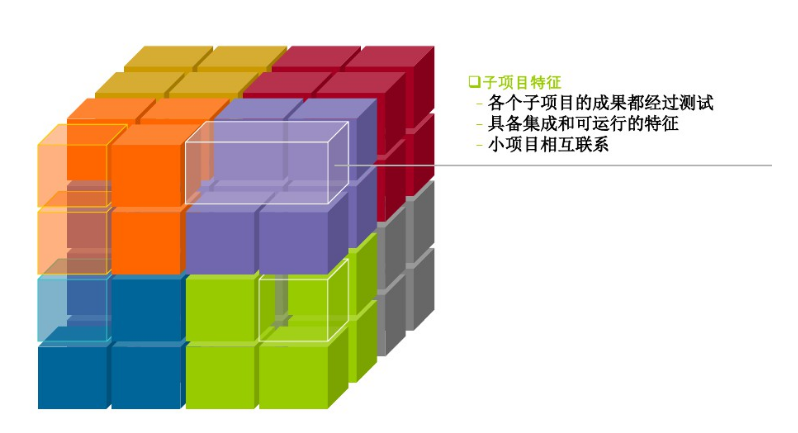
敏捷开发是一种以人为核心、迭代、循环渐进的开发方法。

SCRUM：

* + 是一种迭代的增量化过程，用于产品开发或工作管理

XP: eXtreme Programming 极限编程

* + 思想源自Kent Beck和Ward Cunningham在软件项目中的合作经历



**6.1 软件过程改进的概念&目标意义**

**软件过程改进**，就是在软件过程工程中为了更有效地达到**优化软件过程**的目的，同时**改善或改变**其软件过程的一系列活动。

软件开发的风险之所以大，是由于软件过程能力低。其中最关键的问题在于**软件开发机构不能很好地管理其软件过程**，从而使得一些好的开发方法和技术起不到预期的作用。

**软件过程改进的目标和意义在于**：

1）提升工作效率

2）提高工作质量

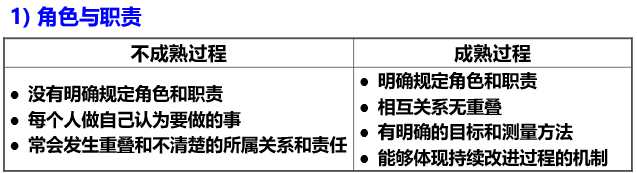
3）降低项目风险

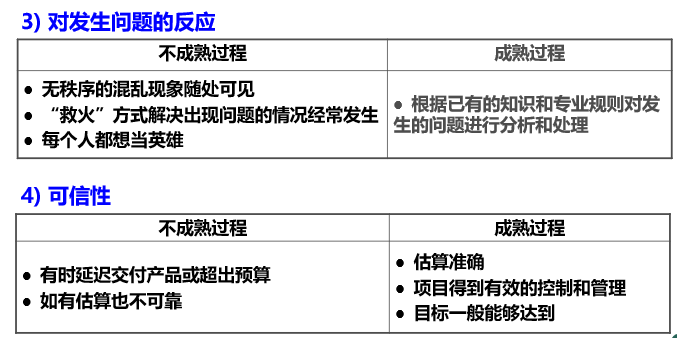
**` 6.2 软件过程成熟度的概念**

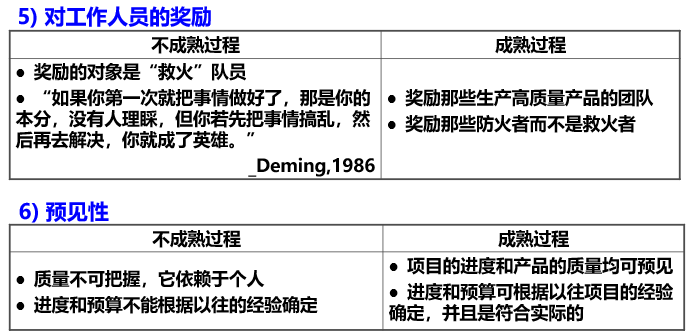
任何一个软件组织，在完成自身的开发、维护等工作中，都有自己的软件过程。这种过程有可能是初级的、低效的，也可能是高效的，在其成熟性方面存在差异，这当然是相互比较而言的。

