第 1 章　基础知识

学习目标

- 掌握数据库及数据库系统的基本概念。
- 了解数据库体系结构的概念。
- 了解数据库技术发展的几个历史阶段。
- 掌握结构化系统开发方法的基本思想和基本原则。
- 掌握结构化系统开发方法的生命周期。
- 了解系统开发的原型方法。

1.1　引　言

一个应用系统往往需要处理很多数据，例如一个饭店管理系统需要管理旅客的预订、人住、消费和结账，就要与旅客的基本资料、预订或入住的客房资料、消费的商品或服务的资料等打交道。旅客的基本资料包括旅客姓名、身份证号码、籍贯、性别、工作单位、住址、预订或入住的客房号码等，客房资料包括客房号码、参考单价、被预订情况、已入住情况、可用状态等，消费的商品或服务的资料包括商品或服务名称、数量、价格、消费时间、旅客房号、消费金额、付账方式等，这些资料就是数据。在系统中对大量数据的管理通常是通过数据库实现的，这样的应用系统就是数据库应用系统。

要开发一个数据库应用系统，首先需要知道这个系统要做些什么事，做这些事要处理哪些数据，如何处理，然后在软件系统中将其实现。例如要开发一个饭店管理系统，要知道饭店有哪些部门，这些部门的职责是什么，各部门设置了哪些岗位，这些岗位的责任是什么，饭店的业务有哪些，每项业务涉及哪些部门和岗位，每项业务是如何进行的，需要哪些数据，产生了哪些数据，等等，即对系统的需求和涉及的数据进行调研，这就是需求调研工作。然后进行分析，判断哪些业务可以通过软件系统实现，如对旅客预订、入住和结账的管理，对客房各种状态的管理，对职工的管理等；哪些业务目前不具备实现的条件，如未与银行联网的饭店必须在与银行联网后才能实现自动刷卡结账等；哪些业务不能或不宜或无须通过软件系统实现，如行李服务业务等。还要对涉及的数据进行分析，这些数据在现实中往往以表格、记录和报表的形式出现，分析时应明确其类型、来源和处理方法等，这就是需求分析和数据分析。需求调研、需求分析和数据分析也统称为需求分析。

在需求分析的基础上，确定组成系统的物理元素——程序、文件、数据库、工作过程和文档等。由于整个系统要实现的功能很多，不能一下子实现，所以可以把系统分解成多个模块逐个解决，如果一个模块的功能还很多，则可以进一步分解。例如饭店管理系统可以分解成总经理办公室子系统、前厅部子系统、客房部子系统、餐饮部子系统等。前厅部子系统可以分解成旅客预订模块、旅客入住模块、旅客结账模块等模块。旅客入住模块还可以进一步分解成散客人住模块、团体旅客入住模块、VIP 入住模块。散客入住模块可以再进一步分解成身份证检验模块、入住登记模块和收银模块等。而入住登记模块和收银模块又可以由散客人住模块、团体旅客入住模块、VIP 入住模块共用，收银模块还可以由各种与旅客消费有关的模块和与旅客结账有关的模块共用。确定组成系统的物理元素以及系统是由哪些模块组成的，以及这些模块相互间的关系是总体设计的主要任务。

总体设计确定了每个模块的外部特征——功能和界面，即模块做什么和模块的输入和输出。确定每个模块的内部特征——模块内部的执行过程，即每个模块的功能怎样去实现，这就是详细设计。详细设计为编程制定出一个周密的计划，然后才能比较顺利地过渡到编程阶段。

上述开发应用系统的方法就是结构化系统开发方法的部分步骤。除了结构化系统开发方法外，还有原型方法和面向对象方法等开发方法。

在本章中，对数据库和系统开发方法的基本知识作了简要的介绍。

1.2 数据库技术

数据库技术是数据管理的技术，近年来发展迅速。目前，各领域对数据管理的需求越来越多，各行各业的信息系统都离不开数据库的支持。可以说，数据库已成为信息社会的重要基础设施，数据库技术成为实现和优化信息系统的基本技术。

1.2.1 数据库的基本概念

1. 数据库

数据库(Data Base,简称 DB)是按一定的组织形式存储在一起的相互关联的数据集合。实际上，数据库就是一个存放大量业务数据的场所，其中的数据具有特定的组织结构。所谓“组织结构”，是指数据库中的数据不是分散的、孤立的，而是按照某种数据模型组织起来的，不仅数据记录内的数据之间是彼此相关的，数据记录之间在结构上也是有机地联系在一起的。数据库具有数据的结构化、独立性、共享性、冗余量小、安全性、完整性和并发控制等基本特点。

2. 数据库管理系统

数据库管理系统（Data Base Management Systems,简称 DBMS）是数据库系统的核心部分，它是在特定的操作系统支持下帮助用户建立、使用和管理数据库的一种计算机软件。DBMS 提供了许多命令、函数和语句让用户对数据库中的数据资源进行管理操作（如数据库文件的建立、数据的输入输出、增加、删除、浏览、查询、修改、统计、分类、连接等）。总之,数据库的一切操作都是通过数据库管理系统来实现的。

3. 数据库系统

数据库系统（Data Base System，简写为 DBS）是指计算机系统引入数据库后的系统构成，是一个具有管理数据库功能的计算机软硬件综合系统。具体地说，它主要包括计算机硬件、操作系统、数据库（DB）、数据库管理系统（DBMS）和建立在该数据库之上的相关软件、数据库管理员和用户等组成部分。数据库系统具有数据的结构化、共享性、独立性、可控冗余度以及数据的安全性、完整性和并发控制等特点。

4. 数据库系统的体系结构

从数据库管理系统角度看，数据库系统是一个三级模式结构；从最终用户角度看，数据库系统分为单用户结构、主从式结构、分布式结构、客户／服务器结构和浏览器／服务器结构。

（1）单用户结构的数据库系统

单用户数据库系统是一种早期的最简单的数据库系统。在单用户系统中，整个数据库系统，包括应用程序、DBMS、数据，都装在一台计算机上，由一个用户独占，不同机器之间不能共享数据。

