第 1 章 计算机组成与体系结构

也许有人认为系统架构设计师不需要硬件或计算机底层原理的知识。因为这个层面的一些处理往往已经封装好，不需要架构师重新进行规划。然而，事实并非如此，系统构建 于硬件与操作系统之上，如果我们不对计算机底层原理有一定认识，会导致一系列安全与性能问题。本章将从计算机的组成，计算机的指令系统，存储系统等方面展开论述。

1.1 计算机系统组成

计算机系统是一个硬件和软件的综合体，可以把它看成按功能划分的多级层次结构。

1.1.1 计算机硬件的组成

硬件通常是指一切看得见，摸得到的设备实体。原始的冯•诺依曼（VonNeumann）计算机在结构上是以运算器为中心的，而发展到现在，已转向以存储器为中心了。图 1-1 所示为计算机最基本的组成框图。

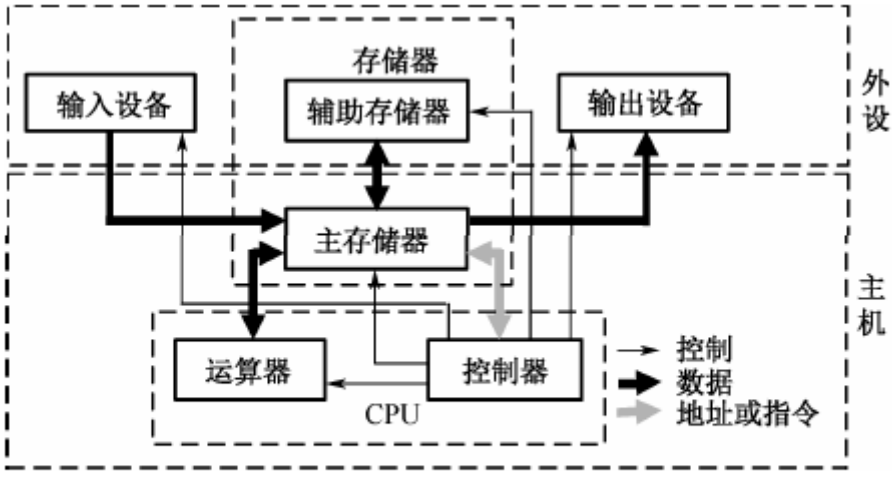


图 1-1 计算机各功能部件之间的合作关系

（1）控制器。控制器是分析和执行指令的部件，也是统一指挥并控制计算机各部件协调工作的中心部件，所依据的是机器指令。控制器的组成包含如下。

① 程序计数器 PC：存储下一条要执行指令的地址；

② 指令寄存器 IR：存储即将执行的指令；

③ 指令译码器 ID：对指令中的操作码字段进行分析解释；

④ 时序部件：提供时序控制信号。

（2）运算器。运算器也称为算术逻辑单元（ArithmeticandLogicUnit，ALU），其主要功能是在控制器的控制下完成各种算术运算和逻辑运算。运算器的组成包含如下。

① 算术逻辑单元 ALU：数据的算术运算和逻辑运算；

② 累加寄存器 AC：通用寄存器，为 ALU 提供一个工作区，用在暂存数据；

③ 数据缓冲寄存器 DR：写内存时，暂存指令或数据；

④ 状态条件寄存器 PSW：存状态标志与控制标志（争议点：也有将其归为控制器的）。

（3）主存储器。主存储器也称为内存储器（通常简称为“内存”或“主存”）。存储现 场操作的信息与中间结果，包括机器指令和数据。

（4）辅助存储器。辅助存储器也称为外存储器，通常简称为外存或辅存。存储需要长期保存的各种信息。

（5）输入设备。输入设备的任务是把人们编好的程序和原始数据送到计算机中去，并 且将它们转换成计算机内部所能识别和接受的信息方式。按输入信息的形态可分为字符（包括汉字）输入、图形输入、图像输入及语音输入等。目前，常见的输入设备有键盘、 鼠标、扫描仪等。

（6）输出设备。输出设备的任务是将计算机的处理结果以人或其他设备所能接受的 形式送出计算机。目前，最常用的输出设备是打印机和显示器。有些设备既可以是输入 设备，同时也可以是输出设备，例如，辅助存储器、自动控制和检测系统中使用的数模转换装置等。

1.1.2 计算机系统结构的分类

计算机的发展经历了电子管和晶体管时代、集成电路时代（中小规模、大规模、超大规模、甚大规模、极大规模）。目前，世界最高水平的单片集成电路芯片上所容纳的元器 件数量已经达到 80 多亿个。

1. 存储程序的概念

“存储程序”的概念是冯•诺依曼等人于 1946 年 6 月首先提出来的，它可以简要地概括为以下几点：

（1）计算机（指硬件）应由运算器、存储器、控制器、输入设备和输出设备五大基本

部件组成。

（2）计算机内部采用二进制来表示指令和数据。

（3）将编好的程序和原始数据事先存入存储器中，然后再启动计算机工作。这就是存储程序的基本含义。冯•诺依曼对计算机世界的最大贡献在于“存储程序控制”概念的提出和实现。六十多年来，虽然计算机的发展速度惊人，但就其结构原理来说，目前绝大多数计算机仍建立在存储程序概念的基础上。通常把符合存储程序概念的计算机统称为冯•诺依曼型计算机。当然，现代计算机与早期计算机相比，在结构上还是有许多改进的。

随着计算机技术的不断发展，也暴露出了冯•诺依曼型计算机的主要弱点：存储器访问会成为瓶颈。目前，已出现了一些突破存储程序控制的计算机，统称为非冯•诺依曼型计算机，例如，数据驱动的数据流计算机、需求驱动的归约计算机和模式匹配驱动的智能计算机等。

2. Flynn 分类

1966 年，Michael．J．Flynn 提出根据指令流、数据流的多倍性特征对计算机系统进行分类（通常称为 Flynn 分类法），有关定义如下。

（1）指令流：指机器执行的指令序列；

（2）数据流：指由指令流调用的数据序列，包括输入数据和中间结果，但不包括输出数据。

Flynn 根据不同的指令流-数据流组织方式，把计算机系统分成以下四类。

（1）单指令流单数据流（Single Instruction stream and Single Data stream，SISD）：SISD 其实就是传统的顺序执行的单处理器计算机，其指令部件每次只对一条指令进行译码，并只对一个操作部件分配数据。

（2）单指令流多数据流（Single Instruction stream and Multiple Data stream，SIMD）：SIMD 以并行处理机（矩阵处理机）为代表，并行处理机包括多个重复的处理单元，由单一指令部件控制，按照同一指令流的要求为它们分配各自所需的不同数据。

（3）多指令流单数据流（Multiple Instruction stream and Single Data stream，MISD）：MISD 具有 n 个处理单元，按 n 条不同指令的要求对同一数据流及其中间结果进行不同的处理。一个处理单元的输出又作为另一个处理单元的输入。这类系统实际上很少见到。有文献把流水线看作多个指令部件，称流水线计算机是 MISD。

（4）多指令流多数据流（Multiple Instruction stream and Multiple Data stream，MIMD）：MIMD 是指能实现作业、任务、指令等各级全面并行的多机系统。如多核处理器、多处理

机属于 MIMD。

1.1.3 复杂指令集系统与精简指令集系统

在计算机系统结构发展的过程中，指令系统的优化设计有两个截然相反的方向，一个是增强指令的功能，设置一些功能复杂的指令，把一些原来由软件实现的、常用的功能改用硬件的指令系统来实现，这种计算机系统称为复杂指令系统计算机（Complex Instruction Set Computer，CISC）；另一个是尽量简化指令功能，只保留那些功能简单，能在一个节拍内执行完成指令，较复杂的功能用一段子程序来实现，这种计算机系统称为精简指令系统计算机（Reduced Instruction Set Computer，RISC）。

1. CISC 指令系统的特点

CISC 指令系统的主要特点如下：

（1）指令数量众多。指令系统拥有大量的指令，通常有 100～250 条。

（2）指令使用频率相差悬殊。最常使用的是一些比较简单的指令，仅占指令总数的 20%，但在程序中出现的频率却占 80%。而大部分复杂指令却很少使用。

（3）支持很多种寻址方式。支持的寻址方式通常为 5～20 种。

（4）变长的指令。指令长度不是固定的，变长的指令增加指令译码电路的复杂性。

（5）指令可以对主存单元中的数据直接进行处理。典型的 CISC 通常都有指令能够直接对主存单元中的数据进行处理，其执行速度较慢。

（6）以微程序控制为主。CISC 的指令系统很复杂，难以用硬布线逻辑（组合逻辑）电路实现控制器，通常采用微程序控制。

2. RISC 指令系统的特点

RISC 要求指令系统简化，操作在单周期内完成，指令格式力求一致，寻址方式尽可能减少，并提高编译的效率，最终达到加快机器处理速度的目的。RISC 指令系统的主要特点如下。

（1）指令数量少。优先选取使用频率最高的一些简单指令和一些常用指令，避免使用复杂指令。只提供了 LOAD（从存储器中读数）和 STORE（把数据写入存储器）两条指令对存储器操作，其余所有的操作都在 CPU 的寄存器之间进行。

（2）指令的寻址方式少。通常只支持寄存器寻址方式、立即数寻址方式和相对寻址方式。

（3）指令长度固定，指令格式种类少。因为 RISC 指令数量少、格式少、相对简单，其指令长度固定，指令之间各字段的划分比较一致，译码相对容易。

（4）以硬布线逻辑控制为主。为了提高操作的执行速度，通常采用硬布线逻辑（组合逻辑）来构建控制器。

（5）单周期指令执行，采用流水线技术。因为简化了指令系统，很容易利用流水线技术，使得大部分指令都能在一个机器周期内完成。少数指令可能会需要多周期，例如，LOAD/STORE 指令因为需要访问存储器，其执行时间就会长一些。

（6）优化的编译器：RISC 的精简指令集使编译工作简单化。因为指令长度固定、格式少、寻址方式少，编译时不必在具有相似功能的许多指令中进行选择，也不必为寻址方式的选择而费心，同时易于实现优化，从而可以生成高效率执行的机器代码。

（7）CPU 中的通用寄存器数量多，一般在 32 个以上，有的可达上千个。

大多数 RISC 采用了 Cache 方案，使用 Cache 来提高取指令的速度。而且，有的 RISC 使用两个独立的 Cache 来改善性能。一个称为指令 Cache，另一个称为数据 Cache。这样，取指令和取数据可以同时进行，互不干扰。

1.1.4 总线

总线是一组能为多个部件分时共享的公共信息传送线路。共享是指总线上可以挂接多个部件，各个部件之间相互交换的信息都可以通过这组公共线路传送；分时是指同一时刻只允许有一个部件向总线发送信息，如果出现两个或两个以上部件同时向总线发送信息，势必导致信号冲突。当然，在同一时刻，允许多个部件同时从总线上接收相同的信息。

按总线相对于 CPU 或其他芯片的位置可分为内部总线和外部总线两种。在 CPU 内部，寄存器之间和算术逻辑部件 ALU 与控制部件之间传输数据所用的总线称为内部总线；外部总线是指 CPU 与内存 RAM、ROM 和输入/输出设备接口之间进行通信的通路。由于 CPU 通过总线实现程序取指令、内存/外设的数据交换，在 CPU 与外设一定的情况下，总线速度是制约计算机整体性能的最大因素。

按总线功能来划分，又可分为地址总线、数据总线、控制总线三类，人们通常所说的总线都包括这三个组成部分，地址总线用来传送地址信息，数据总线用来传送数据信息，控制总线用来传送各种控制信号。

1.2 存储器系统

存储器是用来存放程序和数据的部件，它是一个记忆装置，也是计算机能够实现“存储程序控制”的基础。在计算机系统中，规模较大的存储器往往分成若干级，称为存储器系统。

传统的存储器系统一般分为高速缓冲存储器（Cache）、主存、辅存三级。主存可由 CPU 直接访问，存取速度快，但容量较小，一般用来存放当前正在执行的程序和数据。辅存设置在主机外部，它的存储容量大，价格较低，但存取速度较慢，一般用来存放暂时不参与运行的程序和数据，CPU 不可以直接访问辅存，辅存中的程序和数据在需要时才传送到主存，因此它是主存的补充和后援。当 CPU 速度很高时，为了使访问存储器的速度能与 CPU 的速度相匹配，又在主存和 CPU 间增设了一级 Cache。Cache 的存取速度比主存更快，但容量更小，用来存放当前最急需处理的程序和数据，以便快速地向 CPU 提供指令和数据。因此，计算机采用多级存储器体系，确保能够获得尽可能高的存取速率，同时保持较低的成本。

多层级的存储体系之所以能用低投入换来较高的存取速率，得益于局部性原理。局部性原理是指程序在执行时呈现出局部性规律，即在一较短的时间内，程序的执行仅局限于某个部分。相应地，它所访问的存储空间也仅局限于某个区域。程序局部性包括时间局部性和空间局部性，时间局部性是指程序中的某条指令一旦执行，不久以后该指令可能再次执行。产生时间局部性的典型原因是由于程序中存在着大量的循环操作；空间局部性是指一旦程序访问了某个存储单元，不久以后，其附近的存储单元也将被访问，即程序在一段时间内所访问的地址可能集中在一定的范围内，其典型情况是程序顺序执行。

存储器中数据常用的存取方式有顺序存取、直接存取、随机存取和相联存取四种。

（1）顺序存取：存储器的数据以记录的形式进行组织。对数据的访问必须按特定的线性顺序进行。磁带存储器采用顺序存取的方式。

（2）直接存取：与顺序存取相似，直接存取也使用一个共享的读写装置对所有的数据进行访问。但是，每个数据块都拥有唯一的地址标识，读写装置可以直接移动到目的数据块所在位置进行访问。存取时间也是可变的。磁盘存储器采用直接存取的方式。

（3）随机存取：存储器的每一个可寻址单元都具有自己唯一的地址和读写装置，系统可以在相同的时间内对任意一个存储单元的数据进行访问，而与先前的访问序列无关。主存储器采用随机存取的方式。

（4）相联存取：相联存取也是一种随机存取的形式，但是选择某一单元进行读写是取决于其内容而不是其地址。与普通的随机存取方式一样，每个单元都有自己的读写装置，读

写时间也是一个常数。使用相联存取方式，可以对所有的存储单元的特定位进行比较，选择符合条件的单元进行访问。为了提高地址映射的速度，Cache 采取相联存取的方式。

1.2.1 主存储器

主存用来存放计算机运行期间所需要的程序和数据，CPU 可直接随机地进行读/写。主存具有一定容量，存取速度较高。由于 CPU 要频繁地访问主存，所以主存的性能在很大程度上影响了整个计算机系统的性能。根据工艺和技术不同，主存可分为随机存取存储器和只读存储器。

1.随机存取存储器

随机存取存储器（Random Access Memory，RAM）既可以写入也可以读出，但断电后信息无法保存，因此只能用于暂存数据。RAM 又可分为 DRAM（Dynamic RAM，动态 RAM）和 SRAM（Static RAM，静态 RAM）两种，DRAM 的信息会随时间逐渐消失，因此需要定时对其进行刷新维持信息不丢失；SRAM 在不断电的情况下信息能够一直保持而不会丢失。DRAM 的密度大于 SRAM 且更加便宜，但 SRAM 速度快，电路简单（不需要刷新电路），然而容量小，价格高。

2.只读存储器

只读存储器（Read Only Memory，ROM）可以看作 RAM 的一种特殊形式，其特点是：存储器的内容只能随机读出而不能写入。这类存储器常用来存放那些不需要改变的信息。由于信息一旦写入存储器就固定不变了，即使断电，写入的内容也不会丢失，所以又称为固定存储器。ROM 一般用于存放系统程序 BIOS（Basic Input Output System，基本输入输出系统）。

3.内存编址方法在计算机系统中，存储器中每个单元的位数是相同且固定的，称为存储器编址单位。

不同的计算机，存储器编址的方式不同，主要有字编址和字节编址。

内存一般以字节（8 位）为单位，或者以字为单位（字的长度可大可小，例如 16 位或者 32 位等，在这类试题中，一般会给出字的大小）。

例如，内存地址从 AC000H 到 C7FFFH，则共有 C7FFFFH-AC000H=1BFFFH 个地址单元（转换为十进制后，为 112KB）。如果该内存地址按字（16bit）编址，则共有 112KB*16 位。假设该内存由 28 片存储器芯片构成，已知构成此内存的芯片每片有 16KB 个存储单元，则该芯片每个存储单元存储（112KB*16）/（28*16KB）=4 位。

1.2.2 辅助存储器

1．磁带存储器磁带存储器是一种顺序存取的设备，其特点包括：存取时间较长，但存储容量大，便于携带，价格便宜。磁带应用的场景越来越少，目前主要用于资料的归档保存。

2．硬盘存储器在硬盘中，信息分布呈以下层次：记录面、圆柱面、磁道和扇区，如图 1-2 所示。

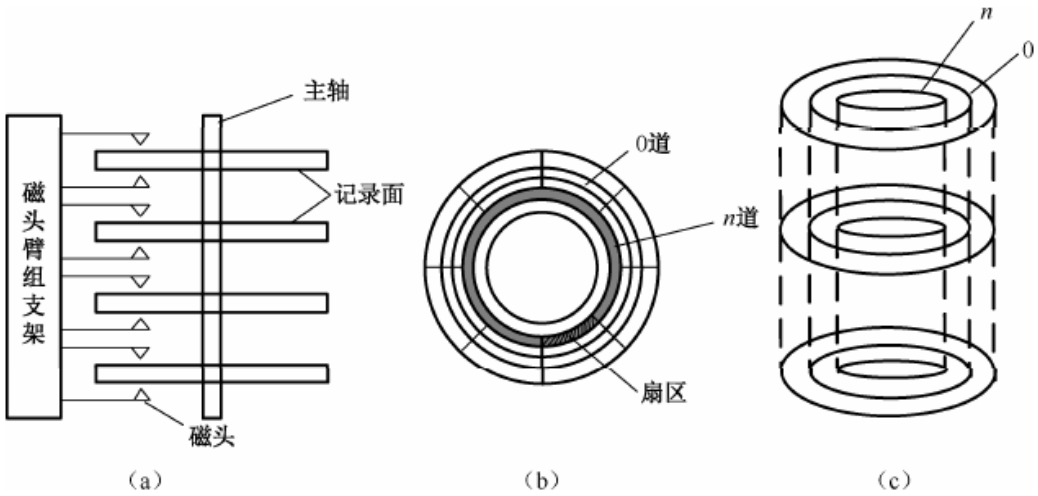


图 1-2　硬盘信息分布示意图

一台硬盘驱动器中有多个磁盘片，每个盘片有两个记录面，每个记录面对应一个磁头，所以记录面号就是磁头号，如图 1-2（a）所示。所有的磁头安装在一个公用的传动设备或支架上，磁头一致地沿盘面径向移动，单个磁头不能单独地移动。在记录面上，一条条磁道形成一组同心圆，最外圈的磁道为 0 号，往内则磁道号逐步增加，如图 1-2（b）所示。在一个盘组中，各记录面上相同编号（位置）的各磁道构成一个柱面，如图 1-2（c）所示。

若每个磁盘片有 m 个磁道，则该硬盘共有 m 个柱面。引入柱面的概念是为了提高硬盘的存储速度。当主机要存入一个较大的文件时，若一条磁道存不完，就需要存放在几条磁道上。这时，应首先将一个文件尽可能地存放在同一柱面中。如果仍存放不完，再存入相邻的柱面内。

通常将一条磁道划分为若干个段，每个段称为一个扇区或扇段，每个扇区存放一个定长信息块（例如，512 个字节），如图 1-2（b）所示。一条磁道划分多少扇区，每个扇区可存放多少字节，一般由操作系统决定。磁道上的扇区编号从 1 开始，不像磁头或柱面编号从 0 开始。

在磁盘上进行信息的读写时，首先需要定位到目标磁道，这个过程称之为寻道，寻道所消耗的时间称为寻道时间，定位到目标磁道后，需要定位到目标扇区，此过程通过旋转盘片完成，平均旋转半圈可到目标位置。故磁盘访问时间为：