**软件过程成熟度（Software Process Maturity）**是软件过程改进的一个重要概念，它用于描述**软件过程被明确和有效地定义、管理、测量和控制的程度**。成熟度意味着能力的增长具有潜力，并表示组织软件过程是珍贵的，它在组织内所有项目中的应用是一致的。通常用软件过程能力来描述软件企业遵循其软件过程能够实现的预期结果。

**6.3 不成熟过程与成熟过程的对比**

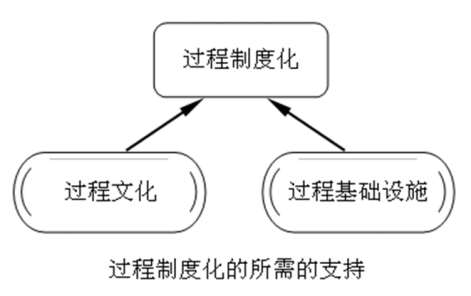
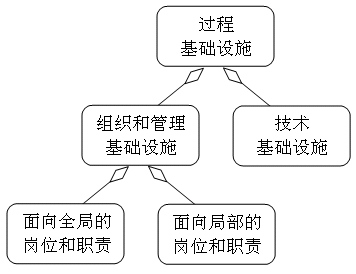






当一个规范化过程已经渗入组织的日常生活之中，过程的要求已经变成全体员工的自觉行动，得到大家的认同和坚持遵循时，**过程便成为制度化的**（process institutionalization）。

做到这一点并不容易，要靠**过程文化**和**过程基础设施**的支持。

 →

**6.4 过程认同和过程制度化**

**过程文化**

过程文化是指**人们的习惯和行为受到过程思维和过程管理原则的影响**。人们对于规范化过程是完全认同的，就是说，**人们的活动自觉地按过程要求去作**。

**过程基础设施**

对于软件过程来说，**基础设施指的是支持软件过程的基础框架和结构基础**。它不仅包括**组织和管理的岗位和职责**，而且包括支持定义过程、开展过程活动、获取和分析过程**有关绩效反馈**以及不断进行过程改进活动所必须的**技术工具和平台**。事实上，过程基础设施包含了**组织和管理基础设施**、**技术基础设施**两个方面。

这里所讲的**问题**是指用户的基本要求，就是**确切地定义用户要求解决的问题**，即确定问题的性质，工程的目标和规模。

**问题定义的目标**是要求在短时间内对用户的要求有一个比较准确的估计，还要弄清用户不打算干什么，在这个系统中哪些内容不用实现等问题。**工作的宗旨是弄清要做什么并划清要实现系统的范围边界**。

**7.2 软件问题定义的主要内容**

**1. 基本要求**

包括软件的**功能**、**性能**、**输入**（数据的来源、类型、数量，数据的组织以及提供的频度）、**输出**（如报告、文件或数据，说明其用途、产生频度、接口及分发对象）、**处理流程**和**数据流程**、**安全**和**保密**方面的要求、同本系统相连接的**其他系统**等。