（2）主从式结构的数据库系统

主从式结构是指一个主机带有多个终端的多用户结构。在这种结构中，数据库系统，包括应用程序、DBMS、数据等都集中存放在主机上，所有处理任务都由主机来完成。各个用户通过主机的终端并发地存取数据，共享数据库中的数据资源。

（3）分布式结构的数据库系统

分布式结构是指数据库中的数据在逻辑上是一个整体，但物理地分布在计算机网络的不同结点上。网络中的每个结点都可以独立处理本地数据库中的数据，执行局部应用，同时也可以同时存取和处理多个异地数据库中的数据，执行全局应用。

（4）客户／服务器结构的数据库系统

在客户／服务器结构中，把 DBMS 功能与应用分开，网络某个结点上的计算机专门用于执行 DBMS 功能，称为数据库服务器，简称服务器，而其他结点上的计算机则安装 DBMS 的外围应用开发工具，支持用户的应用，称为客户机。

（5）浏览器／服务器结构的数据库系统

浏览器／服务器结构的数据库系统与客户／服务器结构数据库系统类似，把 DBMS 功能与应用分开，网络某个结点上的计算机专门用于执行 DBMS 功能，称为数据库服务器，简称服务器，不同的是其他结点上的计算机只需浏览器，即可支持用户的应用。

1.2.2 数据库技术的发展概述

数据处理的核心问题是数据管理。所谓数据管理，是指对数据进行组织、编码、分类、存储、检索与维护等操作。数据管理技术随着计算机硬件和软件的发展而发展。从数据管理的角度看，到目前为止，数据管理技术主要经历了人工管理阶段、文件系统阶段和数据库系统阶段。

1. 人工管理阶段

人工管理阶段是指计算机诞生的初期（1946 年—20 世纪 50 年代中期），这一时期的计算机主要用于科学计算。数据管理的特点是：使用的数据不保存，用完就拿走；没有软件系统对数据进行管理，而是由人工规定数据逻辑结构和存取方法；没有文件概念，用户负责数据的组织方式

时，必须考虑数据存取细节；数据和程序一一对应，不同的应用程序之间不能共享数据。

2. 文件系统阶段

随着计算机技术的发展，计算机的应用范围逐渐扩大，计算机不仅用于科学计算，而且已大量用于数据处理、事务管理、工业控制等领域。这个时期（20 世纪 50 年代中期—60 年代中期）数据管理的特点是：数据可长期保存在外存设备上，文件可以被反复多次地进行查询、添加、删除和修改等操作；有统一的文件管理系统。用户按照统一的方式建立和存取文件，不用去考虑数据的物理存储位置和具体外部设备的物理特性。用户只需集中精力于算法和数据的逻辑组织结构，从而大大提高了数据管理的效率和准确性。

3. 数据库系统阶段

数据库系统阶段是从 20 世纪 60 年代后期开始的。在这个阶段，计算机用于管理的规模更加庞大，应用越来越广泛，文件系统的数据管理方法已无法适应开发应用系统的需要。为解决数据的独立性问题，实现数据的统一管理和共享，于是发展了一种新的数据管理技术——数据库技术。数据库技术的基本特征之一是相互关联的数据的集合，它用综合的方法组织数据，具有较小的数据冗余，可供多个用户共享。它具有较高的数据独立性和安全控制机制，能够保证数据的安全性、可靠性以及一致性和完整性，并允许并发地使用数据库，及时、有效地处理数据。数据库技术主要目的是有效地管理和存取大量的数据资源，把数据集中存放在一个或多个数据库中，用户通过数据库管理系统来使用数据库中的数据。数据库技术作为数据管理的最有效的手段，它的出现极大地促进了计算机应用的发展，目前基于数据库技术的计算机应用已成为计算机应用的主流。数据库系统的出现使信息系统的研制从以加工数据的程序为中心转向围绕共享的数据库来进行。

数据库技术发展到今天已经比较成熟。到目前为止，数据库技术已从第一代的网状、层次数据库，第二代的关系数据库系统，发展到第三代以面向对象模型为主要特征的数据库系统。数据库技术与网络通信技术、人工智能技术、面向对象程序设计技术、并行计算技术等互相渗透，互相结合，成为当前数据库技术发展的主要特征。数据库技术与其他技术相结合而产生的各种新型数据库如图 1.2.1 所示。

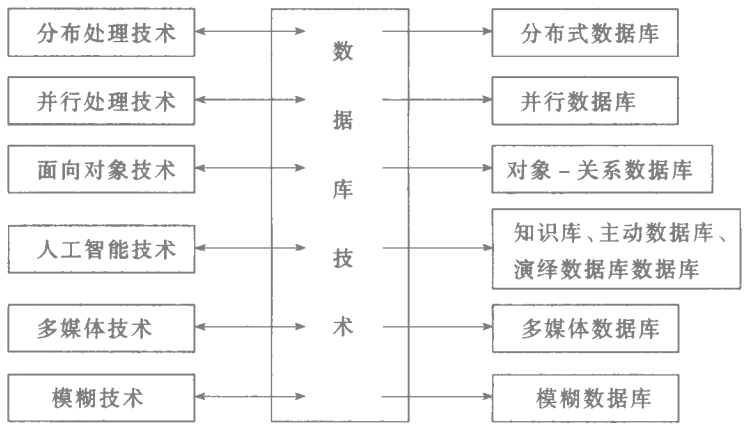

图 1.2.1　数据库技术与其他计算机技术的相互渗透

*1.2.3 面向对象的数据库

面向对象的程序设计（Object Oriented Programming，简称 OOP）是组织大型软件的研制、开发、维护及管理的有效方法。现在，许多软件都使用了面向对象的技术。面向对象的数据库就是面向对象思想和数据库技术相结合的产物。

1. 对象的实质与特点

面向对象技术的基本概念有对象、实例和类、消息与继承等。在面向对象的程序设计语言中，只有一种类型的实体——对象。对象可同时表示为叙述性知识和过程性知识。因此，一个软件就是各种不同对象的集合。一个对象就是一个基本模块，它有自己的一些数据和操作这些数据的过程，即数据和过程是封装在一起的。对象具有自主性、封闭性、交互性、被动性和动态性 5 大特征。自主性、封闭性、交互性描述了对象的能力，被动性刻画了对象的活动特性，动态性指出了对象的生存特性。