磁盘访问时间（存取时间） = 寻道时间+旋转延迟时间

1.2.3 Cache 存储器

Cache 的功能是提高 CPU 数据输入输出的速率，突破所谓的“冯•诺依曼瓶颈”，即 CPU 与存储系统间数据传送带宽限制。高速存储器能以极高的速率进行数据访问，但因其价格高昂，如果计算机的内存完全由这种高速存储器组成，则会大大增加计算机的成本。通常在 CPU 和内存之间设置小容量的 Cache。Cache 容量小但速度快，内存速度较低但容量大，通过优化调度算法，系统的性能会大大改善，仿佛其存储系统容量与内存相当而访问速度近似 Cache。

Cache 通常采用相联存储器（ContentAddressable Memory，CAM）。CAM 是一种基于数据内容进行访问的存储设备。当对其写入数据时，CAM 能够自动选择一个未用的空单元进行存储；当要读出数据时，不是给出其存储单元的地址，而是直接给出该数据或者该数据的一部分内容，CAM 对所有存储单元中的数据同时进行比较，并标记符合条件的所有数据以供读取。由于比较是同时、并行进行的，所以，这种基于数据内容进行读写的机制，其速度比基于地址进行读写的方式要快很多。

1．Cache 基本原理

使用 Cache 改善系统性能的依据是程序的局部性原理。根据程序的局部性原理，最近的、未来要用的指令和数据大多局限于正在用的指令和数据，或是存放在与这些指令和数据位置上邻近的单元中。这样，就可以把目前常用或将要用到的信息预先放在 Cache 中。当 CPU 需要读取数据时，首先在 Cache 中查找是否有所需内容，如果有，则直接从 Cache 中读取；若没有，再从内存中读取该数据，然后同时送往 CPU 和 Cache。如果 CPU 需要访问的内容大多都能在 Cache 中找到（称为访问命中），则可以大大提高系统性能。

如果以 h 代表对 Cache 的访问命中率（“1-h”称为失效率，或者称为未命中率），t1 表示 cache 的周期时间，t2 表示内存的周期时间，以读操作为例，使用“Cache+主存储器”的系统的平均周期为 t3。则：

$$t3 = t1' h + t2' (1-h)$$

系统的平均存储周期与命中率有很密切的关系，命中率的提高即使很小也能导致性能上的较大改善。

例如，设某计算机主存的读/写时间为 l00ns，有一个指令和数据合一的 Cache，已知该 Cache 的读/写时间为 10ns，取指令的命中率为 98%，取数的命中率为 95%。在执行某类程序时，约有 1/5 指令需要存/取一个操作数。假设指令流水线在任何时候都不阻塞，则设置 Cache 后，每条指令的平均访存时间约为：

(2%′ 100ns+98%′ 10ns)+1/5′ (5%′ 100ns+95%′ 10ns)=14.7ns

2．映射机制

当 CPU 发出访存请求后，存储器地址先被送到 Cache 控制器以确定所需数据是否已在 Cache 中，若命中则直接对 Cache 进行访问。这个过程称为 Cache 的地址映射（映像）。在 Cache 的地址映射中，主存和 Cache 将均分成容量相同的块（页）。常见的映射方法有直接映射、全相联映射和组相联映射。

（1）直接映像

直接映像方式以随机存取存储器作为 Cache 存储器，硬件电路较简单。在进行映像时，主存地址被分成三个部分，从高到低依次为：区号、页号以及页内地址，如图 1-3 所示。

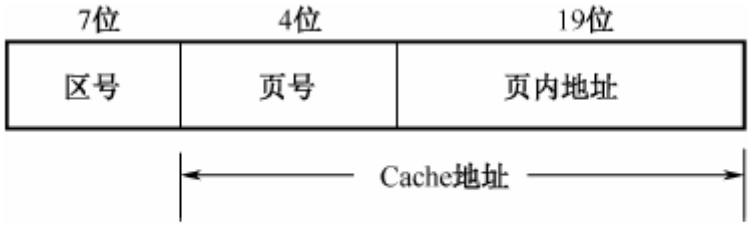


图 1-3　直接映像方式的主存地址

在本例中，内存容量为 1GB，Cache 容量为 8MB，页面的大小为 512KB。直接映像中，先分区，再分页。一个区的大小就是 Cache 容量的大小，所以一共分：1GB/8MB=128 个区，区号 7 位。每个区分：8MB/512KB=16 个页，所以页号为 4 位。

在直接映像方式中，每个主存页只能复制到某一固定的 Cache 页中，如图 1-4 所示。直接映像方式的映像规律是：主存中每个区的第 0 页，只能进入到 Cache 的第 0 页。即：若当前时刻 Cache 中 0 号页已被占据，而 1-15 号页空闲，现在要将 1 区第 0 页（即内存的 16 页）调入 Cache 是会发生冲突的。所以直接映像的块冲突率非常高。

在 Cache 中，为每一个页设立一个 Cache 标记，该标记用于识别当前的 Cache 块来自于哪个内存页。直接映像中，由于每个区的 N 号页，都必须进入到 Cache 的 N 号页，

所以只需要记录区号即可。所以此时标记位的长度是 7 位。

直接映像方式的优点是比较容易实现，缺点是不够灵活，有可能使 Cache 的存储空间得不到充分利用。

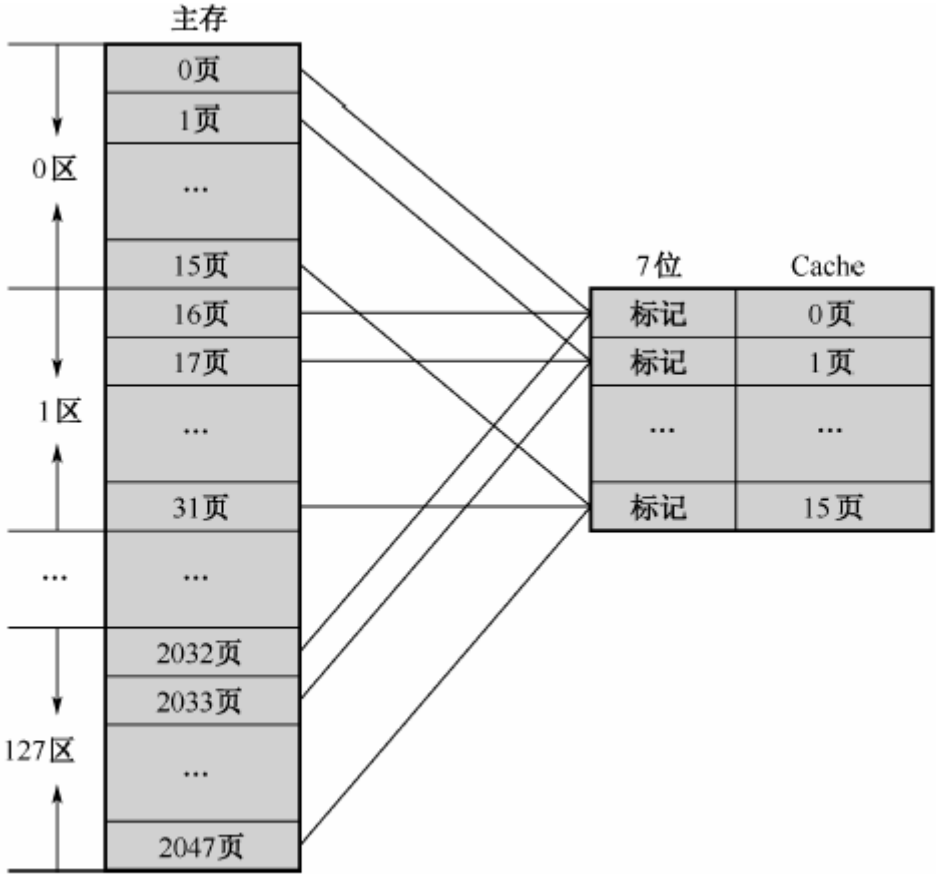


图 1-4　直接映像方式

（2）全相联映像

全相联映像使用相联存储器组成的 Cache 存储器。在全相联映像方式中，主存的每一页可以映像到 Cache 的任一页。如果淘汰 Cache 中某一页的内容，则可调入任一主存页的内容，因而较直接映像方式灵活。

在全相联映像方式中，主存地址分为两个部分，分别为地址部分（主存页标记）和数据部分（页内地址）。数据部分用于存放数据，而地址部分则存放该数据的存储器地址。如图 1-5 所示。

图 1-5　全相联映像方式的主存地址

全相联映像方式的 Cache 组织如图 1-6 所示。

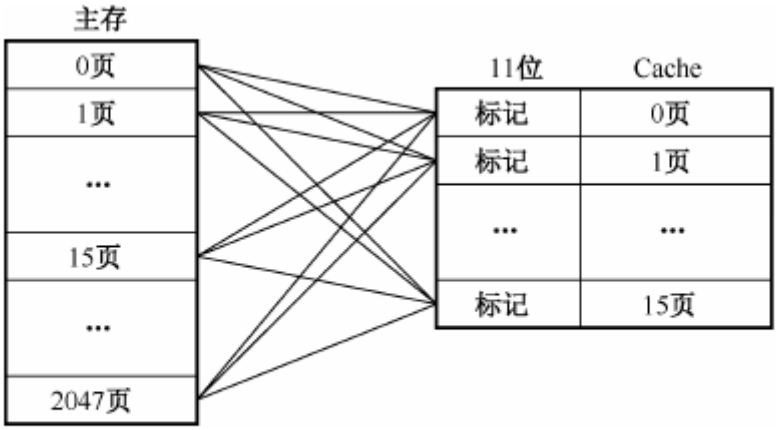


图 1-6　全相联映像方式

当进行映像时，在我们给定的例子中，当程序访存时，则高 11 位给出主存页号，低 19 位给出页内地址。因为每个 Cache 页可映像到 2048 个主存页中的任一页，所以每页的 Cache 标记也需要 11 位，以表明它现在所映像的主存页号。因此，Cache 标记信息位数增加，比较逻辑成本随之增加。

在全相联映像方式中，主存地址不能直接提取 Cache 页号，而是需要将主存页标记与 Cache 各页的标记逐个比较，直到找到标记符合的页（访问 Cache 命中），或者全部比较完后仍无符合的标记（访问 Cache 失败）。因此这种映像方式速度很慢，失掉了高速缓存的作用，这是全相联映像方式的最大缺点。如果让主存页标记与各 Cache 标记同时比较，则成本又太高。全相联映像方式因比较器电路难于设计和实现，只适用于小容量 Cache。

（3）组相联映像

组相联映像（页组映像）介于直接映像和全相联映像之间，是这两种映像的一种折衷方案。全相联映像方式以页为单位，可自由映像，没有固定的对应关系。直接映像方式中，主存分组，主存组内的各页与 Cache 的页之间采取的是固定的映像关系，但各组均可映像到 Cache 中。在组相联映像方式中，主存与 Cache 都分组，主存中一个组内的页数与 Cache 的

分组数相同，如图 1-7 所示。

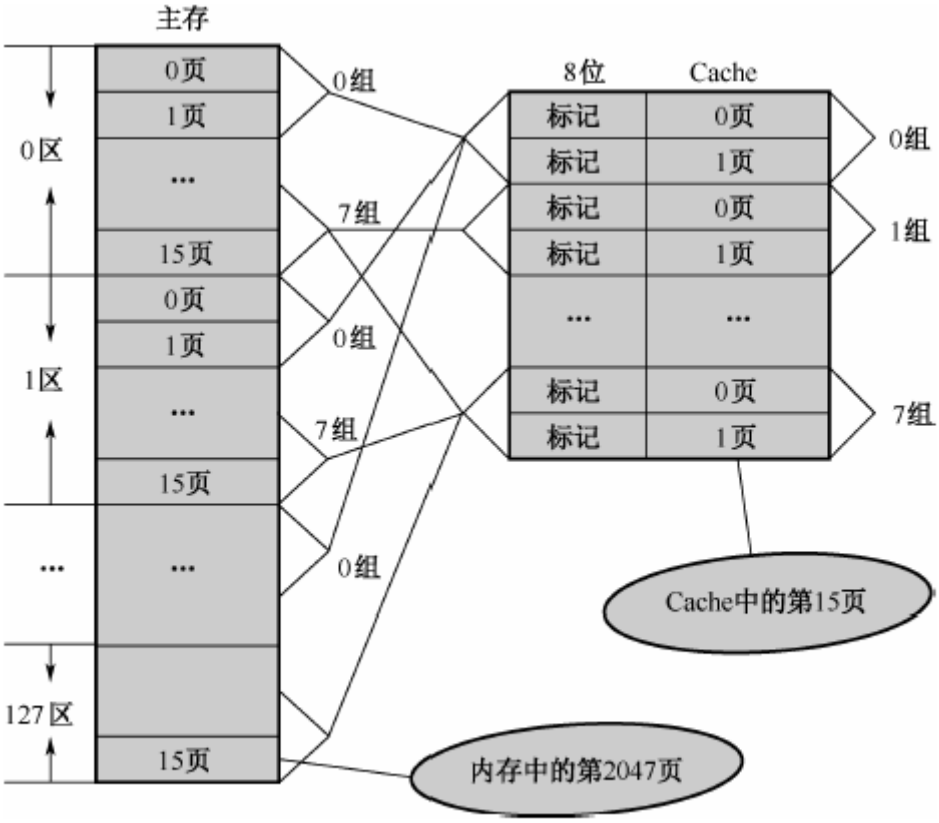


图 1-7　组相联映像方式

在图 1-7 给出的例子中，主存分 128 个区，每个区 8 个组，每个组 2 个页。组相联映像方式的主存地址组织如图 1-8 所示。

图 1-8　组相联映像方式的主存地址

组相联映像的规则是：主存中的组与 Cache 的组形成直接映像关系，而每个组内的页是全相联映像关系。如主存 1 区 0 页，他在 0 组中，所以只能进入 Cache 的 0 组中，至于进入到 Cache 的 0 组 0 页，还是 0 组 1 页，并无强制要求，可任意放置。

在组相联映像中，Cache 中每一页的标记位长度为 8 位，因为此时除了要记录区号，还得记录组号，即区号 7 位加组号 1 位等于 8 位。

容易看出，如果 Cache 中每组只有一页，则组相联映像方式就变成了直接映像方式。如

果 Cache 中每组页数为 16 页（即 Cache 只分一组），则就是全相联映像。因此，在具体设计组相联映像时，可以根据设计目标选取某一折衷值。

在组相联映像中，由于 Cache 中每组有若干可供选择的页，因而它在映像定位方面较直接映像方式灵活；每组页数有限，因此付出的代价不是很大，可以根据设计目标选择组内页数。

希赛教育专家提示：为保障性能，内存与 Cache 之间的映射往往采用硬件完成，所以 Cache 对于程序员而言是透明的，程序员编程时，完全不用考虑 Cache。

3．替换算法

当 Cache 产生了一次访问未命中之后，相应的数据应同时读入 CPU 和 Cache。但是当 Cache 已存满数据后，新数据必须替换（淘汰）Cache 中的某些旧数据。最常用的替换算法有以下三种：

（1）随机算法。这是最简单的替换算法。随机法完全不管 Cache 块过去、现在及将来的使用情况，简单地根据一个随机数，选择一块替换掉。

（2）先进先出（First In and First Out，FIFO）算法。按调入 Cache 的先后决定淘汰的顺序，即在需要更新时，将最先进入 Cache 的块作为被替换的块。这种方法要求为每块做一记录，记下它们进入 Cache 的先后次序。这种方法容易实现，而且系统开销小。其缺点是可能会把一些需要经常使用的程序块（如循环程序）替换掉。

（3）近期最少使用（Least Recently Used，LRU）算法。LRU 算法是把 CPU 近期最少使用的块作为被替换的块。这种替换方法需要随时记录 Cache 中各块的使用情况，以便确定哪个块是近期最少使用的块。LRU 算法相对合理，但实现起来比较复杂，系统开销较大。通常需要对每一块设置一个称为“年龄计数器”的硬件或软件计数器，用以记录其被使用的情况。

4．写操作

因为需要保证缓存在 Cache 中的数据与内存中的内容一致，相对读操作而言，Cache 的写操作比较复杂，常用的有以下几种方法。

（1）写直达（write through）。当要写 Cache 时，数据同时写回内存，有时也称为写通。当某一块需要替换时，也不必把这一块写回到主存中去，新调入的块可以立即把这一块覆盖掉。这种方法实现简单，而且能随时保持主存数据的正确性，但可能增加多次不必要的主存写入，会降低存取速度。

（2）写回（write back）。CPU 修改 Cache 的某一块后，相应的数据并不立即写入内存

单元，而是当该块从 cache 中被淘汰时，才把数据写回到内存中。在采用这种更新策略的 cache 块表中，一般有一个标志位，当一块中的任何一个单元被修改时，标志位被置“1”。在需要替换掉这一块时，如果标志位为“1”，则必须先把这一块写回到主存中去之后，才能再调入新的块；如果标志位为“0”，则这一块不必写回主存，只要用新调入的块覆盖掉这一块即可。这种方法的优点是操作速度快，缺点是因主存中的字块未随时修改而有可能出错。

（3）标记法。对 Cache 中的每一个数据设置一个有效位。当数据进入 Cache 后，有效位置“1”；而当 CPU 要对该数据进行修改时，数据只需写入内存并同时将该有效位置“0”。当要从 Cache 中读取数据时需要测试其有效位，若为“I”则直接从 Cache 中取数，否则，从内存中取数。

1.3 流水线

流水线技术把一个任务分解为若干顺序执行的子任务，不同的子任务由不同的执行机构负责执行，而这些机构可以同时并行工作。在任一时刻，任一任务只占用其中一个执行机构，这样就可以实现多个任务的重叠执行，以提高工作效率。

1.3.1 流水线周期

流水线应用过程中，会将需要处理的工作分为 N 个阶段，最耗时的那一段所消耗的时间为流水线周期。如：使用流水线技术执行 100 条指令，每条指令取指 2ms，分析 4ms，执行 1ms，则流水线周期为 4ms。

1.3.2 计算流水线执行时间

延续上面的场景，将 1 个任务的执行过程可分成 N 个阶段，假设每个阶段完成时间为 t，则完成该任务所需的时间即为 Nt。若以传统的方式，则完成 k 个任务所需的时间是 kNt；而使用流水线技术执行，且花费的时间是 Nt+(k-1)t。也就是说，除了第 1 个任务需要完整的时间外，其他都通过并行，节省下了大量的时间。所以流水线的执行时间可通俗的表达为：

流水线执行时间=第 1 条指令的执行时间+（n-1）*流水线周期

注：n 代表需要处理的任务数量。

在考试时，又需要特别注意一个细节问题，流水线的执行时间计算，其实进一步可以分

理论情况与实践情况两种不同的处理方式。下面以实例进行说明。

例：某计算机系统，一条指令的执行需要经历取指（2ms）、分析（4ms）、执行（1ms）三个阶段，现要执行 100 条指令，利用流水线技术需要多长时间?