**2.系统目标**

例如，人力与设备费用的减少；处理速度的提高；控制精度或生产能力的提高；管理信息服务的改进；人员利用率的改进等。

**3. 系统开发条件，假定和限制**

例如，系统运行寿命的最小值；经费、投资的来源和限制；法律和政策的限制；硬件、软件、运行环境和开发环境的条件及限制；可利用的信息和资源；完成期限等。

**4.可行性研究方法**

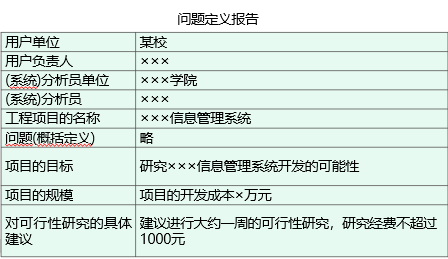
**可采用调查、加权、确定模型、建立基准点或仿真等方法进行可行性研究。**

**5.评价尺度**

例如，经费的多少，各项功能的优先次序，开发时间的长短及使用的难易程度等。

**7.3 软件问题定义报告**

系统分析员对问题有了明确的认识之后，要把自己的认识写在书面报告，形成**问题定义报告**，提交给用户审查，以检验对所要问题的理解是否正确。



**8.1 可行性研究的目的**

**可行性研究的目的是确定问题是否值得去解决!**

可行性研究实质上是要进行一次大大压缩简化了的系统分析和设计过程，即在**较高层次上以较抽象的方式进行的系统分析和设计的过程**。

1）进一步分析和澄清问题定义（规模和目标；肯定or改正；约束和限制）。

2）导出系统的逻辑模型，探索**若干种可供选择的主要解法**（即系统实现方案）。

3）对每种解法从经济、技术和社会因素等方面**每种解法的可行性**

**8.2 典型的可行性研究步骤**



**8.3 技术可行性**

**1.技术可行性定义**

技术可行性是指对**设备条件、技术解决方案的实用性和技术资源的可用性**的度量。

在决定采用何种开发方法和工具时，必须考虑设备条件，通常选择实用的、开发人员掌握较好的一类。

简单表述为：做得了么？做得好么？做得快么？

**8.4 经济可行性**

**1.经济可行性定义**

经济可行性是指希望以**最小的成本开发出具有最佳的经济效益的软件产品**。包括：“成本-收益分析”、“短期-长远利益”分析

**2.投资和效益分析**

（1）**支出**。说明所需的费用，包括基本建设投资，移动设备和固定设备购置费用，操作系统、应用软件、网络软件和数据库管理软件费用，其他一次性支出及非一次性支出费用。

（2）**收益**。

（3）**收益/投资比**。

（4）**投资回收周期**（一般假定为5年）。

（5）**敏感度分析**。

**8.5 社会因素方面的可行性**

**1.社会因素可行性定义**

**社会因素方面的可行性**主要从市场、政策、法律、用户等方面分析可行性。

**2.法律可行性**

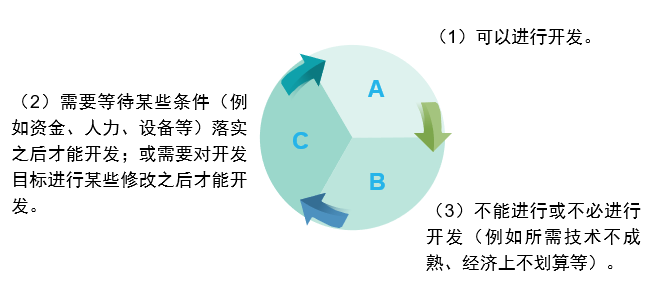
**法律方面的可行性**是指要开发的项目是否存在任何侵权、妨碍等责任问题；

**3.用户使用和操作可行性**

**用户使用可行性**是指使用软件对用户内部组织管理制度的影响程度；

**用户操作可行性**是指软件系统所采用的操作方式对用户来说是否可行。

**8.6 可行性分析结论**



**8.7 可行性分析文档内容**

**1.引言**

引言说明编写本文档的目的，项目的名称、背景，本文档用到的专业术语和参考资料。

**2.可行性分析前提**

可行性分析前提说明开发项目的功能、性能和基本要求、达到的目标，各种限制条件、可行性分析方法和决定可行性的主要因素。

**3.对现有系统的分析**

对现有系统的分析说明现有系统的处理流程和数据流程、工作负荷、各项费用的支出、所需各类专业技术人员和数量、所需各种设备，现有系统存在的问题。

**4.所建设系统的可行性分析**

所建设系统的可行性分析简要说明所建设系统的处理流程和数据流程，与现有系统比较的优越性，采用所建议系统对用户的影响，对各种设备、现有软件、开发环境和运行环境的影响，对经费支出的影响，对技术可行性的评价。

**5.所建设系统的经济可行性分析**

所建设系统的经济可行性分析说明所建设系统的各种支出、各种效益、收益/投资比，资金回收周期。

**6.社会因素可行性分析**

社会因素可行性分析说明法律因素对合同责任、侵犯专利权和侵犯版权等问题的分析。说明用户使用可行性是否满足用户行政管理，工作制度等要求。

**7.其他可选方案**

其他可选方案逐一说明其他可选方案，并说明未被推荐的理由。

**8.结论意见**

结论意见说明项目是否能开发，还需要什么条件才能被开发，对项目目标有何变动等。

**8.8 可行性研究总结**

**可行性研究阶段**不要急于着手解决问题，其主要目的是得到系统确实可行的结论，或及时中止不可行的项目。**应避免在项目进行了较长时间后，才发现项目根本不可行，以致造成浪费**。**可行性报告**要得到用户单位决策者的认可，所提出的结论要有具体、充分的理由。由**用户单位的决策者根据可行性报告，选择决定所采用的具体解决方案**。**可行的项目才能进入下一步——即制订项目的开发计划和实施阶段。**