这里要特别指出对象具有封装性和继承性两个主要特点。封装性是指对象用操作集来描述可见的接口。这个特点保证了对象的界面是独立于对象的内部表达。对象操作的实现以及对象和结构都是不可见的。为了强调对象的独立性，对象的通信用消息传递来实现。一个对象属于一个类，而类也可以处理成对象。每个类都有特殊的操作方法用来产生新的对象。

继承性是对象的另一个特点。继承可以用来定义彼此十分相似的那些对象的类。例如可以说明一个类（子类）是另一个类（超类的）派生，这样可以很方便地管理类之间的层次关系，而软件也就有了可重用性。子类对象继承了超类对象的结构和操作，而且在子类中还可以不管原有的方法和结构，从而形成了一种代码共享的手段。

2. 面向对象数据库系统的基本特性

面向对象技术在数据库中的应用体现在数据库管理系统和应用开发工具两个方面，即面向对象数据库和面向对象的数据库应用开发工具。

面向对象数据库系统应具备以下特征：

（1）必备特性

该特性是面向对象的数据库系统所必须满足的特征：包括复杂对象、对象标识、封装性、类型或类、继承性、计算完备性、可扩充性、持久性、并发性，恢复和即时查询功能。一个面向对象的数据库系统必须是一个数据库管理系统，同时还必须是一个面向对象的系统，在一个可能的范围内，要与当前流行的面向对象的程序设计语言一致。

（2）可选性

多重继承性、类型检查、类型推理、分布、设计事务处理和版本可作为面向对象数据库系统的可选特征。其中，多重继承是指一个类可以有多个父类，可以从多个父类那里继承；分布是指数据库系统既可以是分布的，也可以不是分布的；大多数新型应用都包含设计活动，需要某种形式的版本控制。

（3）开放的可选特性

设计人员可选择的特征有程序设计范型、表示系统、类型系统及单一性。

*1.2.4 Web 数据库

随着 Internet 的快速发展，数据库技术的发展及应用领域将大大拓展。用户可通过 Internet 直接访问远程的数据库服务器，也可通过 Web 服务器或中间服务器访问数据库。当然，网上的数据库需要支持 HTML(超文本链接标记语言)和 XML(可扩展标记语言），以便可将存储在数据库中的数据生成 XML 格式的文本,或将 XML 文本和数据存储到数据库中。

Web 数据库是一个新的研究领域，它是 Web 和数据库技术相结合的产物。Web 数据库应用系统一般由 Web 服务器、服务器组件、数据库服务器和浏览器所构成。Web 数据库技术是指用 Web 浏览器界面来存取数据库内容。用浏览器访问网上数据库的体系结构如图 1.2.2 所示。

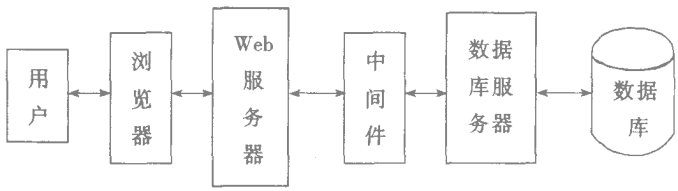

图 1.2.2 浏览器访问网上数据库的体系结构

1.3 系统开发方法

1.3.1 结构化系统开发方法

1. 结构化系统开发方法的基本思想

结构化系统开发方法的基本思想是：用系统工程的思想和工程化的方法，按用户至上原则，用结构化、模块化、自顶向下的方法对系统进行分析与设计。

为了保证系统开发的顺利进行，采用结构化系统开发方法时应遵循以下几个基本原则：

(1) 面向用户的观点

用户需求是应用系统开发的出发点和归宿。应用系统是直接为用户服务的，因此，在系统开发的全过程中要从用户的利益出发，系统开发的主要环节都要有用户单位的管理人员和业务人员参加。系统开发人员始终与用户保持良好的关系，及时交流意见，讨论开发中的问题，以便统一认识，加速开发进度，保证系统质量，以满足用户的需求。

(2) 严格区分工作阶段

系统开发过程划分为若干阶段，每个阶段都有其明确的任务和目标，应该取得相应的成果。例如，将应用系统开发过程划分为系统规划、系统需求分析、系统设计、系统实施、系统维护与评价等阶段，每个阶段又分为若干工作和步骤。这种有序的安排，不仅条理清楚，便于制定计划和

控制进度，而且后面阶段的工作又是以前面阶段工作的成果为依据，可避免重复和返工。

（3）按系统的观点，自顶向下完成系统的开发工作

在系统需求分析阶段，按照全局的观点，自上而下，从粗到精，由表及里，将系统逐层逐级进行分解。在保证总体模块功能的前提下，逐步分层细化，将总体模块划分为适当的模块，完成模块结构设计，在这些模块的基础上进行物理设计和程序设计。

（4）充分预料变化的情况

由于应用系统的环境总是在不断变化之中，因此，用户对系统的要求也总在不断变化，结构化系统方法要考虑这种变化的情况。在系统设计中，要把系统的可变性放在首位，并运用模块方式来组织系统，使系统具有灵活性和变更性。

（5）工作文件的标准化、文献化

系统开发是一项复杂的系统工作，涉及的范围大。参加的人员多，周期也较长。为了保证开发工作的连续性，开发过程中所有的工作内容、各种资料、开发阶段的成果都以文字、图表的方式，按标准格式记录，形成文献，使系统开发人员及用户有共同语言。所有文献资料按标准化要求保证定义的明确性、无二义性，使开发人员阅读方便，用户理解容易。文献资料要编号存档，妥善保存，便于今后查阅。

2. 系统开发的生命周期

结构化系统开发方法，将整个开发过程划分为五个时序相连接的阶段，即系统开发的生命周期。

系统开发生命周期各阶段的主要工作有：

（1）系统规划阶段

系统规划阶段是根据用户的系统开发请求，进行初步调查，明确问题，确定新系统目标和总体结构，确定分段实施进度，进行可行性研究，形成可行性报告。

（2）系统需求分析阶段

系统需求分析阶段的任务是：对现行系统进行详细调查，分析业务流程，分析数据与数据流程，分析功能与数据之间关系。指出现行系统存在的问题和不足之处，确定新系统的基本目标和逻辑功能要求，最后提出分析处理方式和新系统的逻辑模型，这个阶段又称为逻辑设计阶段。逻辑设计解决系统“做什么”。因此，这个阶段是整个系统建设的关键阶段。