理论上来说，1 条指令的执行时间为：2ms+4ms+1ms=7ms。

所以：理论流水线执行时间=2ms+4ms+1ms+(100-1)*4=403ms。

而实际上，真正做流水线处理时，考虑到处理的复杂性，会将指令的每个执行阶段的时间都统一为流水线周期，即 1 条指令的执行时间为：4ms+4ms+4ms=12ms。 所以：实际流水线执行时间=4ms+4ms+4ms+(100-1)*4=408ms。

希赛教育专家提示：考试时 80%以上的概率采用理论公式计算，所以考试时需要以理论公式计算，若计算的结果无正确选项才考虑采用实际公式计算。

1.3.3 流水线的吞吐率

流水线的吞吐率（Though Put rate，TP）是指在单位时间内流水线所完成的任务数量或输出的结果数量。有些文献也称为平均吞吐率、实际吞吐率。计算流水线吞吐率的最基本的公式如下：

$$TP = \frac{n}{T_k}$$

其中 n 为任务数，T_k 是处理完成 n 个任务所用的时间。

流水线的最大吞吐率为：

$$TP_{\max} = \lim_{n\to\infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t}$$

1.3.4 流水线的加速比

在流水线中，因为在同一时刻，有多个任务在重叠地执行，虽然完成一个任务的时间与单独执行该任务相近（甚至由于分段的缘故，可能更多一些），但是从整体上看完成多个任务所需的时间则大大减少。

完成同样一批任务，不使用流水线所用的时间与使用流水线所用的时间之比称为流水线的加速比（speedup ratio）。如果不使用流水线，即顺序执行所用的时间为 T0 ，使用流水线的执行时间为 Tk ，则计算流水线加速比的基本公式如下：

$$S = \frac{T_0}{T_k}$$

如果流水线各个流水段的执行时间都相等（设为 Dt），则一条 k 段流水线完成 n 个连 续任务所需要的时间为(k+n-1)Dt。如果不使用流水线，即顺序执行这 n 个任务，则所需要的时间为 nkDt。因此，各个流水段执行时间均相等的一条 k 段流水线完成 n 个连续任务 时的实际加速比为：

$$S = \frac{nk\Delta t}{(k+n-1)\Delta t} = \frac{nk}{k+n-1}$$

这种情况下的最大加速比为：

$$S_{\max} = \lim_{n\to\infty} \frac{nk}{k+n-1} = k$$

第 2 章 操作系统

本章主要介绍操作系统的基本概念及其形成、发展历史和主要类型，并指出操作系统的 5 大管理功能。掌握操作系统原理的关键在于深入理解“一个观点、两条线索”。一个观点是以资源管理的观点来定义操作系统；两条线索是指操作系统如何管理计算机各类资源和控制程序的执行。操作系统如何实现对这些资源的管理，其内涵、设计和实现是本章的主要内容。

2.1 操作系统的类型与结构

计算机系统由硬件和软件两部分组成。操作系统是计算机系统中最基本的系统软件，它既管理计算机系统的软、硬件资源，又控制程序的执行。操作系统随着计算机研究和应用的发展逐步形成并日趋成熟，它为用户使用计算机提供了一个良好的环境，从而使用户能充分利用计算机资源，提高系统的效率。操作系统的基本类型有：批处理操作系统、分时操作系统和实时操作系统。从资源管理的角度看，操作系统主要是对处理器、存储器、文件、设备和作业进行管理。

2.1.1 操作系统的定义

操作系统（Operating System，OS）是计算机系统中的核心系统软件，负责管理和控制计算机系统中的硬件和软件资源，合理地组织计算机工作流程和有效地利用资源，在计算机与用户之间起接口的作用。操作系统为用户提供的接口表现形式一般为：命令、菜单、窗口之类的，而操作系统为应用程序提供的接口为 API。操作系统与硬件/软件的关系如图 2-1 所示。

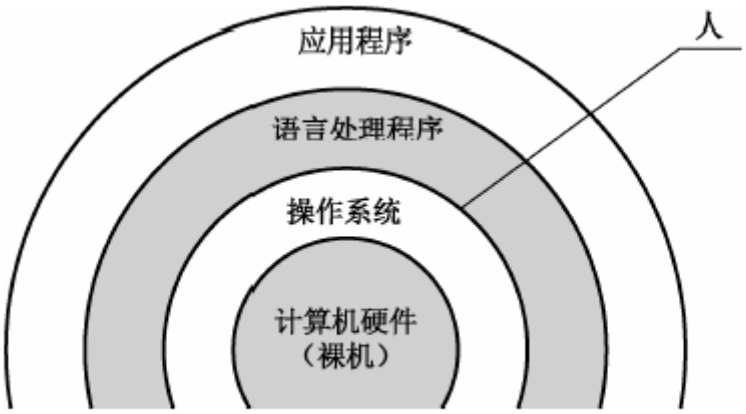


图 2-1　操作系统与硬件/软件的关系

2.1.2 操作系统分类

按照操作系统的功能划分，操作系统的基本类型有批处理操作系统、分时操作系统、实时操作系统、网络操作系统、分布式操作系统、嵌入式操作系统、微内核操作系统等。

2.2 操作系统基本原理

操作系统的主要功能是进行处理机与进程管理、存储管理、设备管理、文件管理和作业管理的工作，本节讨论操作系统是如何完成这些功能的。

2.2.1 进程管理

处理机是计算机系统的核心资源。操作系统的功能之一就是处理机管理。随着计算机的迅速发展，处理机管理显得更为重要，这主要由于计算机的速度越来越快，处理机的充分利用有利于系统效率的大大提高；处理机管理是整个操作系统的重心所在，其管理的好坏直接

影响到整个系统的运行效率；而且操作系统中并发活动的管理和控制是在处理机管理下实现的，处理机管理集中了操作系统中最复杂的部分，它设计的好坏关系到整个系统的成败。

进程是处理机管理中最基本的、最重要的概念。进程是系统并发执行的体现。由于在多道程序系统中，众多的计算机用户都以各种各样的任务，随时随地争夺使用处理机。为了动态地看待操作系统，则以进程作为独立运行的基本单位，以进程作为分配资源的基本单位，从进程的角度来研究操作系统。因此，处理机管理也被称为进程管理。处理机管理的功能就是组织和协调用户对处理机的争夺使用，把处理机分配给进程，对进程进行管理和控制，最大限度也发挥处理机的作用。

1. 进程的概念用静态的观点看，操作系统是一组程序和表格的集合。用动态的观点看，操作系统是进程的动态和并发执行的。而进程的概念实际上是程序这一概念发展的产物。因此，可以从分析程序的基本特征入手，引出“进程”的概念。

顺序程序是指程序中若干操作必须按照某种先后次序来执行，并且每次操作前和操作后的数据、状态之间都有一定的关系。在早期的程序设计中，程序一般都是按顺序执行的。

在多道程序系统中，程序的运行环境发生了很大的变化。主要体现在：

（1）资源共享。为了提高资源的利用率，计算机系统中的资源不再由一道程序专用，而是由多道程序共同使用。

（2）程序的并发执行或并行执行。逻辑上讲允许多道不同用户的程序并行运行；允许一个用户程序内部完成不同操作的程序段之间并行运行；允许操作系统内部不同的程序之间并行运行。物理上讲：内存储器中保存多个程序，I/O 设备被多个程序交替地共享使用；在多处理机系统的情形下，表现为多个程序在各自的处理机上运行，执行时间是重叠的。单处理机系统时，程序的执行表现为多道程序交替地在处理机上相互空插运行。

实际上，在多道程序系统中，程序的并行执行和资源共享之间是相辅相成的。一方面，只有允许程序并行执行，才可能存在资源共享的问题；另一方面，只有有效地实现资源共享，才可能使得程序并行执行。

这样，可增强计算机系统的处理能力和提高机器的利用率。并发操作实际上是这样的：大多数程序段只要求操作在时间上是有序的，也就是有些操作必须在其他操作之前，这是有序的，但其中有些操作却可以同时进行。

2．进程的状态转换

由进程运行的间断性，决定了进程至少具有以下三种状态：

（1）就绪状态。当进程已分配了除 CPU 以外的所有必要的资源后，只要能再获得处

理机，便能立即执行，把这时的进程状态称为就绪状态。在一个系统中，可以有多个进程同时处于就绪状态，通常把它们排成一个队列，称为就绪队列。

（2）执行状态指进程已获得处理机，其程序正在执行。在单处理机系统中，只能有一个进程处于执行状态。

（3）阻塞状态指进程因发生某事件（如请求 I/O、申请缓冲空间等）而暂停执行时的状态，亦即进程的执行受到阻塞，故称这种暂停状态为阻塞状态，有时也称为“等待”状态，或“睡眠”状态。通常将处于阻塞状态的进程排成一个队列，称为阻塞队列。

进程的状态随着自身的推进和外界的变化而变化。例如，就绪状态的进程被进程调度程序选中进入执行状态；执行状态的进程因等待某一事件的发生转入等待状态；等待状态的进程所等待事件来到便进入就绪状态。进程的状态可以动态地相互转换，但阻塞状态的进程不能直接进入执行状态，就绪状态的进程不能直接进入阻塞状态。在任何时刻，任何进程都处于且只能处于这其中一种状态。进程状态的变化情况如下：

（1）运行态→等待态：一个进程运行中启动了外围设备，它就变成等待外围设备传输信息的状态；进程在运行中申请资源（主存储空间及外围设备因得不到满足）时，变成等待资源状态，进程在运行中出现了故障（程序出错或主存储器读写错误等），变成等待干预状态。

（2）等待态→就绪态：外围设备工作结束后等待外围设备传输信息的进程结束等待；等待的资源能得到满足时（另一个进程归还了资源），则等待资源者就结束等待；故障排队后让等待干预的进程结束等待，任何一个结束等待的进程必须先变成就绪状态，待分配到处理器后才能运行。

（3）运行态→就绪态：进程用完了一个使用处理器的时间后强迫该进程暂时让出处理器，当有更优先权的进程要运行时也迫使正在运行的进程让出处理器。由于自身或外界原因成为等待状态的进程让出处理器时，它的状态就变成就绪状态。

（4）就绪态→运行态：等待分配处理器的进程，系统按一种选定的策略从处于就绪状态的进程中选择一个进程，让它占用处理器，那个被选中的进程就变成了运行态。

图 2-2 所示为进程的三种基本状态及各状态之间的转换。

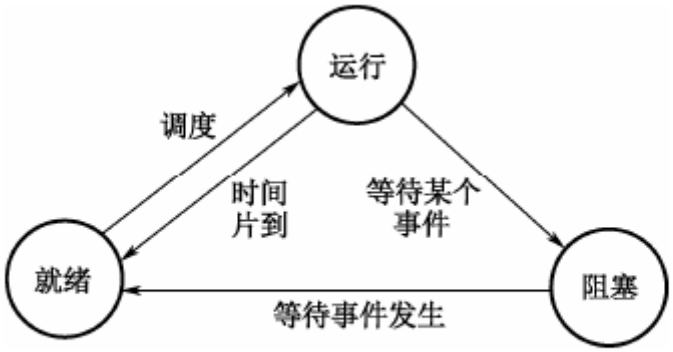


图 2-2　进程三态模型及其状态转换

3．关于挂起状态

在不少系统中，进程只有如图 2-2 所示的三种状态。但在另一些系统中，又增加了一些新状态，其中最重要的是挂起状态。引入挂起状态的原因有：

（1）对换的需要。为了缓和内存紧张的情况，而将内存中处于阻塞状态的进程换至外存上，使进程又处于一种有别于阻塞状态的新状态。因为即使该进程所期待的事件发生，该进程仍不具备执行条件而不能进入就绪队列，称这种状态为挂起状态。

（2）终端用户的请求。当终端用户在自己的程序运行期间，发现有可疑问题时，往往希望使自己的进程暂停下来。也就是说，使正在执行的进程暂停执行，若是就绪进程，则不接受调度以便研究其执行情况或对程序进行修改。把这种静止状态也称为挂起状态。

（3）父进程请求。父进程常希望挂起自己的子进程，以便考查和修改子进程，或者协调各子进程间的活动。

（4）负荷调节的需要。当实时系统中的工作负荷较重，有可能影响到对实时任务的控制时，可由系统把一些不重要的进程挂起，以保证系统正常运行。

（5）操作系统的需要。操作系统希望挂起某些进程，以便检查运行中资源的使用情况及进行记账。

综上所述，不难了解挂起状态具有以下三个属性。

（1）被挂起的进程，原来可能处于就绪状态，此时进程（被挂起）的状态称为挂起就绪；若被挂起的进程原来处于阻塞状态，此时的状态称为挂起阻塞。不论哪种状态，该进程都是不可能被调度而执行的。

（2）处于挂起阻塞状态的进程，其阻塞条件与挂起条件无关；当进程所期待的事件出现后，进程虽不再被阻塞，但仍不能运行，这时，应将该进程从静止阻塞状态转换为挂起就绪状态。

（3）进程可以由其自身挂起，也可由用户或操作系统等将之挂起。其目的都在于阻止进程继续运行，被挂起的进程只能用显式方式来激活，以便从挂起状态中解脱出来。

如图 2-3 所示为具有挂起操作的进程状态的演变情况。

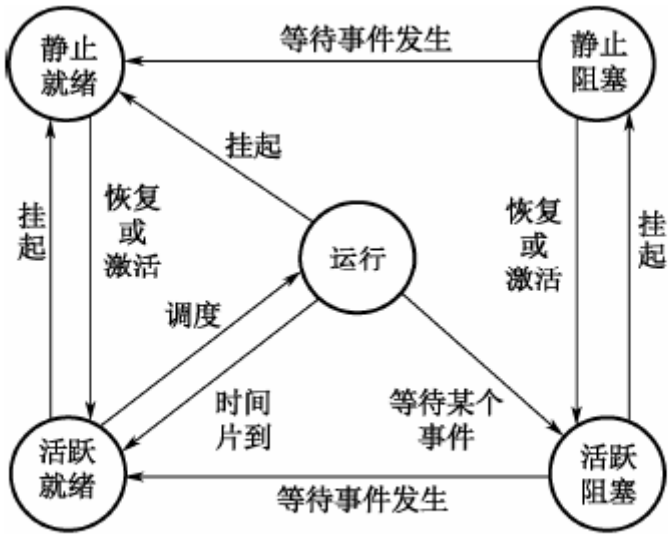


图 2-3　进程状态转换

4．进程互斥与同步进程互斥定义为：一组并发进程中一个或多个程序段，因共享某一共有资源而导致必须以一个不允许交叉执行的单位执行。也就是说互斥是要保证临界资源在某一时刻只被一个进程访问。

进程同步定义为：把异步环境下的一组并发进程因直接制约而互相发送消息而进行互相合作、互相等待，使得各进程按一定的速度执行的过程称为进程同步。也就是说进程之间是异步执行的，同步即是使各进程按一定的制约顺序和速度执行。

希赛教育专家提示：简单一点来说，互斥是资源的竞争关系，而同步是进程间的协作关系。

系统中有些资源可以供多个进程同时使用，有些资源则一次仅允许一个进程使用，将一次仅允许一个进程使用的资源称为临界资源，很多物理设备如打印机、磁带机等都属于临界资源，某些软件的变量、数据、表格也不允许两个进程同时使用，所以也是临界资源。

进程在并发执行中可以共享系统中的资源。但是临界资源的访问则必须互斥进行，即各进程对临界资源进行操作的那段程序的执行也必须是互斥的，只有这样才能保证对临界资源的互斥访问。把一个进程访问临界资源的那段程序代码称为临界区，有了临界区的概念，进程间的互斥就可以描述为：禁止两个或两个以上的进程同时进入访问同一临界资源的临界区。为此，必须有专门的同步机构来协调它们，协调准则如下：

（1）空闲让进。无进程处于临界区时，若有进程要求进入临界区则立即允许其进入；

（2）忙则等待。当已有进程进入其临界区时，其他试图进入各自临界区的进程必须等待，以保证诸进程互斥地进入临界区；

（3）有限等待。有若干进程要求进入临界区时，应在有限时间内使一进程进入临界区，即它们不应相互等待而谁也不进入临界区；

（4）让权等待。对于等待进入临界区的进程必须释放其占有的 CPU。信号量可以有效地实现进程的同步和互斥。在操作系统中，信号量是一个整数。当信号量大于等于 0 时，代表可供并发进程使用的资源实体数，当信号量小于零时则表示正在等待使用临界区的进程数。建立一个信号量必须说明所建信号量代表的意义和设置初值，以及建立相应的数据结构，以便指向那些等待使用该临界区的进程。

对信号量只能施加特殊的操作：P 操作和 V 操作。P 操作和 V 操作都是不可分割的原子操作，也称为原语。因此，P 原语和 V 原语执行期间不允许中断发生。

P（sem）操作的过程是将信号量 sem 值减 l，若 sem 的值成负数，则调用 P 操作的进程暂停执行，直到另一个进程对同一信号量做 V 操作。V（sem）操作的过程是将信号量 sem 值加 1，若 sem 的值小于等于 0，从相应队列（与 sem 有关的队列）中选一个进程，唤醒它。

一般 P 操作与 V 操作的定义如下所述。

P 操作：

P（sem）{

sem = sem - 1;

if（sem < 0）进程进入等待状态；

else 继续进行；}V 操作：

V（sem）{

sem = sem + 1;

if（sem ≤ 0）唤醒队列中的一个等待进程；

else 继续进行；}

为了保护共享资源（如公共变量），使它们不被多个进程同时访问，就要阻止这些进程同时执行访问这些资源（临界资源）的代码段（临界区）；进程互斥不允许两个以上共享临界资源的并发进程同时进入临界区。利用 P、V 原语和信号量可以方便地解决并发进程对临界区的进程互斥问题。

设信号量 mutex 是用于互斥的信号量，初值为 1，表示没有并发进程使用该临界区。于是各并发进程的临界区可改写成下列形式的代码段：

P（mutex）；

临界区

V（mutex）；

要用 P，V 操作实现进程同步，需要引进私用信号量。私用信号量只与制约进程和被制约进程有关，而不是与整组并发进程相关。与此相对，进程互斥使用的信号量为公用信号量。首先为各并发进程设置私用信号量，然后为私用信号量赋初值，最后利用 P，V 原语和私用信号量规定各进程的执行顺序。

经典同步问题的例子是“生产者－消费者”问题。这要求存后再取，取后再存，即有两个制约关系，为此，需要两个信号量，表示缓冲区中的空单元数和非空单元数，记为 Bufempty 和 Buffull，它们的初值分别是 1 和 0，相应的程序段形式是：

生产者

loop

生产一产品 next；

P（Bufempty）；

next 产品存缓冲区；

V（Buffull）；

endloop

消费者

loop

P（Buffulll）；

V（Bufempty）；

从缓冲区中取产品；使用产品

endloop

5．前趋图

前趋图是一个由结点和有向边构成的有向无循环图。该图通常用于表现事务之间先后顺序的制约关系。图中的每个结点可以表示一个语句、一个程序段或是一个进程，结点间的有向边表示两个结点之间存在的前趋关系。

例：在计算机中，经常采用流水线方式执行指令，每一条指令都可以分解为取指、分析

和执行三步。取指操作为 Ai，分析操作为 Bi 和执行操作为 Ci(i=1,2,3)。如图 2-4 所示为三个任务各程序段并发执行的前驱图。

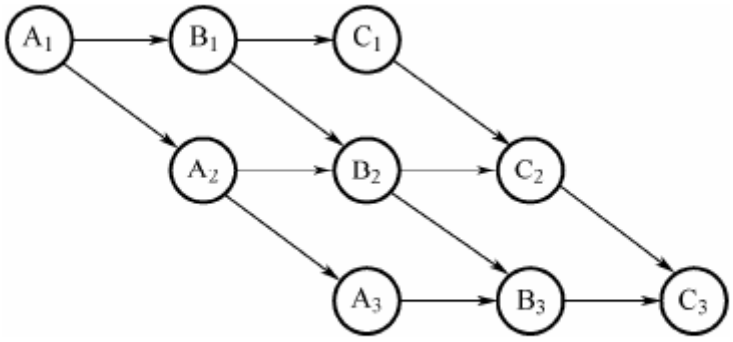


图 2-4　前趋图

图中 A1 没有前趋结点，称为开始结点，它不受任何制约，可以直接执行；而 B1 与 A2 只能在 A1 执行完成之后才能开始，而 B2 必须在 B1 与 A2 完成之后才能开始；C3 没有后继结点，称为终止结点。

在前趋图中，执行先后顺序的制约关系可分为两种：直接制约和间接制约。

直接制约通常是指一个操作中，多个步骤之间的制约关系，也可以说是“同步的进程之间的制约关系”。如图 2-4 所示，A1、B1、C1 是一条指令的取指、分析、执行的三个步骤，所以它们之间的关系是直接制约。

间接制约通常是指多个操作之间相同步骤的制约关系，也可以说是“互斥的进程之间的制约关系”。如图 2-4 所示，A1、A2、A3 之间就存在间接制约的关系。

前趋图的应用广泛，在项目开发中，可用前趋图来分析哪些活动可以并行完成。同时项目管理工具：Pert 图，单（双）代号网络图等都融入了前趋图的思想。

6．进程调度与死锁

进程调度即处理器调度（又称上下文转换），它的主要功能是确定在什么时候分配处理器，并确定分给哪一个进程，即让正在执行的进程改变状态并转入就绪队列的队尾，再由调度原语将就绪队列的队首进程取出，投入执行。