**9.1 需求分析的目标与任务**

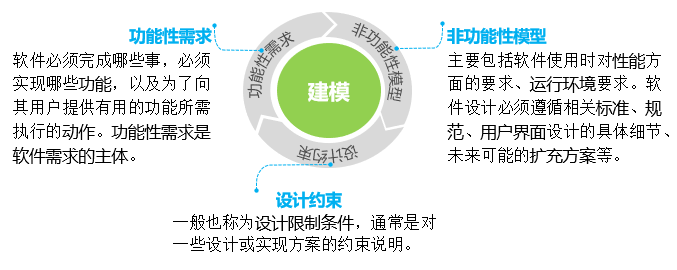
**需求分析的目标：**是把用户对待开发软件提出的“要求”或“需要”进行分析与整理，确认后形成描述完整、清晰规范的文档，**确定软件需要实现哪些功能，完成哪些工作**。此外，还包括软件的一些**非功能性需求**（如软件性能、可靠性、响应时间、可扩展性等）、软件设计的**约束条件**、**运行时与其他软件的关系**等。

**需求分析的主要任务：**

（1）建立分析模型

（2）编写软件需求规格说明书

**9.2 需求分析的内容**



9.3 需求分析的步骤

**一、问题识别**

从系统角度来理解软件，确定对所开发系统的综合要求，并提出这些需求的实现条件，以及需求应该达到的标准。

**二、分析与综合**

逐步细化所有的软件功能，找出系统各元素间的联系，接口特性和设计上的限制，分析他们是否满足需求，剔除不合理部分，增加需要部分。 最终给出详细的逻辑模型。

**三、制定规格说明书**

即编制文档，描述需求的文档称为软件需求规格说明书。需求分析阶段的成果是需求规格说明书，向下一阶段提交。

**四、评审**

对功能的正确性、完整性和清晰性，以及其他需求给予评价。评审通过才可进行下一阶段的工作，否则重新进行需求。

9.4 需求分析的法则

**遵照以下20条法则，客户与开发人员通过评审可达成共识**

1、分析人员要使用符合客户语言习惯的表达 2、分析人员要了解客户的业务及目标

3、分析人员必须编写[软件需求](http://baike.baidu.com/view/10192.htm)报告 4、要求得到需求工作结果的解释说明

5、开发人员要尊重客户的意见 6、开发人员要对需求及产品实施提出建议和解决方案

7、描述产品使用特性 8、允许重用已有的软件组件

9、要求对变更的代价提供真实可靠的评估 10、获得满足客户功能和质量要求的系统

11、给分析人员讲解您的业务 12、抽出时间清楚地说明并完善需求

13、准确而详细地说明需求 14、及时作出决定

15、尊重开发人员的需求可行性及成本评估 16、划分需求的优先级

17、评审需求文档和原型 18、需求变更要立即联系

19、遵照开发小组处理需求变更的过程 20、尊重开发人员采用的需求分析过程

9.5 需求分析的辅助工具

**Rational Rose** 是Rational公司出品的一种面向对象的统一建模语言的可视化建模工具。

**Axure RP** 能帮助网站需求设计者，迅捷而轻便的创立 基于目录组织的原型文档、功能解释、交互界面以及带注释的wireframe网页，并可积极生成用于演示的网页文件和word文档，以供给演示与开发。

**StarUML** 是一套开源的软件，可描摹9款UML图：用例图、类图、序列图、事态图、行动图、通信图、模块图、安排图以及复合构造图等。