系统需求分析阶段的工作成果为系统说明书，这是系统建设的必备文件。系统说明书既要准确又要通俗易懂，用户根据系统说明书可以了解未来系统的功能，判断是不是他们所要求的系统。

系统说明书一经通过，就是系统设计的依据，也是将来评价和验收系统的依据。

（3）系统设计阶段

系统需求分析阶段的任务概括地讲，已解决了系统“做什么”的问题，系统设计阶段要回答的问题则是系统“怎么做”，也就是说，根据系统设计说明书所规定的功能要求，考虑“实际情况，具体设计实现逻辑模型的技术方案，即新系统的物理模型。这个阶段也称为物理设计阶段。这个阶段又可分成总体设计和详细设计两个阶段。

这个阶段的技术文档为系统设计说明书。

（4）系统实施阶段

系统实施阶段是按物理设计的设计方案付诸于系统实现的具体工作。这一阶段的任务是：计算机等设备的购置、安装和调试，编写程序和调试程序，人员培训，数据文件转换，系统调试与转换等。

这个阶段工作量大，互相联系、互相制约的任务同时展开，必须精心安排，周密计划，合理组织，统筹调度和协调，以保证系统开发的顺利进行。

实施阶段是按实施计划分阶段完成的，每个阶段的工作应写出实施进度报告。系统测试之后应写出系统测试分析报告。

（5）系统运行阶段

实施投入运行后，需要进行经常性维护和评价，记录系统运行的情况，根据一定的程序对系统进行必要的修改，评价系统的工作质量和经济效益。

有时也把系统运行阶段称为系统维护与评价阶段。

数据库应用系统开发过程中的成果及审核如图 1.3.1 所示。

3. 结构化开发方法的优缺点

结构化系统开发方法突出的优点是，它强调系统开发过程的整体性和全局性，强调以整体优化为前提，按自顶向下的观点考虑具体的分析设计问题。强调严格地划分阶段，按步骤严格地进行系统需求分析和设计，每一步工作都及时地总结，发现问题并及时反馈和纠正。每一个阶段的工作成果都要进行评审，前一阶段的工作审核不通过，不能进行后面阶段的开发。这样就避免了开发过程中的混乱无序状态，减少损失和返工，这是一种曾广泛采用的开发方法。

但是随着时间的推移和技术的进步，这种方法逐渐暴露了它的缺点和不足。最突出的表现是：

（1）它的起点较低，使用的工具（人工绘制的分析图表）落后；

（2）开发周期较长，在开发过程实施中，系统的情况可能发生较大的变化；

（3）系统开发者在分析设计中就要能掌握用户的需求、管理状况以及预见可能发生的变化，设计开发出适应实际的管理信息系统，这是很难做到的。

（4）随着系统规模及复杂性的不断增大，结构化开发方法在具体实施过程中存在相当大的难度。因此结构化的程序设计思想逐渐被面向对象的程序设计所替代。

1.3.2 原型方法

原型方法是 20 世纪 80 年代随着软件技术的发展，尤其是在关系数据库系统（Relational Data Base Systems RDBS）、第 4 代程序生成语言（4th Generation Language 4GL）和开发生成环境产生的基础上，提出的一种从设计思想、手段、工具都全新的系统开发方法。与结构化系统开发方法相比，它克服了结构化系统分析方法起点较低的弱点，不再需要一步步周密细致地调查分析，并逐步整理出文字档案。原型方法一开始就凭借系统开发人员对用户要求的理解，在强有力的软件环境支持下，先构造一个原型（模型），使用户尽早看到未来系统的概貌，在原型系统运行中发现问题，与用户协商，提出改进意见，再去完善原型，使它满足用户的要求。