引起进程调度的原因有以下几类：

（1）正在执行的进程执行完毕。

（2）执行中的进程自己调用阻塞原语将自己阻塞起来进入睡眠状态。

（3）执行中的进程调用了 P 原语操作，从而因资源不足而阻塞；或调用 V 原语操作激活了等待资源的进程队列。

（4）在分时系统中，当一进程用完一个时间片。

（5）就绪队列中某进程的优先级变得高于当前执行进程的优先级，也将引起进程调度。

进程调度的方式有两类：剥夺方式与非剥夺方式。所谓非剥夺方式是指，一旦某个作业或进程占用了处理器，别的进程就不能把处理器从这个进程手中夺走，直到该进程自己因调用原语操作而进入阻塞状态，或时间片用完而让出处理机；剥夺方式是指，当就绪队列中有进程的优先级高于当前执行进程的优先级时，便立即发生进程调度，转让处理机。

进程调度的算法是服务于系统目标的策略，对于不同的系统与系统目标，常采用不同的调度算法：

（1）先来先服务（First Come and First Serverd，FCFS）调度算法，又称先进先出（First In and First Out，FIFO）。就绪队列按先来后到原则排队。

（2）优先数调度。优先数反映了进程优先级，就绪队列按优先数排队。有两种确定优先级的方法，即静态优先级和动态优先级。静态优先级是指进程的优先级在进程开始执行前确定，执行过程中不变，而动态优先级则可以在进程执行过程中改变。

（3）轮转法（Round Robin）。就绪队列按 FCFS 方式排队。每个进程执行一次占有处理器时间都不超过规定的时间单位（时间片）若超过，则自行释放自己所占有的 CPU 而排到就绪队列的末尾，等待下一次调度。同时，进程调度程序又去调度当前就绪队列中的第一个进程。

进程管理是操作系统的核心，在进程管理的实现中，如果设计不当，会出现一种尴尬的局面——死锁。

当若干个进程互相竞争对方已占有的资源，无限期地等待，不能向前推进时会造成“死锁”。例如，P1 进程占有资源 R1，P2 进程占有资源 R2，这时，P1 又需要资源 R2，P2 也需要资源 R1，它们在等待对方占有的资源时，又不会释放自己占有的资源，因而使双方都进入了无限等待状态。

死锁是系统的一种出错状态，它不仅会浪费大量的系统资源，甚至还会导致整个系统的崩溃，所以死锁是应该尽量预防和避免的。

（1）死锁条件。产生死锁的主要原因是供共享的系统资源不足，资源分配策略和进程的推进顺序不当。系统资源既可能是可重复使用的永久性资源，也可能是消耗性的临时资源。产生死锁的必要条件是：互斥条件、保持和等待条件、不剥夺条件和环路等待条件。

（2）解决死锁的策略。处于死锁状态的进程不能继续执行但又占用了系统资源，从而阻碍其他作业的执行。

解决死锁有两种策略：一种是在死锁发生前采用的预防和避免策略；另一种是在死锁发生后采用的检测与恢复策略。

死锁的预防主要是通过打破死锁产生的 4 个必要条件之一来保证不会产生死锁。采用的死锁预防策略通常有资源的静态分配法或有序分配法，它们分别打破了资源动态分配条件和循环等待条件，因此不会发生死锁。但这样做会大大降低系统资源的利用率和进程之间的并行程度。

死锁避免策略，则是在系统进行资源分配时，先执行一个死锁避免算法（典型的如银行家算法），以保证本次分配不会导致死锁发生。由于资源分配很频繁，因此死锁避免策略要耗费大量的 CPU 和时间。

希赛教育专家提示：实际上，系统出现死锁的概率很小，故从系统所花的代价上看，采用死锁发生后的检测与恢复策略要比采用死锁发生前的预防与避免策略代价小一些。

2.2.2 存储管理

存储器是计算机系统中最重要的资源之一。因为任何程序和数据以及各种控制用的数据结构都必须占有一定的存储空间，因此，存储管理直接影响系统性能。

存储器由内存和外存组成。内存是由系统实际提供的存储单元（常指字节）组成的一个连续地址空间，处理器可直接存取。外存（辅存）是指软盘、硬盘、光盘和磁带等一些外部存储部件，常用来存放暂不执行的程序和数据。处理器不能直接访问外存，需通过启动 I/O（Input/Output，输入/输出）设备才能进行内存、外存交换，其访问速度慢，但价格便宜，常用作内存的后援设备。

内存大小由系统硬件决定，存储容量受到实际存储单元的限制。虚拟存储器（简称虚存）不考虑实际内存的大小和数据存取的实际地址，只考虑相互有关的数据之间的相对位置，其容量由计算机地址的位数决定。

系统中内存的使用一般分成两部分，一部分为系统空间，存放操作系统本身及相关的系统程序；另一部分为用户空间，存放用户的程序和数据。

存储管理主要是指对内存储器的管理，负责对内存的分配和回收、内存的保护和内存的扩充。存储管理的目的是尽量提高内存的使用效率。存储管理的机制经历了多次变迁，由以前的单一连续区管理到分区存储管理再发展为段页式管理。目前前两种技术已逐步被淘汰，下面我们将详细解读段页式存储管理。

1．页式存储管理

分页的基本思想是把程序的逻辑空间和内存的物理空间按照同样的大小划分成若干页面，并以页面为单位进行分配。在页式存储管理中，系统中虚地址是一个有序对（页号，位移）。系统为每一个进程建立一个页表，其内容包括进程的逻辑页号与物理页号的对应关系、状态等。

页式系统的动态地址转换是这样进行的：当进程运行时，其页表的首地址已在系统的动态地址转换机构中的基本地址寄存器中。执行的指令访问虚存地址（p，d）时，首先根据页号 p 查页表，由状态可知，这个页是否已经调入内存。若已调入内存，则得到该页的内存位置 p'，然后，与页内相对位移 d 组合，得到物理地址 r。如果该页尚未调入内存，则产生缺页中断，以装入所需的页，如图 2-5 所示。

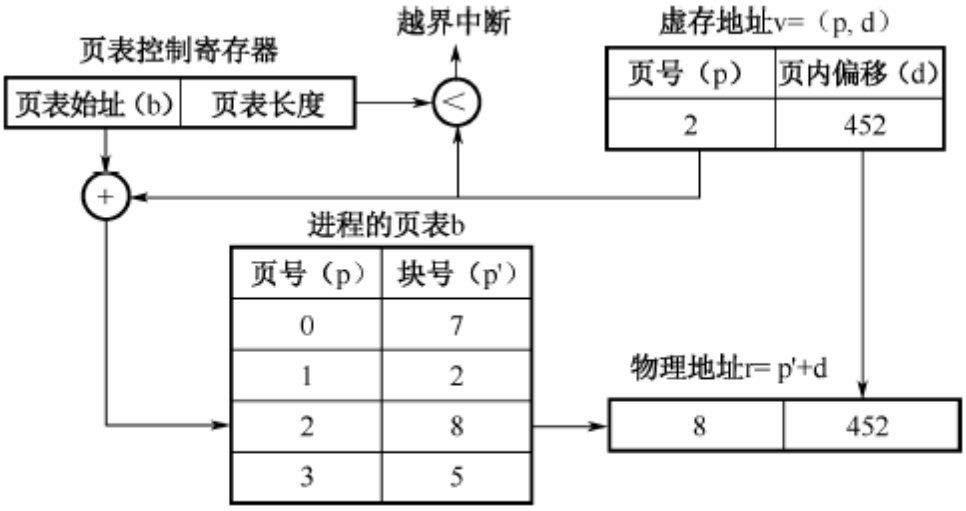


图 2-5　页式存储管理地址转换示意图

页式虚拟存储管理是在页式存储管理的基础上实现虚拟存储器的。首先把作业信息作为副本存放在磁盘上，作业执行时，把作业信息的部分页面装入内存储器，作业执行时若所访问的页面已在内存中，则按页式存储管理方式进行地址转换，得到欲访问的内存绝对地址，若欲访问的页面不在内存中，则产生一个“缺页中断”，由操作系统把当前所需的页面装入内存储器中。

为此，在装入作业时，就应在该作业的页表中指出哪些页已在内存储器中，哪些页还没有装入内存。可用一个标志位指示对应页是否在内存储器，可假设标志位为 1 表示该页在内存，而标志位为 0 表示该页尚未装入内存。为了能方便地从磁盘上找到作业信息的副本，故在页表中还可指出每一页副本在磁盘上的位置。

当要装入一个当前需要的页面时，如果内存储器中无空闲块，则可选择一个已在内存储器中的页面，把它暂时调出内存。若在执行中该页面被修改过，则把该页信息重新写回到磁盘上，否则不必重新写回磁盘。当一页被暂时调出内存后，让出的内存空间用来存放当前需要使用的页面。以后再使用被调出的页面时，可用同样的方法调出另一个页面而将其再装入内存。页面被调出或装入之后都要对页表中的相应表目做修改。

当内存中无空闲块时，为了装入一个页面而必须按某种算法从已在内存的页中选择一页，将它暂时调出内存，让出内存空间以存放所需装入的页面，这个工作称为“页面调度”。如何选择调出的页面是很重要的，如果采用了一个不合适的算法，就会出现这样的现象：刚被调出的页面又立即要用，因而又要把它装入，而装入不久又被选中调出，调出不久又被装入，如此反复，使调度非常频繁。这种现象称为“抖动”。一个好的调度算法应减少或避免抖动现象。常用的页面调度算法有：

（1）最优（OPT）算法。选择不再使用或最远的将来才被使用的页，这是理想的算法，但是难以实现，常用于淘汰算法的比较。

（2）随机（RAND）算法。随机地选择被淘汰的页，开销小，但是可能选中立即就要访问的页。

（3）先进先出算法。选择在内存驻留时间最长的页似乎合理，但可能淘汰掉频繁使用的页。另外，使用 FIFO 算法时，在未给予进程分配足够的页面数时，有时会出现给予进程的页面数增多，缺页次数反而增加的异常现象。FIFO 算法简单，易实现。可以把装入内存储器的那些页的页号按进入的先后顺序排成队列，每次总是调出队首的页，当装入一个新页后，把新页的页号排到队尾。

（4）最近最少使用（Least Recently Used，LRU）算法。选择离当前时间最近的一段时间内使用得最少的页。这个算法的主要出发点是，如果某个页被访问了，则它可能马上就要被访问；反之，如果某个页长时间未被访问，则它在最近一段时间也不会被访问。

2．段式存储管理

段式存储管理与页式存储管理相似。分段的基本思想是把用户作业按逻辑意义上有完整意义的段来划分，并以段为单位作为内外存交换的空间尺度。

一个作业是由若干个具有逻辑意义的段（如主程序、子程序、数据段等）组成。分段系统中，容许程序（作业）占据内存中许多分离的分区。每个分区存储一个程序分段。这样，每个作业需要几对界限地址寄存器，判定访问地址是否越界也就更困难了。在分段存储系统中常常利用存储保护键实现存储保护。分段系统中虚地址是一个有序对（段号，位移）。系

统为每个作业建立一个段表，其内容包括段号、段长、内存起始地址和状态等。状态指出这个段是否已调入内存，即内存起始地址指出这个段，状态指出这个段的访问权限。

分段系统的动态地址转换是这样进行的：进程执行时，其段表的首地址已在基本地址寄存器中，执行的指令访问虚存（s，d）（取指令或取操作数）时，首先根据段号 s 查段表，若段已经调入内存，则得到该段的内存起始地址，然后与段内相对地址（段内偏移量 d）相加，得到实地址。如果该段尚未调入内存，则产生缺段中断，以装入所需要的段。段式存储与页式存储的地址转换方式类似，参看图 1-6。

段式虚拟存储管理仍然以段式存储管理为基础，为用户提供比内存实际容量大的虚拟空间。段式虚拟存储管理把作业中的各个分段信息都保留在磁盘上，当作业可以投入执行时，做如下操作：

（1）首先把当前需要的一段或几段装入内存。

（2）作业执行时，如果要访问的段已经在内存，则按照“段式存储管理”中的方式进行地址转换；如果要访问的段不在内存中，则产生一个“缺段中断”，由操作系统把当前需要的段装入内存。

因此，在段表中应增设段是否在内存的标志以及各段在磁盘上的位置，已在内存中的段仍要指出该段在内存中的起始地址和占用内存区长度。

作业执行要访问的段时，由硬件的地址转换机构查段表。若该段在内存中，则立即把逻辑地址转换成绝对地址；若该段不在内存中，则形成“缺段中断”，由操作系统处理这个中断。

处理的办法是，查内存分配表，找出一个足够大的连续区以容纳该分段，如果找不到足够大的连续区则检查空闲区的总和，若空闲区总和能满足该段要求，那么进行适当移动将分散的空闲区集中；若空闲区总和不能满足该段要求，可把内存中的一段或几段调出，然后把当前要访问的段装入内存中。段被移动、调出和装入后都要对段表中的相应表目做修改。新的段被装入后应让作业重新执行被中断的指令，这时就能找到要访问的段，也可以继续执行下去。

3．段页式存储管理

段页式管理是段式和页式两种管理方法结合的产物，综合了段式组织与页式组织的特点，根据程序模块分段，段内再分页，内存被分划成定长的页。段页式系统中虚地址形式是（段号、页号、页内偏移），如图 2-6 所示。系统为每个进程建立一个段表，为每个段建立一个页表。段页式管理采用段式分配、页式使用的方法，便于动态连接和存储的动态分配。这种

存储管理能提高内存空间的利用率。

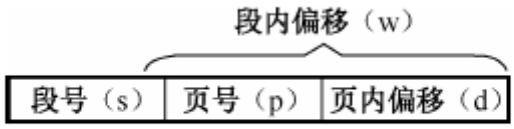


图 2-6　段页式存储管理地址

段式虚拟管理还是以段为单位分配内存空间，整段的调出、装入，有时还要移动，这些都增加了系统的开销。如果按段页式存储管理的方式，把每一段再分成若干页面，那么，每一段不必占用连续的存储空间；甚至当内存块不够时，可只将一段中的部分页面装入内存，这种管理方式称为“段页式虚拟存储管理”。

段页式虚拟存储管理为每一个装入内存的作业建立一张段表，还要为每一段建立页表。段表中指出该段的页表存放位置及长度，页表中应指出该段的各页在磁盘上的位置以及页是否在内存中，若在内存中则填上占用的内存块号。作业执行时按段号查段表，找到相应的页表再根据页号查页表，由标志位判定该页是否已在内存，若是，则进行地址转换；否则进行页面调度。地址转换过程如图 2-7 所示。

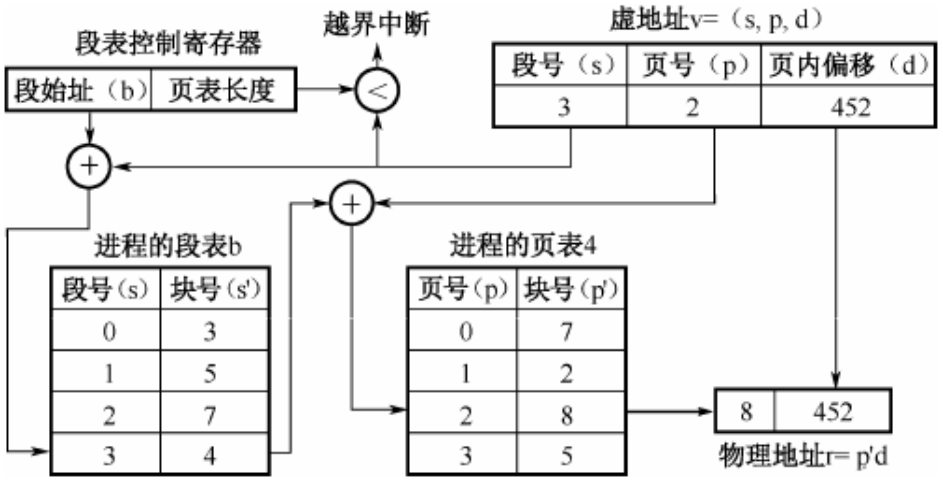


图 2-7　段页式存储管理中的动态地址转换示意图

段页式虚拟存储管理结合了段式和页式的优点，但增加了设置表格（段表、页表）和查表等开销，段页式虚拟存储器一般只在大型计算机系统中使用。

2.2.3 设备管理

在计算机系统中，除了处理器和内存之外，其他的大部分硬设备称为外部设备。它包括输入/输出设备，辅存设备及终端设备等。这些设备种类繁多，特性各异，操作方式的差异很大，从而使操作系统的设备管理变得十分繁杂。在架构师考试中，设备管理需要掌握的知识内容较少，主要为两个方面：

1．数据传输控制方式

设备管理的主要任务之一是控制设备和内存或 CPU 之间的数据传送，本节介绍几种常用的数据传送控制方式。

选择和衡量控制方式的原则如下：

（1）数据传送速度足够高，能满足用户的需要但又不丢失数据。

（2）系统开销小，所需的处理控制程序少。

（3）能充分发挥硬件资源的能力，使得 I/O 设备尽量处于使用状态中，而 CPU 等待时间少。

外围设备和内存之间常用的数据传送控制方式主要有以下几种：

（1）程序控制方式。处理器启动数据传输，然后等设备完成。

（2）中断方式。程序控制方式不能实现并发。中断方式的数据传输过程是这样的，进程启动数据传输（如读）后，该进程放弃处理器，当数据传输完成，设备控制器产生中断请求，中断处理程序对数据传输工作处理之后，让相应进程成为就绪状态。以后，该进程就可以得到所需要的数据。

（3）直接存储访问（Direct Memory Access，DMA）方式。指外部设备和内存之间开辟直接的数据交换通路。除了控制状态寄存器和数据缓冲寄存器外，DMA 控制器中还包括传输字节计数器、内存地址寄存器等。DMA 方式采用窃取（或挪用）处理器的工作周期和控制总线而实现辅助存储器和内存之间的数据交换。有的 DMA 方式也采用总线浮起方式传输大批量数据。

（4）通道方式。通道又称为输入/输出处理器（Input/Output Processor，IOP），可以独立完成系统交付的输入/输出任务，通过执行自身的输入/输出专用程序（称通道程序）进行内存和外设之间的数据传输。主要有 3 种通道：字节多路通道、选择通道和成组多路通道。

2．虚设备与 SPOOLING 技术

采用假脱机技术，可以将低速的独占设备改造成一种可共享的设备，而且一台物理设备

可以对应若干台虚拟的同类设备。假脱机（Simultaneous Peripheral Operation On Line，SPOOLING）的意思是外部设备同时联机操作，又称为假脱机输入/输出操作，采用一组程序或进程模拟一台输入/输出处理器。

SPOOLING 系统的组成如图 2-8 所示。该技术利用了专门的外围控制机将低速 I/O 设备上的数据传送到高速设备上，或者相反。但是当引入多道程序后，完全可以利用其中的一道程序来模拟脱机输入时的外围控制机的功能，把低速的 I/O 设备上的数据传送到高速磁盘上；再利用另一道程序来模拟脱机输出时外围控制机的功能，把高速磁盘上的数据传送到低速的 I/O 设备上。这样便可以在主机的控制下实现脱机输入、输出的功能。此时的外围操作与 CPU 对数据的处理同时进行。

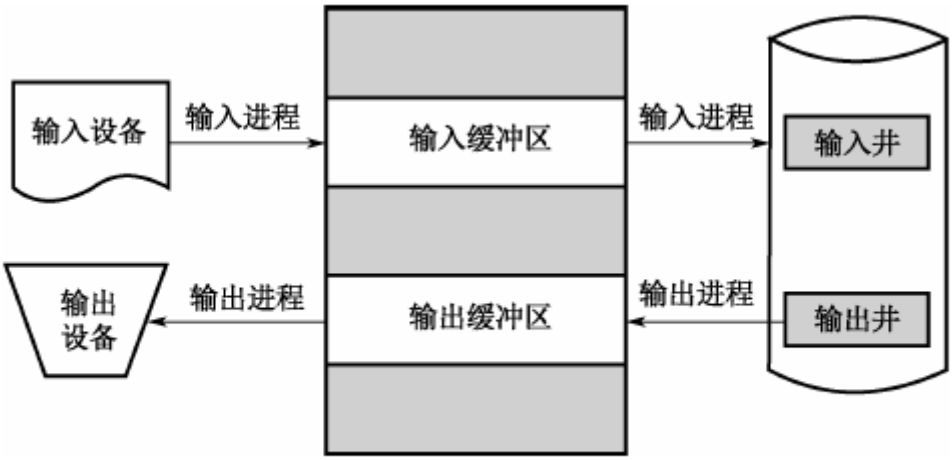


图 2-8　SPOOLING 系统示意图

采用假脱机技术，可以将低速的独占设备改造成一种可共享的设备，而且一台物理设备可以对应若干台虚拟的同类设备。SPOOLING 系统必须有高速、大容量并且可随机存取的外存（例如，磁盘或磁鼓）支持。