**Microsoft visio** 能够发生流程图、组织图、工夫表、营销图和其它更多图表，把特定的图表加入文件，让商业沟通变得更加打听，令演示更加有趣。

**FreeMind** 思维导图软件，是一实用的开源思维导图。它可用来作为管教项目，笔记或知识库，文章写作可能头脑风暴，构造化的存储小型数据库，描摹思维导图，整理软件流程思路。

**9.6 需求分析的方法**

从系统分析出发，可将需求分析方法大致分为

（1）功能分解方法：功能分解

（2）**结构化分析方法**：跟踪数据流，包括数据流图和数据字典

（3）信息建模法：从数据角度对现实世界建立模型，基本工具包括E-R图

（4）**面向对象的分析方法**：识别对象、分析关系

**9.7 结构化分析方法**

**9.7.1 基本思想**

结构化分析方法的基本思想是**“分解”**和**“抽象”**。

**分解**：是指对于一个复杂的系统，为了将复杂性降低到可以掌握的程度，可以**把大问题分解成若干小问题**，然后分别解决。

**抽象**：分解可以**分层**进行，即先考虑问题最本质的属性，暂把细节略去，以后再逐层添加细节，直至涉及到最详细的内容，这种**用最本质的属性表示一个自系统的方法就是“抽象”**。

**9.7.2 分析步骤**

（1）建立**当前系统**的“**具体模型**”，即将当前系统用数据流图（DFD） 图描述出来。

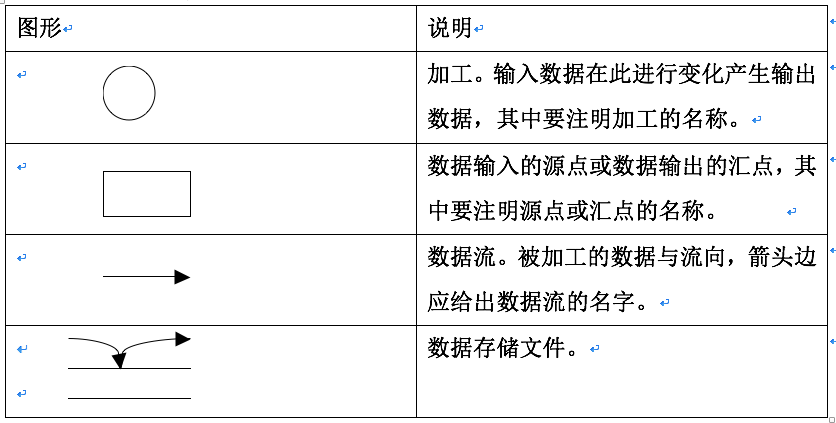
（2）抽象出**当前系统**的**逻辑模型**，分析系统的“具体模型”，获得用DFD 图描述的当前系统的“逻辑模型”。

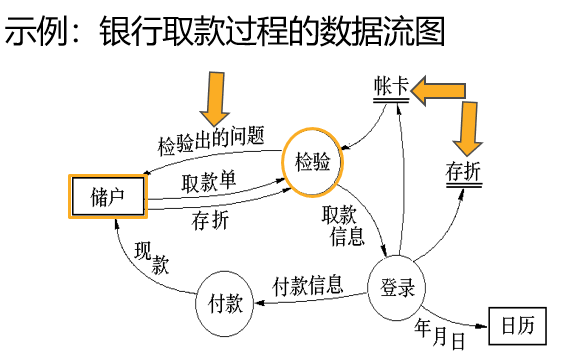
（3）建立**目标系统**的**逻辑模型**，分析目标系统与当前系统逻辑上的**差别**，建立目标系统的“逻辑模型”(修改后的DFD 图)。

（4）为了对目标系统作完整的描述，还需要考虑人机界面和其它一些问题。

**9.7.3 数据流图**

数据流图的基本图形元素有4种：

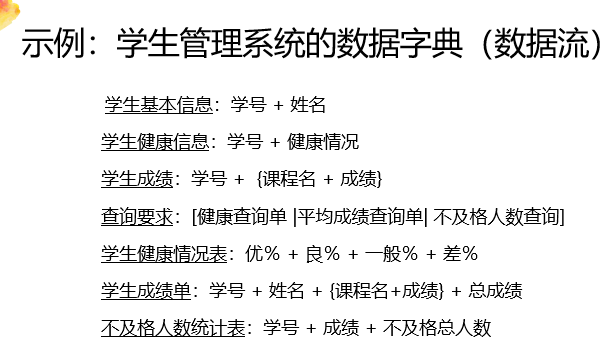




9.7.4 数据字典

数据词典(Data Dictionary，DD)是存放数据库各级模式结构的描述，也是访问数据库的接口。

数据字典是指对**数据**的数据项、数据结构、数据流、数据存储、处理逻辑、外部实体等进行定义和描述，其**目的**是**对数据流程图中的各个元素做出详细的说明**。



**9.7.5 加工逻辑说明**

加工逻辑也称为小说明。

在数据流图中，如果每个加工框中只简单地写上一个加工名，这显然不能表达加工的全部内容。随着自顶向下逐层细化，功能越来越具体，加工逻辑也越来越精细。在最底层，加工逻辑详细到可以实现的程度，也就是基本加工。