1. 原型方法的工作流程

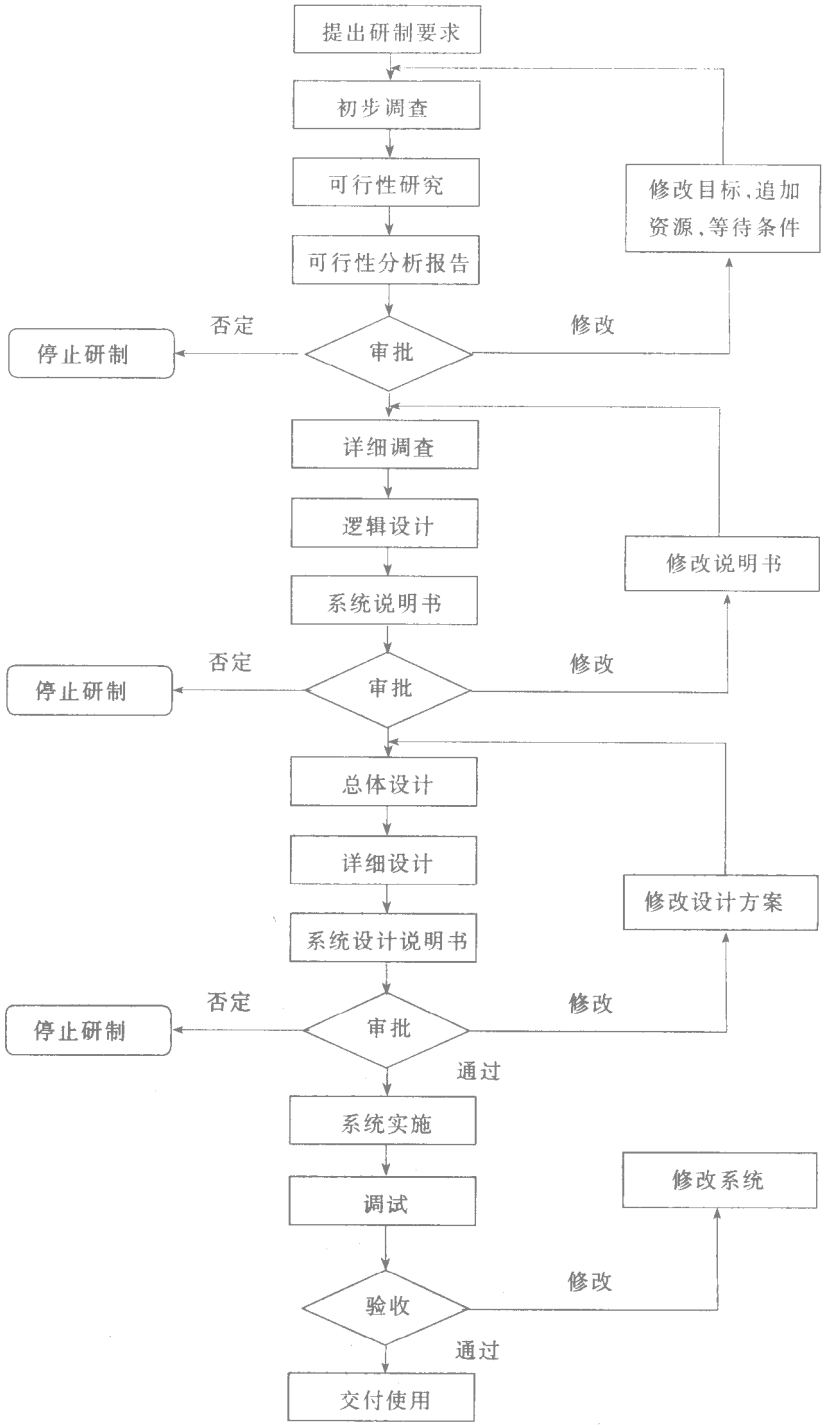

图 1.3.1　数据库应用系统开发过程中的成果及审核

原型方法的工作流程如图 1.3.2 所示。首先用户提出开发要求，系统开发人员识别和归纳用户要求，根据用户的要求及功能要求的数据规范、报告格式、屏幕要求等，构造出一个原型（程序模块），再与用户共同评价这个原型。如果根本不可行，则重新构造原型；如果不满意，则修改原型，直到用户满意为止，这是一个反复的迭代过程。

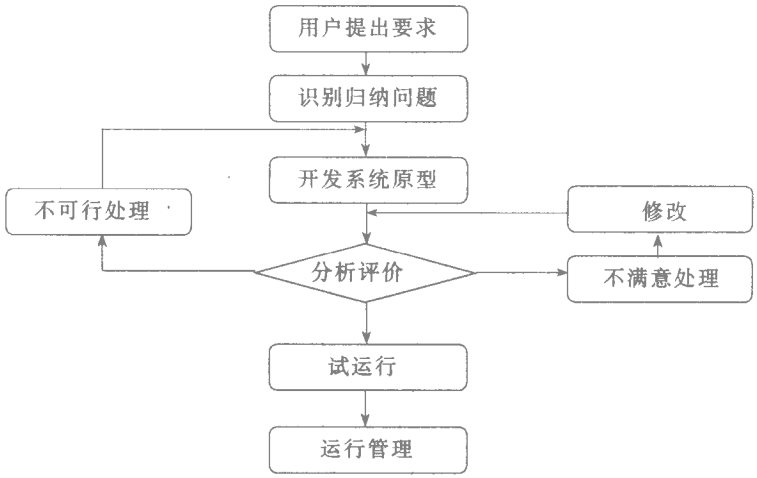

图 1.3.2　原型法工作流程

2. 原型方法的特点

原型方法从原理到流程都很简洁，并不需要深奥的理论和技术，因而受到软件系统开发者的推崇。与结构化系统方法相比，原型方法有如下特点：

（1）尽可能利用现成软件和模型来构造原型。采用开发工具，可以缩短系统开发周期，节省费用，提高效率。

（2）初始系统要能反映用户的基本要求，缩短用户与系统分析人员之间的距离，结构化系统开发方法较难解决这一环节。主要表现在：

1）所有问题的讨论都是围绕原型而进行的，彼此之间减少了误解和答非所问的可能性，为准确认识问题创造了条件。

2）有了原型之后，启发人们对原来想不出或不能准确描述的问题有一个比较确切的描述。

3）能够及早暴露出系统实现后存在的问题，促使开发人员在系统实现之前就及时解决。

4）从认识论角度看，原型方法遵循了认识事物的规律——循序渐进。

原型方法不如结构化系统方法成熟和便于管理控制。实现原型方法的关键是能否提供一个合适的软件开发环境和一套高级的软件工具，能否根据需求说明转换成现实系统。还需要有用户能接受的用户界面和自动转换的工具，这些都还很不完善。

3. 结构化开发与原型化方法结合

结构化开发本质上是一类严格定义的方法，对于一类较难预先定义的问题，把结构化方法与原型化方法结合，可能会出现新的面貌。

实际上在构造原型时，也要按照类似结构化系统开发方法的步骤进行，只不过需求分析比较

粗糙一些，对阶段性文档的要求不那么严格。而结构化系统开发方法中，为加强与用户的沟通，也可以使用一些原型说明问题。

结构化开发与原型化方法结合如图 1.3.3 所示。

因而将原型化方法与结构化方法合为一体的尝试是有益的，而且对于需求分析方面原型化方法要高于结构化开发方法，当然不排除从原型直接过渡到产品的可能，那将是最理想的结果。这些都需要在实践中尝试。