希赛教育专家提示：在现代计算机系统中，还可以用一台设备来模拟自身。例如，常见的多窗口技术，即在一个终端上开多个窗口，每个窗口可以独立地进行显示，以监视用户不同任务的执行情况。这是通过缩小显示区域、平铺或重叠显示来模拟多个显示器的。

2.2.4 文件管理

操作系统对计算机的管理包括两个方面：硬件资源和软件资源。硬件资源的管理包括 CPU 的管理、存储器的管理、设备管理等，主要解决硬件资源的有效和合理利用问题。

软件资源包括各种系统程序、各种应用程序、各种用户程序，也包括大量的文档材料、

库函数等。每一种软件资源本身都是具有一定逻辑意义的相关信息的集合，在操作系统中它们以文件形式存储。

计算机系统的重要作用之一是能快速处理大量信息，因此数据的组织、存取和保护成为一个极重要的内容。文件系统是操作系统中组织、存取和保护数据的一个重要部分。

文件管理的功能包括：建立、修改、删除文件；按文件名访问文件；决定文件信息的存放位置、存放形式及存取权限；管理文件间的联系及提供对文件的共享、保护和保密等。允许多个用户协同工作又不引起混乱。文件的共享是指一个文件可以让多个用户共同使用，它可以减少用户的重复性劳动，节省文件的存储空间，减少输入/输出文件的次数等。文件的保护主要是为防止由于错误操作而对文件造成的破坏。文件的保密是为了防止未经授权的用户对文件进行访问。

文件的保护、保密实际上是用户对文件的存取权限控制问题。一般为文件的存取设置两级控制：第 1 级是访问者的识别，即规定哪些人可以访问；第 2 级是存取权限的识别，即有权参与访问者可对文件执行何种操作。

1．文件的逻辑结构

文件的结构是指文件的组织形式，从用户角度所看到的文件组织形式，称为文件的逻辑结构。

文件的逻辑组织是为了方便用户使用。一般文件的逻辑结构可以分为两种：无结构的字符流文件和有结构的记录文件。记录文件由记录组成，即文件内的信息划分成多个记录，以记录为单位组织和使用信息。

记录文件有顺序文件、索引顺序文件、索引文件和直接文件。

（1）顺序文件。大多数文件是顺序文件。顺序文件的记录定长，记录中的数据项的类型长度与次序固定，一般还有一个可以唯一标识记录的数据项，称为键（key），记录是按键值的约定次序组织的。顺序文件常用于批处理应用，对于查询或更新某个记录的处理性能不太好。

（2）索引顺序文件。索引顺序文件是基于键的约定次序组织的，而且维护键的索引和溢出区域。键的索引也可以是多级索引。索引顺序文件既适用于交互方式应用，也适用于批处理方式应用。

（3）索引文件。索引文件是基于记录的一个键数据项组织的。许多应用需按照别的数据项访问文件，为此，常采用索引文件方法，即对主文件中的记录按需要的数据项（一个或几个）建索引，索引文件本身是顺序文件组织。

（4）直接文件。直接文件又称哈希（Hash）文件。记录以它们在直接访问存储设备上的物理地址直接（随机地）访问。直接文件常用 于需要高速访问文件而且每次仅访问一条记录的应用中。

2．文件的物理结构

文件的物理结构是指文件在存储设备上的存放方法。文件的物理结构侧重于提高存储器的利用效率和降低存取时间。文件的存储设备通常划分为大小相同的物理块，物理块是分配和传输信息的基本单位。文件的物理结构涉及文件存储设备的组块策略和文件分配策略，决定文件信息在存储设备上的存储位置。常用的文件分配策略有：

（1）顺序分配（连续分配）。这是最简单的分配方法。在文件建立时预先分配一组连续的物理块，然后，按照逻辑文件中的信息（或记录）顺序，依次把信息（或记录）按顺序存储到物理块中。这样，只需知道文件在文件存储设备上的起始位置和文件长度，就能进行存取，这种分配方法适合于顺序存取，在连续存取相邻信息时，存取速度快。其缺点是在文件建立时必须指定文件的信息长度，以后不能动态增长，一般不宜用于需要经常修改的文件。

（2）链接分配（串联分配）。这是按单个物理块逐个进行的。每个物理块中（一般是最后一个单元）设有一个指针，指向其后续连接的下一个物理块的地址，这样，所有的物理块都被链接起来，形成一个链接队列。在建立链接文件时，不需要指定文件的长度，在文件的说明信息中，只需指出该文件的第一个物理块块号，而且链接文件的文件长度可以动态地增长。只调整物理块间的指针就可以插入或删除一个信息块。

链接分配的优点是可以解决存储器的碎片问题，提高存储空间利用率。由于链接文件只能按照队列中的链接指针顺序查找，因此搜索效率低，一般只适用于顺序访问，不适用于随机存取。

（3）索引分配。这是另一种对文件存储不连续分配的方法。采用索引分配方法的系统，为每一个文件建立一张索引表，索引表中每一表项指出文件信息所在的逻辑块号和与之对应的物理块号。

索引分配既可以满足文件动态增长的要求，又可以方便而迅速地实现随机存取。对一些大的文件，当索引表的大小超过一个物理块时，会发生索引表的分配问题。一般采用多级（间接索引）技术，这时在由索引表指出的物理块中存放的不是文件存放处而是存放文件信息的物理块地址。这样，如果一个物理块能存储 n 个地址，则一级间接索引将使可寻址的文件长度变成 n2 块，对于更大的文件可以采用二级甚至三级间接索引（例如，UNIX 操作系统采用三级索引结构，如图 2-9 所示）。

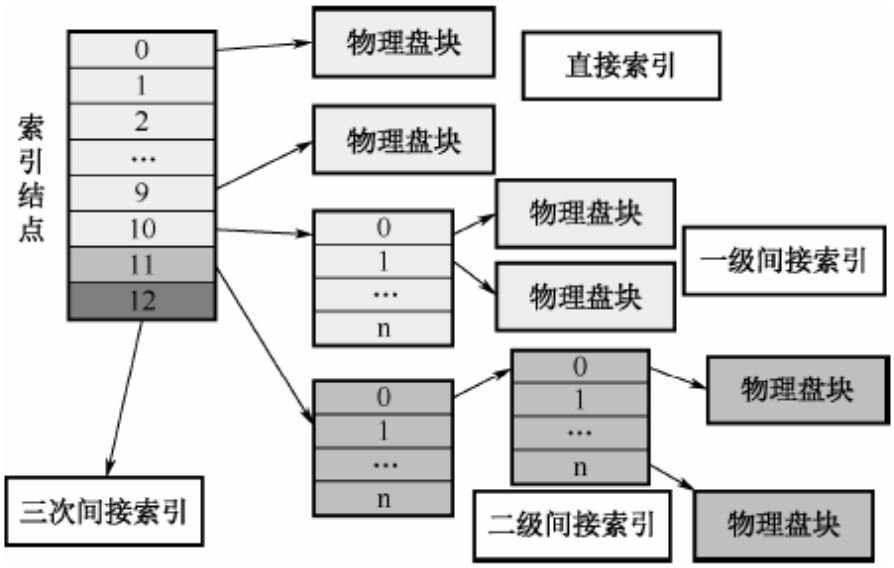


图 2-9　索引结构图

索引文件的优点是既适用于顺序存取，又适用于随机存取。缺点是索引表增加了存储 空间的开销。另外，在存取文件时需要访问两次磁盘，一次是访问索引表，另一次是根据索引表提供的物理块号访问文件信息。为了提高效率，一种改进的方法是，在对某个文件进行操作之前，预先把索引表调入内存。这样，文件的存取就能直接从内存的索引表中确定相应的物理块号，从而只需要访问一次磁盘。

3．文件存储设备管理

文件存储设备管理，就是操作系统要有效地进行存储空间的管理。由于文件存储设备是分成许多大小相同的物理块，并以块为单位交换信息，因此，文件存储设备的管理实质上是对空闲块的组织和管理问题。它包括空闲块的组织，空闲块的分配与空闲块的回收等问题。有 3 种不同的空闲块管理方法，它们分别是索引法、链接法和位示图法。

（1）索引法。索引法把空闲块作为文件并采用索引技术。为了有效，索引对应于一个 或由几个空闲块构成的空闲区。这样，磁盘上每一个空闲块区都对应于索引表中一个条目，这个方法能有效地支持每一种文件分配方法。

（2）链接法。链接法使用链表把空闲块组织在一起，当申请者需要空闲块时，分配程 序从链首开始摘取所需的空闲块。反之，管理程序把回收的空闲块逐个挂入队尾，这个方 法适用于每一种文件分配方法。空闲块的链接方法可以按释放的先后顺序链接，也可以按 空闲块区的大小顺序链接。后者有利于获得连续的空闲块的请求，但在分配请求和回收空闲块时系统开销多一点。

（3）位示图法。该方法是在外存上建立一张位示图（Bitmap），记录文件存储器的使 用情况。每一位仅对应文件存储器上的一个物理块，取值 0 和 1 分别表示空闲和占用。文件存储器上的物理块依次编号为：0、1、2、…。假如系统中字长为 32 位，有 4096 个物理块，那么在位示图中的第 1 个字对应文件存储器上的 0、1、2、…、31 号物理块；第 2 个字对应文件存储器上的 32、33、34、…、63 号物理块；第 128 字对应文件存储器上的 4064、4065、…、4095 号物理块。这样位示图的大小为 32 字。

位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，如图 2-10 所示。当其值为“0”时，表示对应的盘块空闲；为“1”时表示已分配。由所有盘块对应的位构成一个集合，称为位示图。位示图也可描述为一个二维数组 map：Varmap:array[1．…m,1．…n]of bit;

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
第1个字	1	1	0	0	0	1	1	1	0	0	1	0	1	1	1	0
第2个字	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
第3个字	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
第4个字																
⋮																
第15个字																

图 2-10　位示图

4．树型目录结构

在计算机的文件系统中，一般采用树型目录结构。在树型目录结构中，树的根结点为根目录，数据文件作为树叶，其他所有目录均作为树的结点。

根目录隐含于一个硬盘的一个分区中，根目录在最顶层。它包含的子目录是一级子目录。每一个一级子目录又可以包含若干二级子目录，…，这样的组织结构就叫作目录树。

当前盘和当前目录是系统默认的操作对象。如果用户没有指明操作对象，系统就将用户命令指向当前盘和当前目录。

路径是指从根目录或者当前目录开始到访问对象（目录或者文件），在目录树中路经的所有目录的序列。例如“c:\dos\lmouse\mouse”就是 Windows 系统中的一条路径。在树型目录结构中，从根目录到任何数据文件之间，只有一条唯一的通路，从树根开始，把全部目录文件名与数据文件名，依次用“/”（UNIX/Linux 系统）或“\”（Windows 系统）连接起来，构成该数据文件的路径名，且每个数据文件的路径名是唯一的。这样，便可以解决文件重名问题。

从树根开始的路径为绝对路径，如果文件系统有很多级时，使用不是很方便，所以引入相对路径，即从当前目录开始，再逐级通过中间的目录文件，最后到达所要访问的数据文件。

绝对路径给出文件或目录位置的完全描述，通常由层次结构的顶端开始（根目录），通常第一个字符是“/”（UNIX/Linux 系统）或者是盘符（Windows 系统）。相对路径通常由目录结构中的当前位置开始，一般都比绝对路径要短。

父目录是指当前路径的上一层目录。每个目录下都有代表当前目录的“.”文件和代表当前目录父目录的“..”文件，相对路径名一般就是从“..”开始的。

2.2.5 作业管理

从用户的角度看，作业是系统为完成一个用户的计算任务（或一次事务处理）所做的工作总和。例如，对于用户编制的源程序，需经过对源程序的编译、连接编辑或连接装入及运行产生计算结果。这其中的每一个步骤，常称为作业步，作业步的顺序执行即完成了一个作业。

从系统的角度看，作业则是一个比程序更广的概念。它由程序、数据和作业说明书组成。系统通过作业说明书控制文件形式的程序和数据，使之执行和操作。而且，在批处理系统中，作业是占据内存的基本单位。

用户的作业可以通过直接的方式，由用户自己按照作业步顺序操作；也可以通过间接的方式，由用户率先编写的作业步依次执行的说明，一次交给操作系统，由系统按照说明依次处理。前者称为联机方式，后者称为脱机方式。

1．作业状态及其转换

一个作业从交给计算机系统到执行结束退出系统，一般都要经历提交、后备、执行和完成 4 个状态。其状态转换如图 2-11 所示。

（1）提交状态。作业由输入设备进入外存储器（也称输入井）的过程称为提交状态。处于提交状态的作业，其信息正在进入系统。

（2）后备状态。当作业的全部信息进入外存后，系统就为该作业建立一个作业控制块（Job Control Block，JCB）。系统通过 JCB 感知作业的存在。JCB 主要内容包括作业名、作业状态、资源要求、作业控制方式、作业类型及作业优先权等。

（3）执行状态。一个后备作业被作业调度程序选中而分配了必要的资源并进入了内存，作业调度程序同时为其建立了相应的进程后，该作业就由后备状态变成了执行状态。

（4）完成状态。当作业正常运行结束，它所占用的资源尚未全部被系统回收时的状态为完成状态。

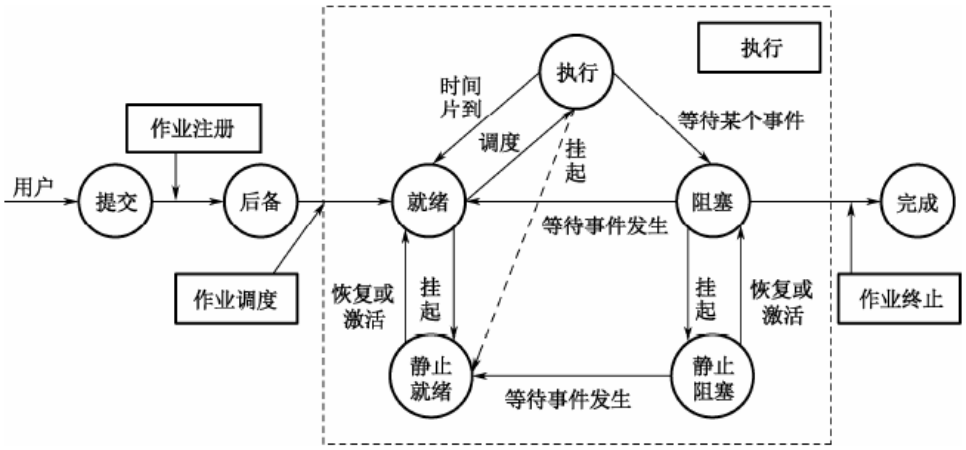


图 2-11　作业的状态及其转换

2．用户接口

用户接口也称为用户界面，其含义有两种，一种是指用户与操作系统交互的途径和通道，即操作系统的接口；另一种是指这种交互环境的控制方式，即操作环境。

（1）操作系统的接口。操作系统的接口又可分成命令接口和程序接口。命令接口包含键盘命令和作业控制命令；程序接口又称为编程接口或系统调用，程序经编程接口请求系统服务，即通过系统调用程序与操作系统通信。系统调用是操作系统提供给编程人员的唯一接口。系统调用对用户屏蔽了操作系统的具体动作而只提供有关功能。系统调用大致分为设备管理、文件管理、进程控制、进程通信和存储管理等。

（2）操作环境。操作环境支持命令接口和程序接口，提供友好的、易用的操作平台。操作系统的交互界面已经从早期的命令驱动方式，发展到菜单驱动方式、图符驱动方式和视窗操作环境。

第 3 章　数据库系统

随着应用系统的规模越来越大，现在的系统开发大部分都是基于数据库的应用，因此，作为一名系统架构设计师，要熟练地掌握数据库系统的设计方法和技术。

本章在宏观上就系统架构设计师必须要掌握的内容进行介绍，有关细节方面的知识，如果读者感兴趣，可以参考数据库专业教程。

3.1 数据库管理系统的类型

数据库管理系统的类型通常有多个分类标准。如按数据模型分类、按用户数分类、按数据库分布站点分类等。但我们需要了解的，主要还是按数据模型分类。

当前，许多商业 DBMS 中所用的主要数据模型仍是关系数据模型。有些商业系统中实现了对象数据模型，但未得到广泛使用。近几年随着 NoSQL 技术的兴起，也产生了一些新的数据模型。目前常见的 DBMS 按数据模型划分，包括：关系型 DBMS、文档型 DBMS、键值型 DBMS、对象型 DBMS 等。

3.2 数据库模式与范式

数据库模式与范式是数据库系统中的两个重要概念，是进行数据库设计的基础。

3.2.1 数据库的结构与模式

数据库技术中采用分级的方法将数据库的结构划分为多个层次。最著名的是美国 ANSI/SPARC 数据库系统研究组 1975 年提出的三级划分法，如图 3-1 所示。

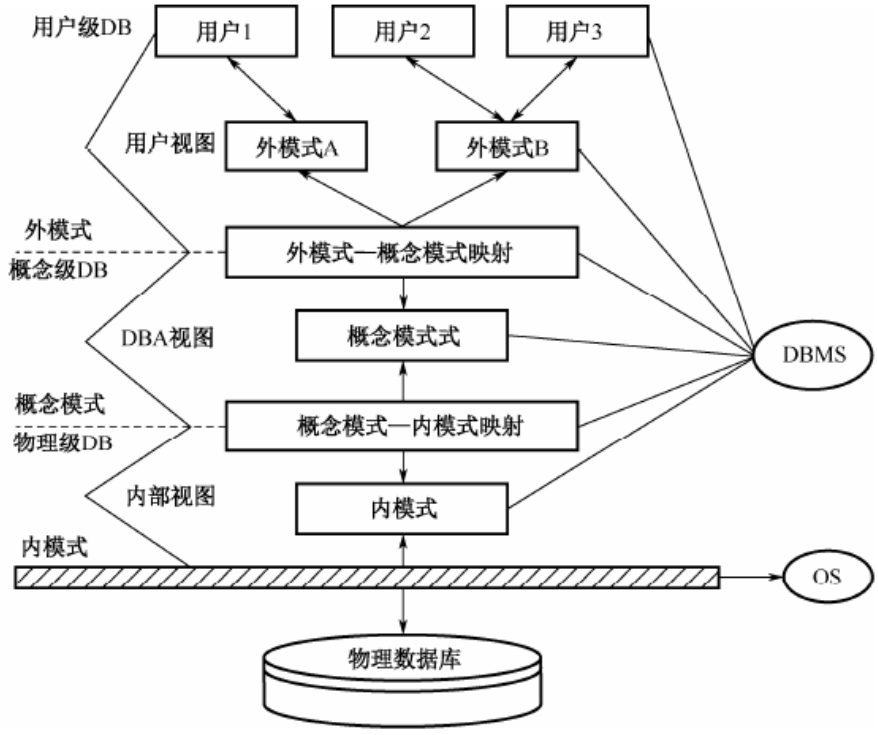


图 3-1　数据库系统结构层次

1．三级抽象

数据库系统划分为三个抽象级：用户级、概念级、物理级。

（1）用户级数据库。用户级数据库对应于外模式，是最接近用户的一级数据库，是用户可以看到和使用的数据库，又称用户视图。用户级数据库主要由外部记录组成，不同的用户视图可以互相重叠，用户的所有操作都是针对用户视图进行的。