如果能够写出**每个基本加工的全部详细逻辑功能**，再自底向上综合，就能完成全部逻辑加工。

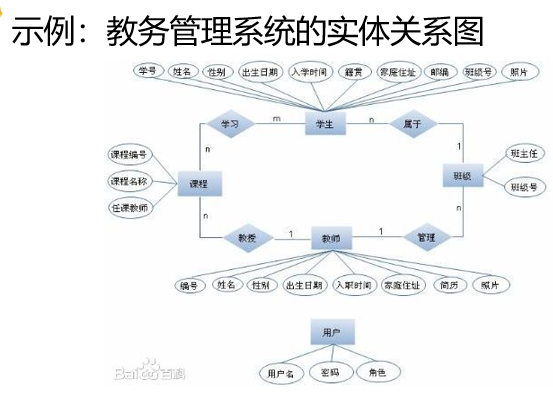
描述加工逻辑一般用以下三种工具：结构化语言、判定表、判定树。

**9.7.6 实体关系图**

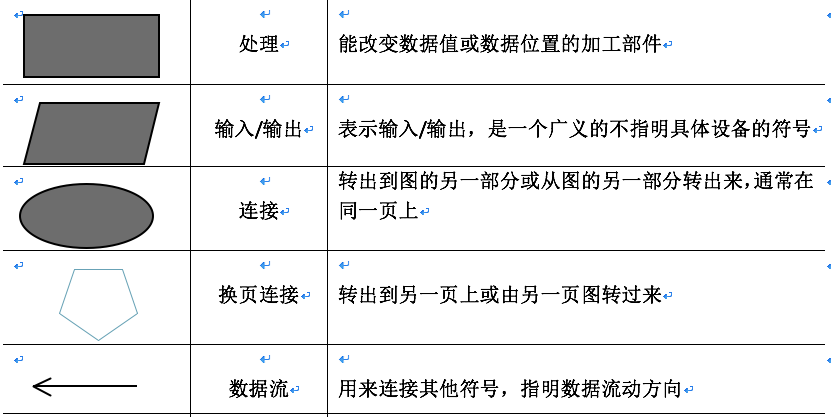
为了把用户数据要求清晰地表达出来，满足**分析数据要求**，分析人员经常使用**实体关系(Entity Relationship，ER)图**来描述现实世界中的**实体及相互间的关系**。

ER图中包含实体、属性和联系3个基本成分。

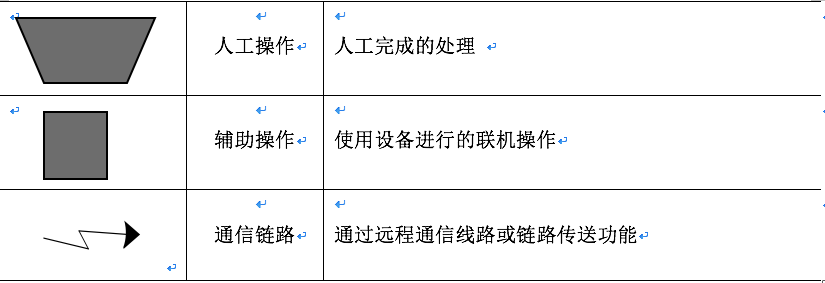
实体之间的联系可以有3类：一对一的联系(1：1)、一对多的联系(1：M)、多对多的联系(M：N)。

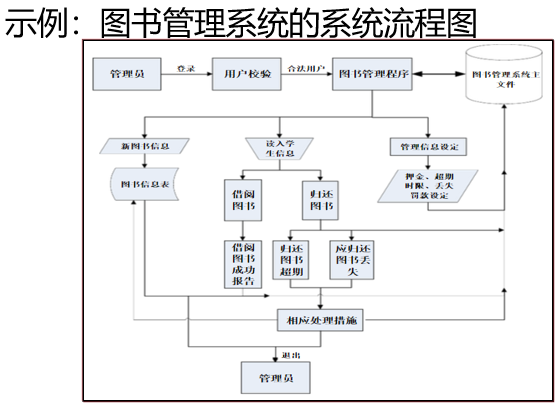


**9.7.7 系统流程图**









**9.8 UML概述**

3.8.1 UML的发展

**统一建模语言（UML）**是由世界著名的**面向对象**技术  
专家Grady Booch，Jim Rumbaugh和Ivar Jacobson发起，  
在**面向对象**的Booch方法、**对象建模技术**（Object Modeling Technique，OMT）和**面向对象软件工程**（Object Oriented Software Engineering，OOSE）的基础上，不断进行完善和发展的一种建模语言。