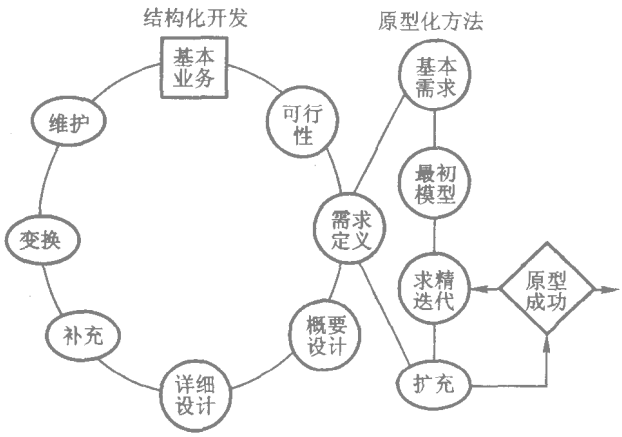

图 1.3.3 结构化开发与原型化方法结合

*1.3.3 面向对象方法

系统开发有许多基本方法，在不同发展阶段中产生，适应了当时的需要，尤其是结构化方法，得到广泛的应用。但是，随着用户需求的日益复杂，人们必然会发掘出更好的方法来解决这个矛盾，这就是面向对象（Object Oriented，OO）方法。面向对象方法作为一种新颖的、具有独特优越性的方法引起人们的关注，把面向对象方法视为解决软件危机的突破口。面向对象方法被扩展到各个领域，如面向对象的体系结构、面向对象的硬件支持、面向对象的软件开发环境、面向对象数据库、面向对象程序设计语言等。

面向对象方法作为一种认识方法论，强调了对现实世界的理解和模拟，把现实世界到计算机实现的转换减到最少，所以面向对象方法特别适合于系统分析和设计。

1. 面向对象方法的基本思想

面向对象方法认为，客观世界是由各种各样的对象组成。每种对象都有各自的内部形态和运动规律，不同对象之间相互作用和联系构成了各种各样的系统。如果能在满足需求的前提下，把系统设计成由一些不可变的（相对固定）部分组成的最小集合，这个设计就是最优良的。这些不可变的部分就是对象。

对象是面向对象方法的主体。广义地讲，对象可以是任何人或事物。精确地讲，对象是一些属性及专用服务的封装体，它是问题空间中一些东西的抽象。对象就是我们在问题空间中要考虑的那些人、事、物，它具有一组属性和一组操作，这些属性的值刻画了一个对象的状态。这些操

作是对象的行为，通过它们改变对象的状态（即属性值）。

面向对象方法具有如下特征：

（1）抽象性

抽象是对复杂世界的简明表示。对象具有极强的抽象表达能力，既可表达结构化数据，也可表达非结构化数据，如图形、声音、复杂的规则等，因此面向对象方法具有很强的建模能力。将数据结构上的抽象与功能上的抽象结合起来，统一于对象之中，实现更高的抽象。

（2）封装性

封装即信息隐藏。当开发一个总体程序时，程序的每个成分应该封装或隐蔽一个单一模块。所定义的每一个模块应尽可能不显露其内部处理。封装性指保护软件的内部实现，只给用户提供良好的界面，用户不必了解其内部实现细节。对象是封装良好的模块，这种模块更易于重用。对象的界面即是它的一组属性和控制这些属性状态的一组操作。

（3）继承性

继承性体现了面向对象方法的共享机制。对象共享它所在类的结构、操作和约束等。在类层次中，子类可继承父类的全部语义特性，继承简化对象、对象类的定义和创建，是信息组织与分类的有效方法。

（4）多态性（多形性）

多态性指相同的操作（函数，过程）可作用于多种类型的对象并获得不同的结果。给不同类型的对象发送相同的消息，不同的对象会分别作出不同的处理。例如，发送一个做“加法运算”的消息，整数对象接收此消息后做整数加法，复数对象做复数加法，产生的结果显然是不同的。其多变性增强了软件的灵活性、重用性和可理解性。

2. 面向对象分析方法（OOA）

面向对象分析方法，即 OOA 方法，要求把问题空间分解成一些类或对象，找出这些对象的特点（属性和服务），以及对象间的关系（一般与特殊、整体与部分等），并由此产生一个规格说明。

在 OOA 中，直接从问题空间映射到模型，对象抽象了问题空间中的因素，使我们对问题空间的理解更直接、准确、快捷，减少了语义差异和转换，而另一个相似的项目可以重用以前的分析结果。

人类构造空间问题的三个基本法则分别是：对象及其属性；整体及其部分；类属及其成员。有了这个共同法则，分析人员与用户之间的交流就变得准确无误了。

（1）提出 OOA 的原因，可归纳为：

1）它使我们能在人类概括客观事物的三个基本方法（对象及其属性，分类结构和组装结构）的框架上定义和交流系统要求。

2）它将属性及专用那些属性的服务视为一个固定的整体。

3）使用自包含分块（对象之间依赖性最小）进行分析和说明。

4）它首先着眼于问题空间的理解。

5）它提供一个支持分析（做什么）和设计（怎么做）的一致性表示工具。

6）它能适应分列化的分流和当前实际运行分流的调整。

（2）OOA 作为完整的方法，由以下步骤构成：

1）标识对象。对象是一组数据及其操作的描述，反映了系统保存有关信息和与现实世界交

互的能力，对象是稳定的部分，标识对象能产生一个稳定的框架模型，避免在分析设计时改变系统的基本表示。

定义对象包括寻找对象、挑剔对象、命名对象。

①寻找对象。寻找对象需要观察问题空间，同用户讨论，以及阅读资料，重视必须记住、必须服务的事物。对象具有较强的独立自治性，它是保持完整信息并与外界事物交互作用的单位。

②挑剔对象。那些不必记住的事物与服务、单个属性或派生结果（如“年龄”是“出生日期”的派生结果）可能并不是对象。

③命名对象。用单数名词或“形容词＋名词”作对象名，尽量选择反映主题的标准词汇作对象名。

④符号表示。用对象名来命名属性和服务，其符号表示一律按标准的画法，并使用相应的符号。

2）标识结构。结构分为分类结构和组装结构。

①分类结构。分类结构指“一般—特殊”结构，与“类属及其成员”相对应。

分类结构有助于描绘问题空间的类一成员层次，通过搜集公共特性，并把这种特性扩充到特例之中来显示现实事件的通用性及专用性。

继承的概念是分类结构的重要部分，继承提供了一个用于标识和表示公共属性服务的显示方法。分类结构中，继承使共享属性或共享服务、增加属性或增加服务成为可能。

定义分类结构时，要分析在问题空间和系统责任的范围内，通用类是否表达了专用类的共性，专用类是否表示了个性。子类把不需继承的服务和属性另用标记“ X ”以示差别。

②组装结构。组装结构对应“整体一部分”组织方法。

组装结构描述了一个整体及其组成部分。例如，一辆汽车由发动机、传动装置和刹车装置等组成。定义组装结构时，要明确以下问题：

系统需要跟踪它的一部分的每一个实例吗？它的一部分的每个实例能用属性描述吗？该部分在所考虑的系统范围之内吗？该部分反映的是现实世界的部分属性吗？同样，考虑整体时也一样。