（2）概念级数据库。概念级数据库对应于概念模式，介于用户级和物理级之间，是所有用户视图的最小并集，是数据库管理员可看到和使用的数据库，又称 DBA（DataBase Administrator，数据库管理员）视图。概念级数据库由概念记录组成，一个数据库可有多个不同的用户视图，每个用户视图由数据库某一部分的抽象表示所组成。一个数据库应用系统只存在一个 DBA 视图，它把数据库作为一个整体的抽象表示。概念级模式把用户视图有机地结合成一个整体，综合平衡考虑所有用户要求，实现数据的一致性、最大限度降低数据冗余、准确地反映数据间的联系。

（3）物理级数据库。物理级数据库对应于内模式，是数据库的低层表示，它描述数据的实际存储组织，是最接近于物理存储的级，又称内部视图。物理级数据库由内部记录组成，物理级数据库并不是真正的物理存储，而是最接近于物理存储的级。

2．三级模式

数据库系统的三级模式为外模式、概念模式、内模式。

（1）概念模式。概念模式（模式、逻辑模式）用以描述整个数据库中数据库的逻辑结构，描述现实世界中的实体及其性质与联系，定义记录、数据项、数据的完整性约束条件及记录之间的联系，是数据项值的框架。

数据库系统概念模式通常还包含有访问控制、保密定义、完整性检查等方面的内容，以及概念/物理之间的映射。

概念模式是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。一个数据库只有一个概念模式。

外模式是数据库用户（包括程序员和最终用户）能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。一个数据库可以有多个外模式。一个应用程序只能使用一个外模式。

（2）外模式。外模式（子模式、用户模式）用以描述用户看到或使用的那部分数据的逻辑结构，用户根据外模式用数据操作语句或应用程序去操作数据库中的数据。外模式主要描述组成用户视图的各个记录的组成、相互关系、数据项的特征、数据的安全性和完整性约束条件。

（3）内模式。内模式是整个数据库的最低层表示，不同于物理层，它假设外存是一个无限的线性地址空间。内模式定义的是存储记录的类型、存储域的表示以及存储记录的物理顺序，指引元、索引和存储路径等数据的存储组织。

内模式是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。一个数据库只有一个内模式。

内模式、模式和外模式之间的关系如下：

（1）模式是数据库的中心与关键；

（2）内模式依赖于模式，独立于外模式和存储设备；

（3）外模式面向具体的应用，独立于内模式和存储设备；

（4）应用程序依赖于外模式，独立于模式和内模式。

3．两级独立性

数据库系统两级独立性是指物理独立性和逻辑独立性。三个抽象级间通过两级映射（外模式—模式映射，模式—内模式映射）进行相互转换，使得数据库的三级形成一个统一的整体。

（1）物理独立性。物理独立性是指用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的。当数据的物理存储改变时，应用程序不需要改变。

物理独立性存在于概念模式和内模式之间的映射转换，说明物理组织发生变化时应用程序的独立程度。

（2）逻辑独立性。逻辑独立性是指用户的应用程序与数据库中的逻辑结构是相互独立的。当数据的逻辑结构改变时，应用程序不需要改变。

逻辑独立性存在于外模式和概念模式之间的映射转换，说明概念模式发生变化时应用程序的独立程度。

希赛教育专家提示：逻辑独立性比物理独立性更难实现。

3.2.2 数据模型

数据模型主要有两大类，分别是概念数据模型（实体—联系模型）和基本数据模型（结构数据模型）。

概念数据模型是按照用户的观点来对数据和信息建模，主要用于数据库设计。概念模型主要用实体—联系方法（Entity-Relationship Approach）表示，所以也称 E-R 模型。

基本数据模型是按照计算机系统的观点来对数据和信息建模，主要用于 DBMS 的实现。基本数据模型是数据库系统的核心和基础。基本数据模型通常由数据结构、数据操作和完整性约束三部分组成。其中数据结构是对系统静态特性的描述，数据操作是对系统动态特性的描述，完整性约束是一组完整性规则的集合。

常用的基本数据模型有层次模型、网状模型、关系模型和面向对象模型。

层次模型用树形结构表示实体类型及实体间的联系。层次模型的优点是记录之间的联系通过指针来实现，查询效率较高。层次模型的缺点是只能表示 1:n 联系，虽然有多种辅助手段实现 m:n 联系，但比较复杂，用户不易掌握。由于层次顺序的严格和复杂，导致数据的查询和更新操作很复杂，应用程序的编写也比较复杂。

网状模型用有向图表示实体类型及实体间的联系。网状模型的优点是记录之间的联系通过指针实现，m:n 联系也容易实现，查询效率高。其缺点是编写应用程序的过程比较复杂，程序员必须熟悉数据库的逻辑结构。

关系模型用表格结构表达实体集，用外键表示实体间的联系。其优点有：

（1）建立在严格的数学概念基础上；

（2）概念（关系）单一，结构简单、清晰，用户易懂易用；

（3）存取路径对用户透明，从而数据独立性、安全性好，简化数据库开发工作。

3.2.2 关系代数

关系代数的基本运算主要有并、交、差、笛卡尔积、选择、投影、连接和除法运算。

（1）并。计算两个关系在集合理论上的并集，即给出关系 R 和 S（两者有相同元/列数），R∪S 的元组包括 R 和 S 所有元组的集合，形式定义如下：

$$R \bigcup S \equiv \{t \mid t \in R \vee t \in S\}$$

式中 t 是元组变量（下同）。显然，R∪S=S∪R。

（2）差。计算两个关系的区别的集合，即给出关系 R 和 S（两者有相同元/列数），R-S 的元组包括 R 中有而 S 中没有的元组的集合，形式定义如下：

$$R - S \equiv \{t \mid t \in R \wedge t \notin S\}$$

（3）交。计算两个关系集合理论上的交集，即给出关系 R 和 S（两者有相同元/列数），R∩S 的元组包括 R 和 S 相同元组的集合，形式定义如下：

$$R \bigcap S \equiv \{t \mid t \in R \wedge t \in S\}$$

显然，R∩S = R-(R-S)　和　R∩S = S-(S-R)成立。

（4）笛卡尔积。计算两个关系的笛卡尔乘积，令 R 为有 m 元的关系，S 为有 n 元的关系，则 R×S 是 m+n 元的元组的集合，其前 m 个元素来自 R 的一个元组，而后 n 个元素来自 S 的一个元组。形成定义如下：

$$R \times S \equiv \{t \mid t = < t_r , t_s > t_r \in R \wedge t_s \in S\}$$

若 R 有 u 个元组，S 有 v 个元组，则 R×S 有 u×v 个元组。例如，有关系 R 与关系 S 如表 3-1 和表 3-2 所示。

表 3-1　关系 R

U1	U2	U3	U4
a	b	c	d
a	b	e	f
c	a	c	d

表 3-2　关系 S

U3	U4
c	d
e	f

对 R 与 S 做笛卡尔积运算，其结果有 4+2＝6 列，元组数量有 3*2＝6 条。如表 3-3 所示。

表 3-3　关系 R×S

U1	U2	U3	U4	U3	U4
a	b	c	d	c	d
a	b	c	d	e	f
a	b	e	f	c	d
a	b	e	f	e	f
c	a	c	d	c	d
c	a	c	d	e	f

（5）投影。从一个关系中抽取指明的属性（列）。令 R 为一个包含属性 A 的关系，则

$$\pi_A(R) \equiv \{t \mid t \in [A] \mid t \in R\}$$

例如，对表 3-1 关系 R 做投影操作， p1,2(R)，则结果如表 3-4 所示。

注意：在关系代数操作中涉及的数字代表的是列号，， p1,2(R)操作是对第 1 列与第 2 列做投影。

表 3-4　关系 R 投影操作

U1	U2
a	b
c	a

（6）选择。从关系 R 中抽取出满足给定限制条件的记录，记作：

$$\sigma_F(R) \equiv \{t \mid t \in R \wedge F(t) = true\}$$

例如，对表 3-1 关系 R 做选择操作，$\sigma_{1=3}(R)$，则结果如表 3-5 所示。

表 3-5　关系 R 选择操作

U1	U2	U3	U4
c	a	c	d

其中 F 表示选择条件，是一个逻辑表达式（逻辑运算符+算术表达式）。选择运算是从元组（行）的角度进行的运算。

（7）θ 连接。θ 连接从两个关系的笛卡儿积中选取属性之间满足一定条件的元组，记作：

$$R \underset{A\theta B}{\triangleright\triangleleft} S \equiv \{t_r t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[B]\}$$

其中 A 和 B 分别为 R 和 S 上元数相等且可比的属性组。θ 为“=”的连接，称为等值连接，记作：

$$R \underset{A=B}{\triangleright\triangleleft} S \equiv \{t_r t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

如果两个关系中进行比较的分量必须是相同的属性组，并且在结果中将重复的属性去掉，则称为自然连接，记作：

$$R \triangleright\triangleleft S \equiv \{t_r t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

例如，对表 3-1 关系 R 与表 3-2 关系 S 做自然连接操作。结果集如表 3-6 所示。

表 3-6　关系 R 与关系 S 自然连接

U1	U2	U3	U4
a	b	c	d
a	b	e	f
c	a	c	d

（8）除。设有关系 R(X，Y)与关系 S(Z)，Y 和 Z 具有相同的属性个数，且对应属性出自相同域。关系 R(X，Y)÷S(Z)所得的商关系是关系 R 在属性 X 上投影的一个子集，该子

集和 S(Z)的笛卡尔积必须包含在 R(X，Y)中，记为 R÷S，其具体计算公式为：

$$R \div S = \pi_{1,2,\ldots,r-s}(R) - \pi_{1,2,\ldots,r-s}((\pi_{1,2,\ldots,r-s}(R) \times S) - R)$$

例如，对表 3-1 的关系 R 与表 3-2 的关系 S 做除法运算。

求解过程为：首先，按除运算定义要求，确定 X，Y，Z 属性集合。Y 是关系 R 中的属性集合，Z 是 S 中全部属性的集合，即 Z={U3，U4}，由于 Y=Z，因此，Y={U3，U4}， X={U1，U2}。也就是说，R÷S 结果集包含属性 U1 和 U2；然后，将关系 R 的 U1、U2（共有<a，b>、<c，a>两个元组）与关系 S 作笛卡尔积操作，结果如表 3-7 所示。

表 3-7　R(U1,U2)×S

U1	U2	U3	U4
a	b	c	d
a	b	e	f
c	a	c	d
c	*a*	*e*	*f*

通过检查表 3-7，可以发现元组<a，b>与 S(Z)的笛卡尔积被包含在 R(X，Y)中，而元组<c，a>与 S(Z)的笛卡尔积有一个元组未被包含在 R(X，Y)中，所以，结果集中只有元组<a，b>。

3.2.4 数据的规范化

关系模型满足的确定约束条件称为范式，根据满足约束条件的级别不同，范式由低到高分为 1NF（第一范式）、2NF（第二范式）、3NF（第三范式）、BCNF（BC 范式）、4NF（第四范式）等。不同的级别范式性质不同。

把一个低一级的关系模型分解为高一级关系模型的过程，称为关系模型的规范化。关系模型分解必须遵守两个准则。

（1）无损连接性：信息不失真（不增减信息）。

（2）函数依赖保持性：不破坏属性间存在的依赖关系。

规范化的基本思想是逐步消除不合适的函数依赖，使数据库中的各个关系模型达到某种程度的分离。规范化解决的主要是单个实体的质量问题，是对于问题域中原始数据展现的正规化处理。

规范化理论给出了判断关系模型优劣的理论标准，帮助预测模式可能出现的问题，是数据库逻辑设计的指南和工具，具体有：

（1）用数据依赖的概念分析和表示各数据项之间的关系。

（2）消除 E-R 图中的冗余联系。

1．函数依赖

通俗地说，就像自变量 x 确定之后，相应的函数值 f(x)也就唯一确定了一样，函数依赖是衡量和调整数据规范化的最基础的理论依据。

例如，记录职工信息的结构如下：

职工工号（EMP_NO）

职工姓名（EMP_NMAE）

所在部门（DEPT）。

则说 EMP_NO 函数决定 EMP_NMAE 和 DEPT，或者说 EMP_NMAE，DEPT 函数依赖于 EMP_NO，记为：EMP_NO→EMP_NMAE，EMP_NO→DEPT。

关系 R<U，F>中的一个属性或一组属性 K，如果给定一个 K 则唯一决定 U 中的一个元组，也就是 U 函数完全依赖于 K，就称 K 为 R 的码。一个关系可能有多个码，选中其中一个作为主码。

包含在任一码中的属性称为主属性，不包含在任何码中的属性称为非主属性。

关系 R 中的属性或属性组 X 不是 R 的码，但 X 是另一个关系模型的码，称 X 是 R 的外码。

主码和外码是一种表示关系间关联的重要手段。数据库设计中一个重要的任务就是要找到问题域中正确的关联关系，孤立的关系模型很难描述清楚业务逻辑。

2．第一范式

1NF 是最低的规范化要求。如果关系 R 中所有属性的值域都是简单域，其元素（即属性）不可再分，是属性项而不是属性组，那么关系模型 R 是第一范式的，记作 RÎ1NF。这一限制是关系的基本性质，所以任何关系都必须满足第一范式。第一范式是在实际数据库设计中必须先达到的，通常称为数据元素的结构化。

例如，表 3-8 所示的结构就不满足 1NF 的定义。

表 3-8　不满足 1NF 的结构

职工工号	职工姓名	住　址
32060231	张晓明	江苏省南通市人民路 56 号[226001]

表 3-8 为非第一范式，分解如表 3-9 所示。

表 3-9 满足 1NF 的结构

职工工号	职工姓名	所在省	所在市	详细地址	邮 编
32060231	张晓明	江苏省	南通市	人民路 56 号	226001

就满足了第一范式。经过处理后，就可以以省、市为条件进行查询和统计了。

满足 1NF 的关系模型会有许多重复值，并且增加了修改其数据时引起疏漏的可能性。为了消除这种数据冗余和避免更新数据的遗漏，需要更加规范的 2NF。

3．第二范式

如果一个关系 R 属于 1NF，且所有的非主属性都完全依赖于主属性，则称之为第二范式，记作 RÎ2NF。

为了说明问题，现举一个例子来说明：

有一个获得专业技术证书的人员情况登记表结构为：

省份、姓名、证书名称、证书编号、核准项目、发证部门、发证日期、有效期。

这个结构符合 1NF，其中“证书名称”和“证书编号”是主码，但是因为“发证部门”只完全依赖于“证书名称”，即只依赖于主关键字的一部分（即部分依赖），所以它不符合 2NF，这样首先存在数据冗余，因为证书种类可能不多。其次，在更改发证部门时，如果漏改了某一记录，存在数据不一致。再次，如果获得某种证书的职工全部跳槽了，那么这个发证部门的信息就可能丢失了，即这种关系不允许存在某种证书没有获得者的情况。

可以用分解的方法消除部分依赖的情况，而使关系达到 2NF 的标准。方法是，从现有关系中分解出新的关系表，使每个表中所有的非关键字都完全依赖于各自的主关键字。可以分解成两个表（省份、姓名、证书名称、证书编号、核准项目、发证日期、有效期）和（证书名称、发证部门），这样就完全符合 2NF 了。

4．第三范式

如果一个关系 R 属于 2NF，且每个非主属性不传递依赖于主属性，这种关系是 3NF，记作 RÎ3NF。

从 2NF 中消除传递依赖，就是 3NF。例如，有一个表（职工姓名，工资级别，工资额），其中职工姓名是关键字，此关系符合 2NF，但是因为工资级别决定工资额，也就是说非主属性“工资额”传递依赖于主属性“职工姓名”，它不符合 3NF，同样可以使用投影分解的办法分解成两个表：（职工姓名，工资级别），（工资级别，工资额）。

5．BC 范式

一般满足 3NF 的关系模型已能消除冗余和各种异常现象，获得比较满意的效果，但无论 2NF 还是 3NF 都没有涉及主属性间的函数依赖，所以有时仍会引起一些问题。由此引入 BC 范式（由 Boyeet 和 Codd 提出）。通常认为 BCNF 是第三范式的改进。

BC 范式的定义：如果关系模型 R∈1NF，且 R 中每一个函数依赖关系中的决定因素都包含码，则 R 是满足 BC 范式的关系，记作 RÎBCNF。

当一个关系模型 R BCNF，则在函数依赖范畴里，就认为已彻底实现了分离，消除了插入、删除的异常。

综合 1NF、2NF 和 3NF、BCNF 的内涵可概括如下：

（1）非主属性完全函数依赖于码（2NF 的要求）；

（2）非主属性不传递依赖于任何一个候选码（3NF 的要求）；

（3）主属性对不含它的码完全函数依赖（BCNF 的要求）；

（4）没有属性完全函数依赖于一组非主属性（BCNF 的要求）。

3.2.5 反规范化

数据库中的数据规范化的优点是减少了数据冗余，节约了存储空间，相应逻辑和物理的 I/O 次数减少，同时加快了增、删、改的速度，但是对完全规范的数据库查询，通常需要更多的连接操作，从而影响查询速度。因此，有时为了提高某些查询或应用的性能而破坏规范规则，即反规范化（非规范化处理）。

常见的反规范化技术包括：

（1）增加冗余列

增加冗余列是指在多个表中具有相同的列，它常用来在查询时避免连接操作。例如：以规范化设计的理念，学生成绩表中不需要字段“姓名”，因为“姓名”字段可以通过学号查询到，但在反规范化设计中，会将“姓名”字段加入表中。这样查询一个学生的成绩时，不需要与学生表进行连接操作，便可得到对应的“姓名”。

（2）增加派生列

增加派生列指增加的列可以通过表中其他数据计算生成。它的作用是在查询时减少计算量，从而加快查询速度。例如：订单表中，有商品号、商品单价、采购数量，我们需要订单总价时，可以通过计算得到总价，所以规范化设计的理念是无须在订单表中设计“订单总价”字段。但反规范化则不这样考虑，由于订单总价在每次查询都需要计算，这样会占用系

统大量资源，所以在此表中增加派生列“订单总价”以提高查询效率。

（3）重新组表

重新组表指如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。

（4）分割表

有时对表做分割可以提高性能。表分割有两种方式。

水平分割：根据一列或多列数据的值把数据行放到两个独立的表中。水平分割通常在下面的情况下使用。

情况 1：表很大，分割后可以降低在查询时需要读的数据和索引的页数，同时也降低了索引的层数，提高查询效率。

情况 2：表中的数据本来就有独立性，例如表中分别记录各个地区的数据或不同时期的数据，特别是有些数据常用，而另外一些数据不常用。

情况 3：需要把数据存放到多个介质上。

垂直分割：把主码和一些列放到一个表，然后把主码和另外的列放到另一个表中。如果一个表中某些列常用，而另外一些列不常用，则可以采用垂直分割，另外垂直分割可以使得数据行变小，一个数据页就能存放更多的数据，在查询时就会减少 I/O 次数。其缺点是需要管理冗余列，查询所有数据需要连接操作。

3.3 数据库设计

数据库设计的过程是将数据库系统与现实世界密切地、有机地、协调一致地结合起来的过程。数据库的设计质量与设计者的知识、经验和水平密切相关。作为数据库应用系统的重要组成部分，数据库设计的成败往往直接关系到整个应用系统的成败。