1996年年底，UML已经稳定地占领了面向对象技术市场的85%，成为事实上的工业标准。1997年11月，国际对象管理组织（OMG）批准把UML 1.1作为基于面向对象技术的标准建模语言。在计算机学术界、软件产业界、商业界，UML已经逐渐成为人们为各种系统建立模型，描述系统体系结构、商业体系结构和商业过程时使用的统一工具，在实践过程中人们还在不断扩展它的应用领域。对象技术组织（Object Technology Organization）已将UML作为对象建模技术（OMT）的行业标准。

**1.模型是为了理解事物而对事物做出的一种抽象**，是对事物的一种书面描述。通常，模型由一组图形符号和组织这些符号的规则组成，模型的描述应当无歧义。

在开发软件系统时，建立模型的目的是为了更好地理解问题、减少问题的复杂性、验证模型是否满足用户对系统的需求，并在设计过程中逐步把实现的有关细节加进模型中，最终用程序实现模型。

**2.UML采用了面向对象的概念**，引入了各种独立于语言的表示符号。UML通过建立用例模型、静态模型和动态模型完成对整个系统的建模，所定义的概念和符号可用于软件开发过程的分析、设计和实现的全过程。软件开发人员不必在开发过程的不同阶段进行概念和符号的转换。

3.面向对象软件工程（OOSE）方法的最大特点是**面向用例（Use Case）**。用例代表某些用户可见的功能，实现一个具体的用户目标。用例代表一类功能而不是使用该功能的某一具体实例。用例是精确描述需求的重要工具，贯穿于整个软件开发过程，包括对系统的测试和验证过程。

**9.8.2 UML内容**

1.UML是一种描述、构造、可视化和文档化的**软件建模语言**。

2.UML是**面向对象**技术中软件分析与设计的**标准建模语言**，它统一了面向对象建模的基本概念、术语及其图形符号，建立了便于交流的通用语言。

3.UML采用图形表示法，是一种**可视化的图形建模语言**。UML的主要内容包括UML语义、UML表示法和几种模型

4.UML表示法为建模者和建模工具的开发者提供了**标准的图形符号和文字表达的语法**。这些图形符号和文字所表达的是应用级的模型，使用这些图形符号和正文语法为系统建模构造了标准的系统模型。UML表示法由UML图、视图、模型元素、通用机制和扩展机制组成。

**9.9 面向对象分析方法**

面向对象分析（object-oriented analysis，OOA）是利用面向对象的概念和方法来开发一系列模型，这些模型描述计算机软件，从而满足客户定义的需求。

面向对象分析方法从理解系统的“**使用实例**”开始，基本步骤：

1）定义系统的**用例（User Case）**

2）在领域分析的基础上建立**问题域的类**（**对象模型**）

3）建立**对象关系**、**对象行为模型**

**9.9.1 定义系统用例**

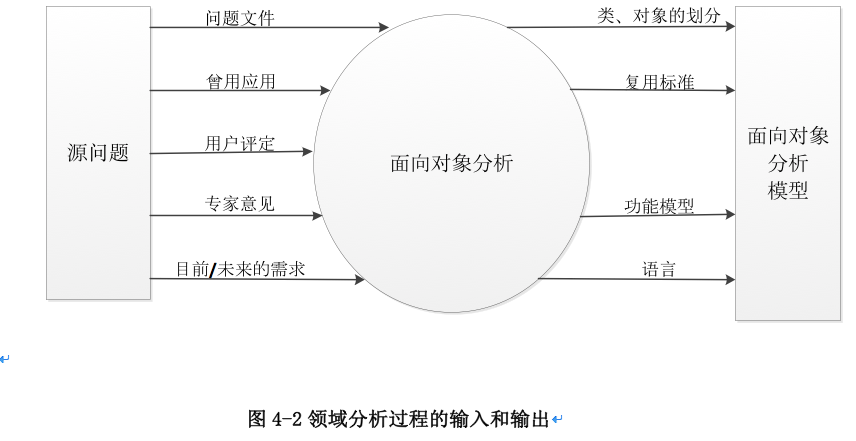
一个**用例**就是参与者与计算机之间为达到某个  
目的的一次典型交换作用。

用例由**参与者和动作**组成：

* **参与者**代表了系统运行时的人员(或设备)，是存在于系统外部与系统通信的实体。**用户与参与者不是一回事**，一个典型用户可能在使用系统时扮演一系列不同的角色，而一个参与者表示的是一类外部实体，仅扮演一种角色。
* **动作**是系统与角色通信的一次执行，或进行一次计算。