③标识主题。主题给出了 OOA 模型中各图的概貌，使读者在更高层次上观察模型全貌。为读者在一定时间内所考虑的模块数提供了一个控制机制。

主题就是对 OOA 模型的提炼。

使用主题的原因是，实际系统有大量的对象和结构，人们在一定时间内的短期记忆是有限的，因此，必须控制一次见到的模块数，才能更好地理解系统模型。

主题可以通过以下方式定义：

a. 标识主题的基础是用分类结构和组装结构所标识的问题空间的结构。

b. 为每一个对象和每一个结构增加一个相应主题，如果主题的个数超过 7 个，则要再提炼主题。对象和结构之间的连接得到标识后，根据需要，把紧耦合的主题合在一起提供一个更好的模型概貌给读者。

c. 列出主题及主题层上各主题之间的消息连接。

d. 对主题进行编号，在层次图上列出主题，指导读者从主题到主题，每一层都可以组织成按主题划分的图。

④定义属性。对象通过属性来描述。属性在对象库中进一步说明，使系统模型更明确、详细。问题空间中对象相当稳定，属性却较容易改变，属性由该对象的服务专门操作。

定义属性的步骤是：

a. 标识属性——寻找并表示属性。

b. 属性定位——通用属性应放在结构的高层，特殊性应放在低层。如果某个属性适用于大多数的特殊分类，可将其置于通用的地方，在不需要的地方将其覆盖（用“ X ”等记号指出不需要继承此属性），如果发现某个属性的值有意义，但有时却不适用，则应考虑分类结构。根据发现的属性，可进一步修订对象。

c. 标识实例连接——实例连接就是一个实例与另一个实例的映射关系，它表示了两个实例间的联系，每一条实例连接都意味着一条相对应的消息连接线。

实例连接有 1 对 1 或 1 对多，即(1:1)或(1:m)，分别表示实例可对应一个或多个实例，这叫多重性。例如，1 个车主有 1 辆汽车，则车主到汽车的实例连接为 1:1；1 个车主有多辆汽车则为 1:m。若 1 辆汽车被多个车主共有，则汽车到车主实例连接也是 1:m。

实例连接还有一种性质叫参与性，表示对象间实例连接是强制性的还是任意性的，即一个对象不存在时，另一个对象是否仍有意义，或者说这种连接是否必须存在。例如，“居民”到“登记和发身份证”之间的实例连接是任意性的，因为当“登记和发身份证”没有实例时，“居民”实例仍然有意义 反之，当‘居民’实例不存在时，‘登记和发身份证’也没有意义，则‘登记和发身份证”与“居民”的连接是强制性的。

d. 说明属性和实例连接的约束——用名字和描述说明属性，属性有四类：描述性的、定义性的、永远可导出的和偶尔可导出的。

⑤定义服务。分析员最终必须提供系统处理和流程需求的详细描述，由于这一部分很不稳定，但又非常重要，因此在考虑了对象、结构、属性（及实例连接）之后，才进行服务和消息连接的工作。

服务是这样定义的：服务是收到一条消息之后所执行的处理，服务定义了对象及类所需的行为。行为的分类包括直接动因的行为、进化史上的相似行为、功能相似的行为。

标识服务（即对象的行为）的策略分别对应以上所述的三类行为：

a. 直接动因对应：状态—事件—响应（辅助策略）；

b. 进化史对应；对象生命历程（辅助策略）；

c. 功能对应：最基本的服务（基本策略）。

在标识上述三类服务后；还要标识对象实例之间必要的通信，消息相当于发出命令或要求，用于实例间的交互。

标识服务的最后一环是详细说明服务和消息连接。

标识服务的基本策略，每个对象或分类、结构有三种基本服务：

a. Occur(实例的增加、修改、删除和选择)；

b. Calculate(计算)；

c. Monitor(监控)。

“Occur”服务是建立并维护对象或分类结构的实例。每一个对象或分类结构都需要“Occur’服务，只把它当作隐含的服务，不出现在 OOA 模型图中。

“Calculate ”服务为某个实例计算出结果。注意服务名要能说明主题，而不用“ Calculate”。

“Monitor ”服务执行对外界系统、设备或用户的运行监控。

标识服务的辅助策略，即对象生命历程和状态事件、响应对象生命历程，描绘一个对象或分类结构以后发生的事件，分为以下几步：

a. 定义基本的对象生命历程序列，如增加、修改、选择和删除；

b. 检查每一步的演变，即扩展每一步都要反映出增加、修改、删除的演变；

. . 增加基本序列，即对象或分类结构是否响应其他事件；

d. 增加服务，一般所增加的服务是对象基本服务的演变。

根据“ 状态一事件一响应 ”，这个策略也用于发现对象的附加服务，又分为以下几步：

a. 定义系统的主要状态；

b. 对于每一个状态，列出外部事件和所需要的响应；

c. 根据事件和响应增加服务和消息连接；

d. 辅助策略有助于发现可能遗漏的“ Calculate ”和“ Monitor ’服务。

e. 标识消息连接。

消息连接结合了事件响应和数据流这两者，一个消息连接既表示发出一条消息，又表示接收一个响应。消息连接将一个实例对应于另一个实例，发送者发送一条消息给接收者去完成一些处理，并给出需要的服务名，而接收者的服务说明中定义这个处理。

f. 考虑消息连接。

在已经存在实例连接的对象和分类结构之间增加消息连接。之后，检查对象和分类结构（包括封装在其中的属性）。找出一个实例所需要的另一个实例的服务，从中找到消息。然后，在发送者的服务说明中建立消息连接的文档，在接收者的服务说明中建立相应的执行服务的文档。

g. 详细说明服务。

服务的规格说明有：集中所需要的外部可见的行为；使用一个模板建立需求框架表：增加图示的简化说明；增加支持的表；建立服务的文字叙述；汇集所有的文档。

3. 面向对象设计方法（ OOD）

面向对象设计方法(OOD ）是面向对象方法中的一个过渡环节。其主要作用是对 OOA 分析的结果作进一步的规范化整理，以便能被 OOP 直接接受。

OOD 是对 OOA 产生的结果增添计算机实现的细节，包括人机交互、任务管理和数据管理的细节。OOA 到 OOD 是累进的模型扩充过程。

OOA 的各层模型化了‘ 空间问题 ”,而 OOD 则模型化了一个特定的“ 实现空间 ”。

面向对象方法运用到系统的根本问题是改进设计，增进软件生产效率，提高软件质量以及加强可维护性。OOD 可以通过从一个项目向另一个项目提供一些重用类的实际机制来提高生产效率。