以数据库为基础的数据库应用系统与其他计算机应用系统相比往往具有数据量庞大、数据保存时间长、数据关联复杂、用户要求多样化等特点。

数据库设计中面临的主要困难和问题有：

（1）同时具备数据库知识与应用业务知识的人很少。懂得计算机与数据库的人一般都缺乏应用业务知识和实际经验，而熟悉应用业务的人又往往不懂计算机和数据库。

（2）项目初期往往不能确定应用业务的数据库系统的目标。

（3）缺乏完善的设计工具和设计方法。

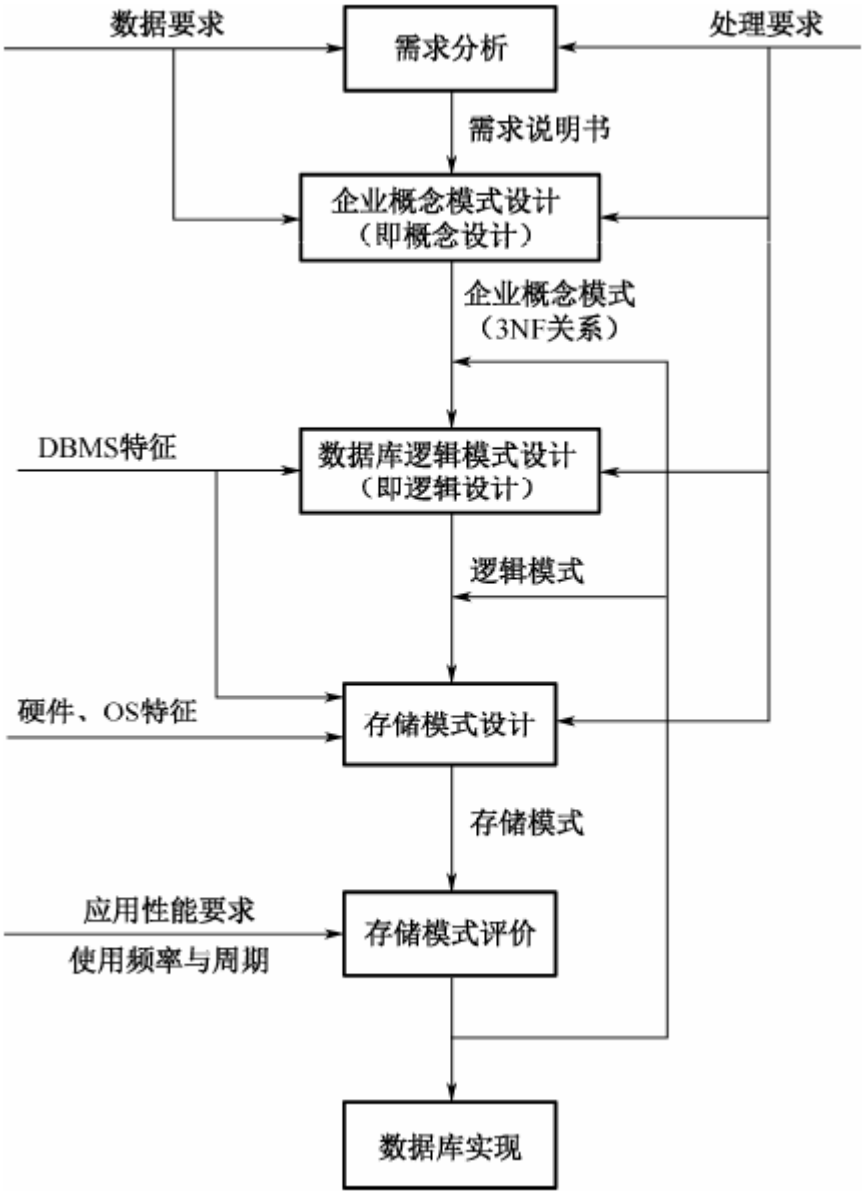
（4）需求的不确定性。用户总是在系统的开发过程中不断提出新的要求，甚至在数据库建立之后还会要求修改数据库结构或增加新的应用。

（5）应用业务系统千差万别，很难找到一种适合所有业务的工具和方法，这就增加了研究数据库自动生成工具的难度。因此，研制适合一切应用业务的全自动数据库生成工具是不可能的。

3.3.1 数据库设计的方法

目前已有的数据库设计方法可分为四类，即直观设计法、规范设计法、计算机辅助设计法和自动化设计法。直观设计法又称单步逻辑设计法，它依赖于设计者的知识、经验和技巧，缺乏工程规范的支持和科学根据，设计质量也不稳定，因此越来越不适应信息管理系统发展的需要。为了改变这种状况，1978 年 10 月来自 30 多个欧美国家的主要数据库专家在美国新奥尔良市专门讨论了数据库设计问题，提出了数据库设计规范，把数据库设计分为需求分析、概念结构设计、逻辑结构设计和物理结构设计 4 个阶段。目前，常用的规范设计方法大多起源于新奥尔良方法，如基于 3NF 的设计方法、LRA 方法、面向对象的数据库设计方法及基于视图概念的数据库设计方法等。架构设计师考试中，主要了解基于 3NF 的数据库设计方法即可。

基于 3NF 的数据库设计方法是由 S.Atre 提出的数据库设计的结构化设计方法，其基本思想是在需求分析的基础上，识别并确认数据库模式中的全部属性和属性间的依赖，将它们组织成一个单一的关系模型，然后再分析模式中不符合 3NF 的约束条件，用投影和连接的办法将其分解，使其达到 3NF 条件。其具体设计步骤分为 5 个阶段，如图 3-2 所示。



3-2 基于 3NF 的数据库设计方法

（1）设计企业模式。利用上述得到的 3NF 关系模型画出企业模式。具体包括：

分析应用环境，并设定环境中所使用的各种资料。

确定每一种报表各自所包含的数据元素。

确定数据元素之间的关系，如确定主关键字和一般的数据元素。

对每一组或若干组数据元素推导出 3NF 的关系模型。

在 3NF 关系模型的基础上画出数据库的企业模式。

（2）设计数据库逻辑模式。根据上一步得到的企业模式选定数据模型，从而得出适用 于某个 DBMS 的逻辑模式。根据逻辑模式导出各种报表与事务处理所使用的外模式。

（3）设计数据库物理模式（存储模式）。根据数据库的逻辑模式和给定的计算机系统 设

计物理模式。

（4）评价物理模式。对物理模式估算空间利用情况，并推算输入输出的概率。必要时 根据物理模式调整各种报表与事务处理的外模式。对外模式进行存取时间的估算。

（5）数据库实现。具体实现数据库。

3.3.2 数据库设计的基本步骤

分步设计法遵循自顶向下、逐步求精的原则，将数据库设计过程分解为若干相互独立又相互依存的阶段，每一阶段采用不同的技术与工具，解决不同的问题，从而将问题局部化，减少了局部问题对整体设计的影响。目前，此方法已在数据库设计中得到了广泛应用并获得了较好的效果。

在分步设计法中，通常将数据库的设计分为需求分析、概念结构设计、逻辑结构设计和数据库物理设计 4 个阶段，如图 3-3 所示。

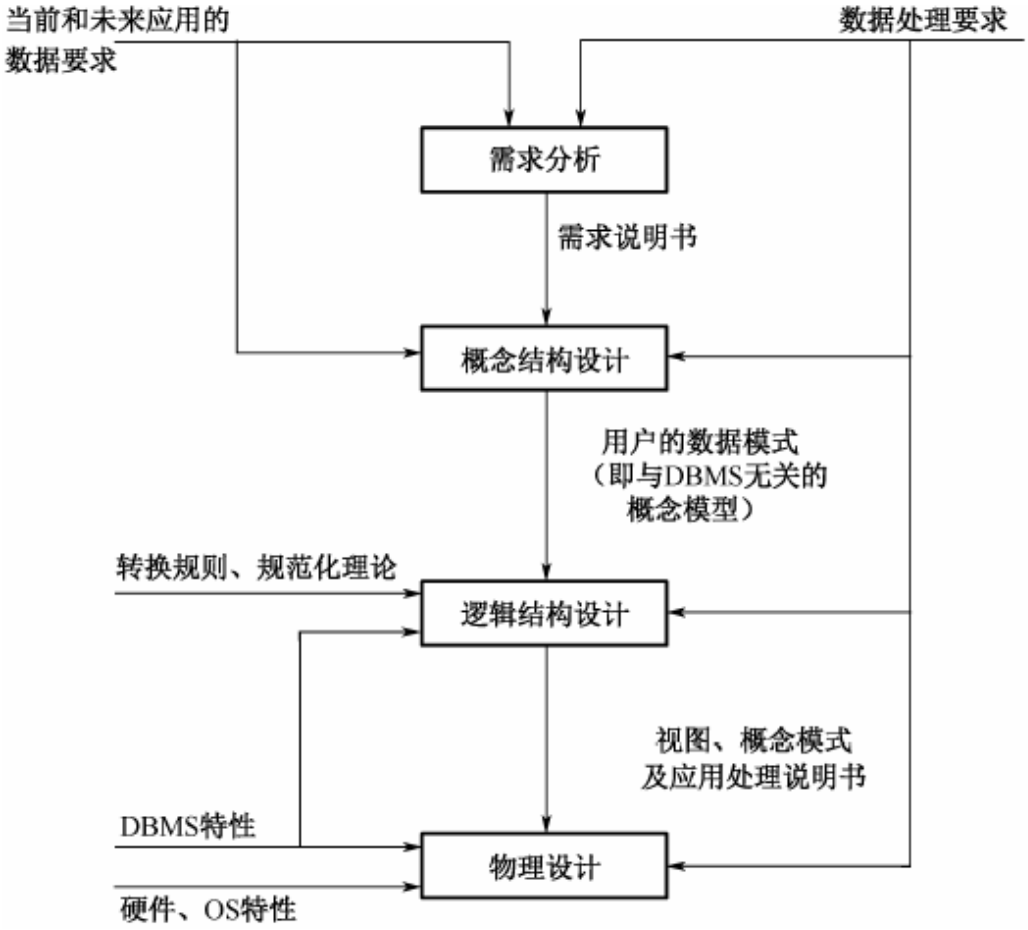


图 3-3 数据库的设计步骤

1．需求分析

需求分析是指收集和分析用户对系统的信息需求和处理需求，得到设计系统所必需的需求信息，建立系统说明文档。其目标是通过调查研究，了解用户的数据要求和处理要求，并按一定格式整理形成需求说明书。需求说明书是需求分析阶段的成果，也是今后设计的依据，它包括数据库所涉及的数据、数据的特征、使用频率和数据量的估计，如数据名、属性及其类型、主关键字属性、保密要求、完整性约束条件、更改要求、使用频率、数据量估计等。这些关于数据的数据称为元数据。在设计大型数据库时，这些数据通常由数据字典来管理。用数据字典管理元数据有利于避免数据的重复或重名，以保持数据的一致性及提供各种统计数据，因而有利于提高数据库设计的质量，同时可以减轻设计者的负担。

2．概念结构设计

它是数据库设计的第二阶段，其目标是对需求说明书提供的所有数据和处理要求进行抽象与综合处理，按一定的方法构造反映用户环境的数据及其相互联系的概念模型，即用户的数据模型或企业数据模型。这种概念数据模型与 DBMS 无关，是面向现实世界的、极易为用户所理解的数据模型。为保证所设计的概念数据模型能正确、完整地反映用户的数据及其相互关系，便于进行所要求的各种处理，在本阶段设计中可吸收用户参与和评议设计。在进行概念结构设计时，可先设计各个应用的视图（view），即各个应用所看到的数据及其结构，然后再进行视图集成，以形成一个单一的概念数据模型。这样形成的初步数据模型还要经过数据库设计者和用户的审查与修改，最后形成所需的概念数据模型。

3．逻辑结构设计

这一阶段的设计目标是把上一阶段得到的与 DBMS 无关的概念数据模型转换成等价的，并为某个特定的 DBMS 所接受的逻辑模型所表示的概念模式，同时将概念设计阶段得到的应用视图转换成外部模式，即特定 DBMS 下的应用视图。在转换过程中要进一步落实需求说明，并满足 DBMS 的各种限制。该阶段的结果是用 DBMS 所提供的数据定义语言（DDL）写成的数据模式。逻辑设计的具体方法与 DBMS 的逻辑数据模型有关。逻辑模型应满足数据库存取、一致性及运行等各方面的用户需求。

4．数据库物理设计

物理设计阶段的任务是把逻辑设计阶段得到的满足用户需求的已确定的逻辑模型在物理上加以实现，其主要内容是根据 DBMS 提供的各种手段，设计数据的存储形式和存取路径，如文件结构、索引设计等，即设计数据库的内模式或存储模式。数据库的内模式对数据库的性能影响很大，应根据处理需求及 DBMS、操作系统和硬件的性能进行精心设计。

实际上，数据库设计的基本过程与任何复杂系统开发一样，在每一阶段设计基本完成后，

都要进行认真的检查，看是否满足应用需求，是否符合前面已执行步骤的要求和满足后续步骤的需要，并分析设计结果的合理性。在每一步设计中，都可能发现前面步骤的遗漏或处理不当之处，此时，往往需要返回去重新处理并修改设计和有关文档。所以，数据库设计过程通常是一个反复修改、反复设计的迭代过程。

3.3.3 需求分析

需求分析是数据库设计过程的第一步，是整个数据库设计的依据和基础。需求分析做得不好，会导致整个数据库设计重新返工。需求分析的目标是通过对单位的信息需求及处理要求的调查分析得到设计数据库所必需的数据集及其相互联系，形成需求说明书，作为后面各设计阶段的基础。因此，这一阶段的任务是：

1．确认需求、确定设计目标

数据库设计的第一项工作就是要对系统的整个应用情况进行全面、详细的实地调查，弄清现行系统的组织结构、功能划分、总体工作流程，收集支持系统总的设计目标的基础数据和对这些数据的处理要求，明确用户总的需求目标；通过分析，确定相应的设计目标，即确定数据库应支持的应用功能和应用范围，明确哪些功能由计算机完成或准备让计算机完成，哪些环节由人工完成，以确定应用系统实现的功能。这一阶段收集到的基础数据和一组数据流程图是下一步进行概念设计的基础。

2．分析和收集数据

这是整个需求分析的核心任务。它包括分析和收集用户的信息需求、处理需求、完整性需求、安全性需求，以及对数据库设计过程有用的其他信息。

信息需求是指在设计目标范围内涉及的所有实体、实体的属性及实体间的联系等数据对象，包括用户在数据处理中的输入/输出数据及这些数据间的联系。在收集中，要收集数据的名称、类型、长度、数据量、对数据的约束及数据间联系的类型等信息。

处理需求是指为了获得所需的信息而对数据加工处理的要求。它主要包括，处理方式是实时还是批处理，各种处理发生的频度、响应时间、优先级别及安全保密要求等。所要收集的其他信息还有企业在管理方式、经营方式等方面可能发生的变化等。

分析和收集数据的过程是数据库设计者对各类管理活动进行深入调查研究的过程，调查对象包括数据管理部门的负责人、各使用部门的负责人及操作员等各类管理人员，通过与各类管理人员相互交流，逐步取得对需求的一致认识。

3．整理文档

分析和收集得到的数据必须经过筛选整理，并按一定格式和顺序记载保存，经过审核成为正式的需求说明文档，即需求说明书。实际上，需求说明书是在需求分析的过程中逐渐整理形成的，是随着这一过程的不断深入而反复修改与完善的对系统需求分析的全面描述。由用户、领导和专家共同评审，是以后各设计阶段的主要依据。这一步的工作是进行全面的汇总与整理，使之系统化，以形成标准化的统一形式。

3.3.4 概念结构设计

概念结构设计阶段所涉及的信息不依赖于任何实际实现时的环境，即计算机的硬件和软件系统。概念结构设计的目标是产生一个用户易于理解的，反映系统信息需求的整体数据库概念结构。概念结构设计的任务是，在需求分析中产生的需求说明书的基础上按照一定的方法抽象成满足应用需求的用户的信息结构，即通常所称的概念模型。概念结构的设计过程就是正确选择设计策略、设计方法和概念数据模型并加以实施的过程。

概念数据模型的作用是：提供能够识别和理解系统要求的框架；为数据库提供一个说明性结构，作为设计数据库逻辑结构，即逻辑模型的基础。

概念模型的描述工具应该能够体现概念模型的特点，如 E-R 模型。近年来，由于面向对象数据模型具有更丰富的语义、更强的描述能力而越来越受到人们的重视，不但出现了商品化的面向对象 DBMS，而且开始实际应用于概念模型的设计中，作为数据库概念设计的工具。Teory 等人提出的扩展的 E-R 模型增加了类似面向对象数据模型中的普遍化和聚合等语义描述机制，为这种最为人们熟悉的数据模型注入了新的生机，为概念模型的描述增加了一种理想的选择。

概念结构的设计策略主要有自底向上、自顶向下、由里向外和混合策略。在具体实现设计目标时有两种极端的策略或方案，一是建立一个覆盖整个单位所有功能域的全局数据库，称之为全局方案或全局策略；另一种则是对每一个应用都建立一个单独的数据库，称之为应用方案或应用策略。

由于各个部门对于数据的需求和处理方法各不相同，对同一类数据的观点也可能不一样，它们有自己的视图，所以可以首先根据需求分析阶段产生的各个部门的数据流图和数据字典中的相关数据设计出各自的局部视图，然后进行视图集成。

1．视图设计

在实体分析法中，局部视图设计的第一步是确定其所属的范围，即它所对应的用户组，然后对每个用户组建立一个仅由实体、联系及它们的标识码组成的局部信息结构（局部数据模式）框架，最后再加入有关的描述信息，形成完整的局部视图（局部数据模式）。这样做的目的是为了集中精力处理好用户数据需求的主要方面，避免因无关紧要的描述细节而影响局部信息结构的正确性。整个过程可分为 4 个步骤：确定局部视图的范围；识别实体及其标识；确定实体间的联系；分配实体及联系的属性。

（1）确定局部视图的范围。需求说明书中标明的用户视图范围可以作为确定局部视图范围的基本依据，但它通常与子模式范围相对应，有时因为过大而不利于局部信息结构的构造，故可根据情况修改，但也不宜分得过小，过小会造成局部视图的数量太大及大量的数据冗余和不一致性，给以后的视图集成带来很大的困难。

局部视图范围确定的基本原则是：

各个局部视图支持的功能域之间的联系应最少。

实体个数适量。一个局部视图所包含的实体数量反映了该局部视图的复杂性，按照信息论中“7 2”的观点，人们在同一时刻可同时顾及的事情一般在 5～9 之间，其中以 6 或 7 最为适当。因此，一个局部视图内的实体数不宜超过 9 个，否则就会过于复杂，不便于人们理解和管理。

（2）识别实体及其标识。在需求分析中，人们已经初步地识别了各类实体、实体间的联系及描述其性质的数据元素，这些统称为数据对象，它们是进一步设计的基本素材。这一步的任务就是在确定的局部视图范围内，识别哪些数据对象作为局部视图的基本实体及其标识，并定义有关数据对象在 E-R 模型中的地位。

（3）确定实体间的联系。实际上，识别联系的主要任务是在需求分析阶段完成的。这里的工作一是从局部视图的角度进行一次审核，检查有无遗漏之处，二是确切地定义每一种联系。

在现实世界中，诸多形式的联系大致可分为三类：存在性联系、功能性联系和事件联系。存在性联系如学校有教师、教室有学生、工厂有产品、产品有顾客等；功能性联系如教师讲授课程、教师参与科研、仓库管理员管理仓库等；事件联系如学生借书、产品发运等。

根据上述分类仔细检查在给定的局部视图范围内是否有未识别的联系，在确认所有的联系都已识别并无遗漏之后，还需对联系进行正确的定义。定义联系就是对联系语义的仔细分析，识别联系的类型，确定实体在联系中的参与度。

① 二元联系的类型与定义。二元联系是指两个实体类之间的联系。根据参与联系的两

个实体类值之间的对应关系分为一对一、一对多及多对多三种类型。

一对一联系：这是一种最简单的联系类型。若对于实体集 A 中的每一个实体，实体集 B 中至多有一个实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1:1。例如在一个施工单位中，如果规定每项工程最多只能由一名工程师负责管理，而一名工程师最多也只能负责一项工程，则工程师与工程间的这种管理联系便是一对一联系。

一对多联系：若对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n≥0）与之联系；反之，对于实体集 B 中的每一个实体，实体集 A 中至多有一个实体与之联系，则称实体集 A 与实体集 B 有一对多的联系，记为 1:n。以专业与学生间的关系为例：如规定一个专业可以管理许多学生，每个学生只能属于一个专业，这种联系就是一对多联系。