**9.9.2 领域分析**

面向对象领域分析的**目标**是发现或创建可广泛应用的**类**，使它们可以被复用，在某个特定的应用领域中，公共的、可复用的标识、分析和规约，表现形式为**公共的对象、类、部件和框架**。



**9.9.3 类和对象建模**

1、标识分析模型中的**类和对象 🡨** 用例

2、定义**对象之间**的结构和层次

3、定义**主题**或**子系统**（各个子集相互协作完成一个**内聚的功能**）

**9.9.4 建立对象-关系模型**

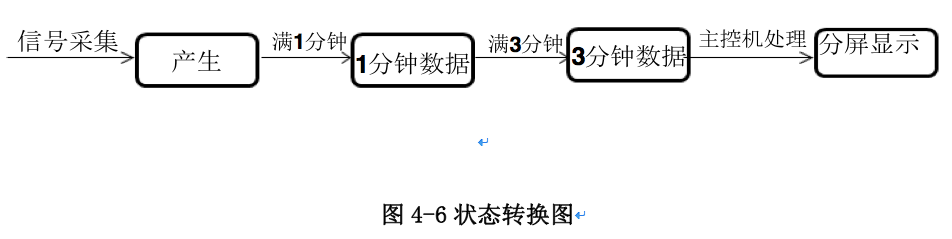
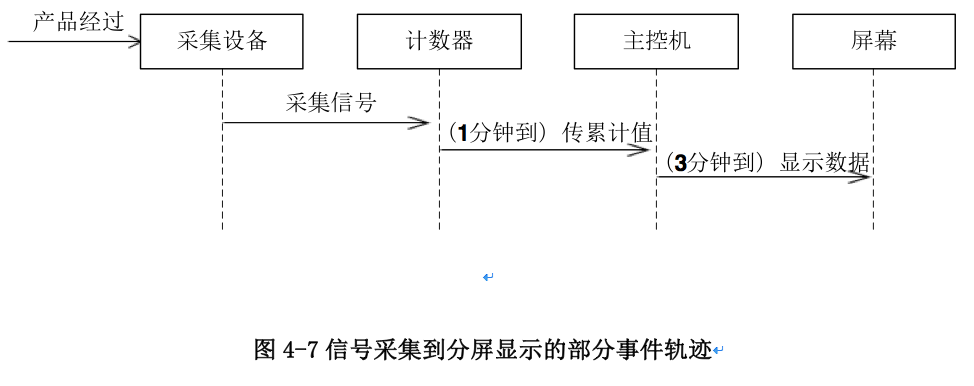
1、分析类之间的关系，**静态描述**类之间的联系

2、经常通过**类的属性**来表示类对另一个类的依赖关系

**9.9.5 建立对象-行为模型**

描述类之间的**动态行为**，即：系统如何应对外部事件。

信号采集到分屏显示的部分事件轨迹



状态转换图

**10.1 系统设计的任务**

**系统设计的任务**是在明确用户需求后，为满足这些需求而**设计具体的功能**。基本任务包含**体系结构设计**、**模块设计**、**数据结构与算法设计**、**用户界面设计**4个方面。

系统设计是一个**迭代过程**，通过系统设计将用户的需求变为实现软件的蓝图，最初蓝图只描述软件的整体框架，随着设计活动的进行，不断对软件描述进行细化，最终形成一个可实施的的软件设计文件。

**系统设计方法可分为：**

传统软件设计方法

面向对象的软件设计方法

**10.2 结构化设计方法**

软件设计是一个把需求分析变换成软件表示的过程，它分为**概要设计**和**详细设计**两个阶段。

**概要设计阶段**：包括体系结构的设计、接口设计、数据设计和过程设计等内容。

**详细设计阶段**：确定各个软件组件的数据结构和操作，产生描述各软件组件的详细设计文档

**10.2.1 概要设计的任务**

**体系结构设计**

定义软件主要**组成部件之间的关系**