改善软件质量不仅仅要求“ 无错误 ”，还要有更高的标准，如界面友好、可移植性好、易修改等，在开发过程中及开发之后都应如此。

OOD 的优点如下：

（1）OOA 加强了对问题空间的理解。 OOD 和 OOP 维持问题空间的定义。

(2) 改善问题空间专家、分析员、设计员、程序员之间的交流，OOD 运用在人的思维中普遍采用的组织方法来组织设计。

(3) 分析、设计和编程之间的内在一致性好，OOD 把属性和服务视为一个内在的整体，因此，缩小了各种不同活动间的距离。

(4) 明确表示共性，OOD 运用继承来识别和概括属性与服务的共性。

(5) 构造对变化具有弹性的系统，OOD 将系统结构中易变部分打包，对于变化的需求和相似的系统能提供稳定性。

(6) OOA、OOD、OOP 结果重用，既适合于系统序列，也适合于系统中的特殊交换。OOD 基于问题空间和实现空间约束，构造的结果支持重用和续后的重用。

(7) 提供一种对 OOA 做什么，OOD 怎么做和 OOP 相互一致的基本表述。OOD 提供连贯的表示，把 OOA 的结果系统地延伸到 OOD 和 OOP。

OOD 可分为 4 个部分：

1) 问题空间部分的设计。

2) 人机交互部分的设计。

3) 任务管理部分的设计。

4) 数据管理部分的设计。

当然，OOD 是一项十分周密，工作量也比较大的工程。在实际执行和设计过程中还有很多具体的计划和安排，这里不再多述。

*1.3.4 系统的生命周期模型

几十年来，软件系统开发生命周期的发展有了很大变化，提出了一系列的模型以适应软件系统开发发展的需要。

1. 瀑布模型

瀑布模型(Waterfall Model)是由 Winston Royce 在 1970 年提出来的。该模型规定了包括上述六个工程活动，并且规定了它们自上而下、相互衔接的固定次序，如同瀑布流水，逐级下落，如图 1.3.4 所示。

软件系统开发的实践表明，上述各项活动之间并非完全是自上而下，呈线性的。实际上，各项开发活动均具有：

(1) 接受上一项活动的工作成果，作为输入。

(2) 利用这一输入实施该项活动应完成的内容。

(3) 给出该项活动的工作成果，作为输出传给下一项开发活动。

(4) 对该项活动实施的工作进行评审。若其工作得到确认，则继续进行下一项活动，如图 1.3.5 中逆时针的箭头所示；否则就要返回前一项，甚至更前项的活动进行返工，如图 1.3.5 中顺时针的箭头所示。

在瀑布模型中，还存在着一些问题：

(1) 阶段与阶段划分被完全固定，每个阶段都会产生大量的文档。

(2) 由于开发主要呈线性，当开发结果尚未经过测试时，用户无法看到系统的效果。这样系

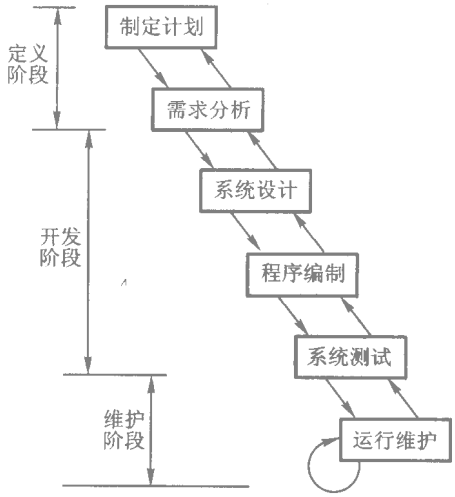

图 1.3.4 瀑布模型

统与用户见面的时间间隔较长。

(3) 前面未发现的错误传到后面的开发活动中时，可能会将错误进一步“放大”，进而造成更大的错误。

需要注意：软件系统维护在软件系统生命周期中有它的自身特点。一方面，维护的具体要求是在软件系统投入运行以后提出来的，经过“评价”，确定变更的必要性，才进入维护工作。另一方面，维护过程中对软件系统的变更仍然要经历上述软件系统生命周期，即开发中已经历过的各个阶段。将维护活动与软件系统本身的生命周期结合后，形成软件系统生命周期循环，如图 1.3.5 所示。

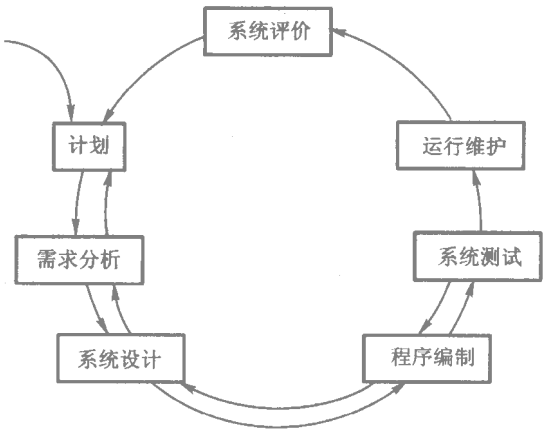

图 1.3.5 系统生命周期循环

由于系统在投入使用后可能要经历多次改变，为把开发活动与维护活动区别，便有了软件系统生命周期瀑布模型的变种，如图 1.3.6 所示。