多对多联系：若对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n≥0）与之联系，反过来，对实体集 B 中每一个实体，实体集 A 中也有 m 个实体（m≥0）与之联系，则称实体集 A 与实体集 B 具有多对多联系，记为 m:n。教师与学生这两个实体类间的教与学的联系就是多对多的联系。这时，只有<教师、学生>对才能确定一个特定的教学联系。因此，一般情况下可以以两个关联实体的标识拼凑作为联系的标识。但这种方法对某些情况就不能构成有效的联系标识。当一个实体值在同一个联系上可能存在多个不同的联系值时，就会出现这种情况。如教师与其讲授的课程之间的联系，同一个教师可讲授几门不同的课程，也可以多次讲授同一门课程，这是一种特殊的多对多联系。显然，对于教师与讲授课程间的联系，如在教师档案中要求记录担任教学工作的情况，就需要在联系标识中增加表示授课日期的属性，即其合适的联系标识可能为（教师号，课程号，授课日期）。

实体类内部的联系：这种联系发生在同一类实体的不同实体之间，因此称为内部联系或自联系，它也是一种二元联系，其表示方式与前面的二元联系并无不同，要注意仔细区别同一实体类中的不同实体在联系中扮演的不同角色及联系标识的选择。例如在职工类实体中间就存在着管理者与被管理者的联系。一个职工可以管理别的职工，称为管理者或领导者。一个管理者可以管理多个职工，而一个职工最多只从属于一位管理者，从而构成了一对多联系。若规定所有职工都要受管理（最高管理者考虑自己管理自己），但不是所有职工都是管理者，则此联系在“多”端呈现强制性。其中每个联系实体包含两个职工号值：职工号和管理员职工号，以区别不同的实体在联系中的角色。

若略去实体与其属性图，以上实体间的二元联系可用图 3-4 表示。

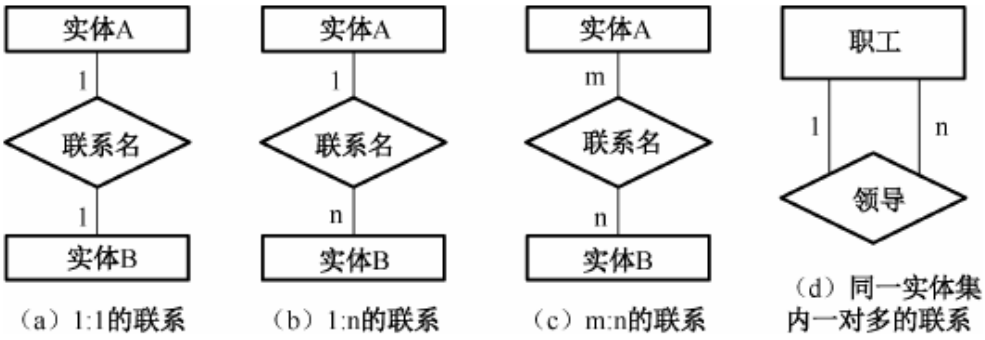


图 3-4　实体间的二元联系

② 多元联系的识别与定义。两个以上的实体类之间的联系称为多元联系。例如在供应商向工程供应零件这类事件中，如果任一供应商可向任一工程供应任一种零件，则为了确定哪个供应商向哪个工程供应了何种零件，就必须定义一个三元联系，因为只有供应商、工程及零件三者一起才能唯一地确定一个联系值。其联系的标识由参与联系的实体类的标识拼接而成，在此例中由供应商、工程、零件三个实体类的标识拼接而成。

2．视图集成

视图集成就是要将反映各用户组数据的局部数据模式综合成单位中某个确定范围内的单一数据视图，即全局数据模式，又称模式汇总。该全局数据模式是未来数据库结构的基础，因此视图集成是数据库设计过程中一个十分重要的步骤，也是一项较为复杂和困难的任务。当所有局部视图设计完毕，就可开始视图集成。集成过程中会发生一些冲突，冲突的表现和处理策略如下：

① 同名异义。为了发现不同视图间的同名异义问题，可以列出所有同名数据对象，然后逐一判别其语义。对同名异义冲突通常采用换名加以解决，既可对同名者之一换名，也可对两者都给以重新命名。识别语义的主要方法是进行值域分析。

② 异名同义。识别异名同义比较困难，一般由设计者对所有对象一个不漏地逐一鉴别。它同样采用换名的方法解决。若归并时试图将它们合并为一个对象，则可以把其中之一的名称作为合并后的对象名；若集成后，它们仍以两个不同的对象存在，则可对其一换名。当然，若原名都不合适，则可以对两者都重新命名。

③ 同名不同层次。如果两个对象同名，但其中之一是作为一个视图中的实体，而另一个是另一视图中的属性，则在集成时就会发生同名不同层次的冲突。解决这种冲突的办法有两个，一是将属性转换为实体，二是将实体变换成属性。

例如，设一局部视图中有一部门实体 DEPT，而在另一与之集成的视图中有职工实体

EMP，且 EMP 有属性 DEPT，于是发生了同名不同层次的冲突。此时，可将 EMP 的 DEPT 属性去掉，另设一个实体 DEPT 与 EMP 建立联系，这时再与另一视图集成就容易多了。

再如，设一局部视图中有一名为 STOR 的仓库实体，其中含有一属性部门号（DEPT-NO）；在另一局部视图中有一单位实体 DEPT，其中仅含有一个属性 DEPT-NO。对这类同名不同层次的冲突，可将 DEPT 实体变换为其所在视图中与 DEPT 相关的另一实体的属性，然后再进行集成。

希赛教育专家提示：实体变换为属性时通常要满足一些特定条件，例如，该实体通常只含有一个与同名属性具有共同特征的属性，且一定存在一个与该实体存在联系的另外的实体。

④ 虽同名同义，但对象联系测度不同。所谓联系测度是指实体的联系是一对一、一对多还是多对多。若同名同义对象在一个局部视图中为一对多联系，在另一局部视图中为多对多联系，则在集成时将发生联系测度冲突。一般而言，一对多包含一对一，多对多包含一对多。所以解决这种冲突的方法往往取较高测度为集成后的相应联系的测度。

3.3.5 逻辑结构设计

数据库逻辑结构设计的任务就是把概念结构设计阶段设计好的基本 E-R 图转换为与具体机器上的 DBMS 产品所支持的数据模型相符合的逻辑结构。这一阶段是数据库结构设计的重要阶段。

数据库逻辑设计的基础是概念设计的结果，而其成果应包括某 DBMS 所支持的外模式、概念模式及其说明及建立外模式和概念模式的 DDL 程序。因此，进行逻辑设计前，必须了解数据库设计的需求说明和概念设计的成果（包括 E-R 图和其他文档），并仔细阅读有关 DBMS 的文件。数据库的外模式和概念模式是用户所看到的数据库，是应用程序访问数据库的接口，因此在数据库逻辑设计阶段，还必须提供应用程序编制的有关说明，最好试编一些典型的访问数据库的应用程序，以检验所设计的概念模式是否满足使用要求。概念模式是数据库的基础，它的设计质量对数据库的使用和性能，以及数据库在今后发展过程中的稳定性均有直接的影响。为了设计出能够正确反映一个项目数据间内在联系的好的概念模式，设计时必须正确处理各种应用程序之间、数据库性能与数据模式的合理性和稳定性之间的各种矛盾，对设计中出现的各种矛盾要权衡利弊，分清主次，按统筹兼顾的原则加以正确处理。

逻辑结构设计一般分为以下几个步骤：

（1）将概念结构向一般关系模型转化。

（2）将第一步得到的结构向特定的 DBMS 支持下的数据模型转换。

（3）依据应用的需求和具体的 DBMS 的特征进行调整与完善。

下面以常用的 E-R 模型和扩充 E-R 模型为主，针对关系数据库的逻辑设计介绍基本原则和方法。

1.基本 E-R 模型向关系模型的转换

基本 E-R 模型主要包含实体和联系两个抽象概念，实体和联系本身还可能附有若干属性。其转换的基本原则是，实体和联系分别转换成关系，属性则转换成相应关系的属性。因此，E-R 模型向关系模型的转换比较直观，但不同元数的联系具体转换方法稍有不同，下面根据不同的情况分别讨论。

（1）一对一联系。设有两个实体 E1 和 E2 之间为一对一联系。此情况存在三种可能的转换方案。

方案 1：将实体 E1、E2 和联系名 R 分别转换成为关系 E1、E2 和 R，它们的属性分别转为相应关系的属性，即得到：

El（k1，a）

E2（k2，b）

R（k1，k2，r）（k2 是候选关键字）

其中属性下面带一横线者为关系的关键字。

方案 2：将实体 El 转换为关系 El ，将实体 E2 与联系名 R 一起转换成关系 E2 ，E2 的属性由 E2 和 R 的属性加上 E1 的关键字组成，其关键字 k1、k2 为其候选关键字。转换后的关系为：

E1（kl，a）

E2′（k2，b，k1，r），（k1 是候选关键字）

方案 3：与方案 2 类似，不过把实体 E1 与联系 R 一起转换成关系 E1 后，其结果为：

E1′（kl，a，k2，r），（k2 是候选关键字）

E2（k2，b）

上述三个方案实际上可归结为转换成三个关系和转换成两个关系两种。如果每个实体的属性数较少，而联系的属性与两个实体之一关系又较密切，则可采用方案 2 或方案 3，其优点是可减少关系数，有利于减少连接运算从而提高查询效率，但如果每个实体的属性较多，且合并后，会造成较大数据冗余和操作异常，则以采用方案 1 为宜。

（2）一对多联系。这种情况存在两种转换方案，其一是把两个实体类和一个联系类分

别转换成对应的关系，实体类的属性转换为对应关系的属性，其标识属性即为对应关系的关键字，而联系类转换得到的关系，其属性由两个实体的标识属性和联系类本身的属性组成，并以多端实体类的标识属性为其关键字。其转换结果为三个关系。第二个方案是转换成两个关系，设少端和多端的两个实体类分别为 EI、E2，联系名 R。转换时，将实体类 EI 转换为一个关系 EI，E2 和 R 合起来转换成一个关系 E2′，E2′的属性由 E2 和 R 的属性加上 E1 的标识属性组成，并以 E 2 的标识属性为其关键字。

（3）多对多联系。由两个实体类之间多对多联系组成的 E-R 模型向关系模型转换时，将两个实体类和一个联系类分别转换成关系，实体类的属性分别转换成对应关系的属性，其标识属性为其关键字，由联系类转换得到的关系的属性由两个实体类的标识属性和联系类本身的属性组成，其关键字是由两个联系的实体类的标识属性组成。

（4）多元联系。实体类分别转换为相应的关系，三个实体类间的多元联系转换为以该联系名为关系名的关系，关系的属性由各实体的标识属性及其联系的属性组成，并以各实体的标识属性为其关键字。例如图 3-5 所示的部件（PART）、工程（PROJECT）和供应者（SUPPLIER）三者之间的联系为 PJS，其属性为 QTY。转换时，把 PART、PROJECT、 SUPPLIER 和联系 PJS 分别转换为相应的关系，其结果如下：

PART（P#，PN）

PROJECT（PNO，PNAME）

SUPPLIER（S#，SN）

PJS（P#，PNO，S#，QTY）

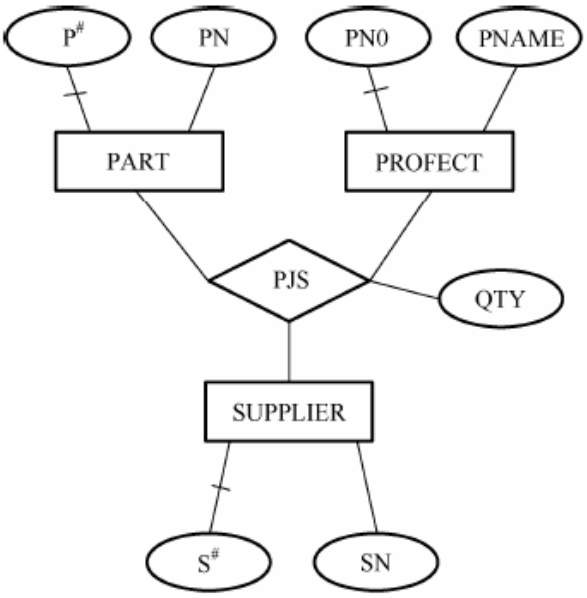


图 3-5　部件工程和供应者之间的关系

（5）自联系。自联系是同一实体集的实体间的联系。例如对于职工实体类内部有领导与被领导的联系，在部件这个实体集的实体之间有组成成分与组成者之间的联系等，均属于实体类的自联系。在这种联系中，参与联系的实体虽然来自同一实体类，但所起的作用不一样。

（6）弱实体类的转换。一个实体类，如果它的存在依赖于另一实体类，则称之为弱实体类。例如职工的亲属（DEPENDENTS）是依赖于职工（EMPLOYEE）实体类而存在的，因为实体集亲属（DEPENDENTS）是弱实体类，它们之间的关系如图 3-6 所示。由于弱实体类不能独立地存在，而是由其他实体标识而存在，所以不能单独地转换成一个关系，因此图 3-6 可转换成如下两个关系：

EMPLOYEE（empno，name，birthday）

DEPENDENTS（empno，name，sex，age，kinship）

其中 kinship 表示职工亲属与职工的关系，可以取值为“配偶”、“儿子”、“女儿”等。

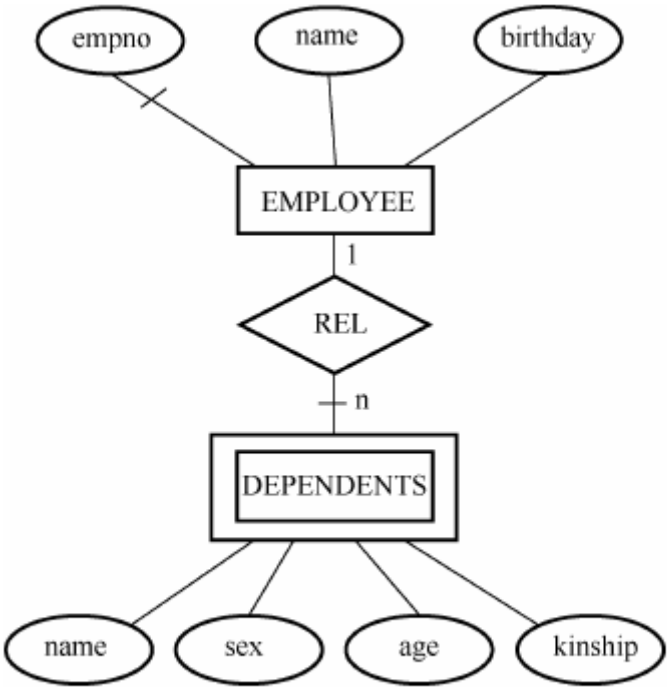


图 3-6　职工与职工的亲属间的关系

2．数据模型的优化

由 E-R 图表示的概念模型转换得到的关系模型经过规范化以后，基本上可以反映一个企业数据的内在联系，但不一定能满足应用的全部需要和系统要求，因此，还必须根据需求分析对模型做进一步的改善和调整，其内容主要是改善数据库的性能和节省存储空间两个方面。

（1）改善数据库性能的考虑。查询速度是关系数据库应用中影响性能的关键问题，必须在数据库的逻辑设计和物理设计中认真加以考虑，特别是那些对响应时间要求较苛刻的应用，应予以特别注意。

就数据库的逻辑设计而论，可从下列几个方面提高查询的速度。

① 减少连接运算。连接运算对关系数据库的查询速度有着重要的影响，连接的关系越多，参与连接的关系越大，开销也越大，因而查询速度也越慢。对于一些常用的、性能要求较高的数据库查询，最好是一元查询，但这与规范化的要求相矛盾。有时为了保证性能，往往不得不牺牲规范化要求，把规范化的关系再合并起来，称之为逆规范化。当然，这样做会引起更新异常。总之，逆规范化有得有失，设计者可根据实际情况进行权衡。

② 减小关系大小及数据量。被查询的关系的大小对查询速度影响较大。为了提高查询速度，可以采用水平分割或垂直分割等方法把一个关系分成几个关系，使每个关系的数据量

减少。例如，对于大学中有关学生的数据，既可以把全校学生的数据集中在一个关系中，也可以用水平分割的方法，分系建立关系，从而减少了每个关系的元组数。前者对全校范围内的查询较方便，后者则可以显著提高对指定系的查询速度。也可采用垂直分割的方法，把常用数据与非常用数据分开，以提高常用数据的查询速度。例如，高校中教职工档案，属性很多，有些需经常查询，有些则很少查询，如果放在一起，则关系的数据量就会很大，影响查询速度，因此把常用属性和非常用属性分开，就可提高对常用属性的查询速度。

③ 尽量使用快照。快照是某个用户所关心的那部分数据，与视图一样是一种导出关系，但它与视图有两点不同：一是视图是虚关系，数据库中并不存储作为视图的导出关系，仅仅保留它的定义，快照则是一个由系统事先生成后保留在数据库中的实关系；二是视图随数据当前值的变化而变化，快照则不随原来关系中数据的改变而及时改变，它只反映数据库中某一时刻的状态，不反映数据库的当前状态，犹如照片只反映某一时刻的情景，不能反映情景变化一样，之所以称它为快照，原因就在于此。但它与照片又有不同，快照不是一成不变的，它可以由系统周期性地刷新，或由用户用命令刷新。刷新时用当前值更新旧值。在实际应用中，快照可满足相当一部分应用的需要，甚至有些应用就是需要快照，而不是当前值。例如注明列出“某年某月某日截止”的统计或报表就是快照。由于快照是事先生成并存储在数据库中的，因而可大大缩短响应时间。目前不少 DBMS，如 Oracle、 MS SQL Server 等支持快照。对不支持快照的 DBMS，用户也可以把需要作为实关系使用的导出关系作为一个独立关系存于数据库中，但这种做法只能供查询使用，对它们的刷新及管理由用户负责。

（2）节省存储空间的一些考虑。尽管随着硬件技术的发展，提供给用户使用的存储空间越来越大，但毕竟是有限度的。而数据库，尤其是复杂应用的大型数据库，需要占用较大的外存空间。因此，节省存储空间仍是数据库设计中应该考虑的问题，不但要在数据库的物理设计中考虑，而且还应在逻辑设计中加以考虑。在数据库逻辑设计中可采取以下措施：

① 缩小每个属性占用的空间。减少每个属性占用的空间，是节省存储空间的一个有效的措施。通常可以有两种方法：即用编码和用缩写符号表示属性，但这两种方法的缺点是失去了属性值含义的直观性。

② 采用假属性。采用假属性可以减少重复数据占用的存储空间。设某关系模型 R 的属性 A 和 B 之间存在函数依赖 A→B，B 的每一个值需要占用较大的空间，但 B 的域中不同的值却比较少，A 的域中具有较多的不同值，则 B 的同一值可能在多个元组中重复出现，从而需要占用较多的空间。为了节省空间，可利用属性 B 的域中不同值少的特点，对 B 的值进行分类，用 B′ 表示 B 的类型，则 A→B 可分解成两个函数依赖，即：

A→B′，B′→B

这样，就可用 B′代替原来元组较多的关系 R 中的属性 B，而另外建立一个较小的关系 R′来描述 B′与 B 的对应关系。这里 B′在原关系 R 中起了属性 B 的替身的作用，所以称 B′ 为假属性。例如，在职工关系中，职工的经济状况这一属性通常由职工号决定，一个大型企业的职工人数很多，如每一职工逐一填写，就要占用较多的空间，为了节省空间可把经济状况分为几种类型，在元组较多的职工关系中用经济状况的类型代替原来的经济状况，这里经济状况的类型就是假属性，另外建立一个较小的关系来描述每种经济状况类型的具体内容。

希赛教育专家提示：数据库设计与数学问题求解不同，它是一项综合性工作，受到各种各样的要求和因素的制约，有些要求往往又是彼此矛盾的，因此，设计结果很难说是最佳的，常常有得有失，设计者必须根据实际情况，综合运用上述原则和有关理论，在基本合理的总体设计的基础上，做一些仔细的调整，力求最大限度地满足用户各种各样的要求。

3.3.6 物理结构设计

数据库在实际的物理设备上的存储结构和存取方法称为数据库的物理结构。数据库物理设计是利用已确定的逻辑结构及 DBMS 提供的方法、技术，以较优的存储结构、数据存取路径、合理的数据存储位置及存储分配，设计出一个高效的、可实现的物理数据库结构。显然，数据库的物理设计是完全依赖于给定的硬件环境和数据库产品的。数据库物理设计过程如图 3-7 所示。

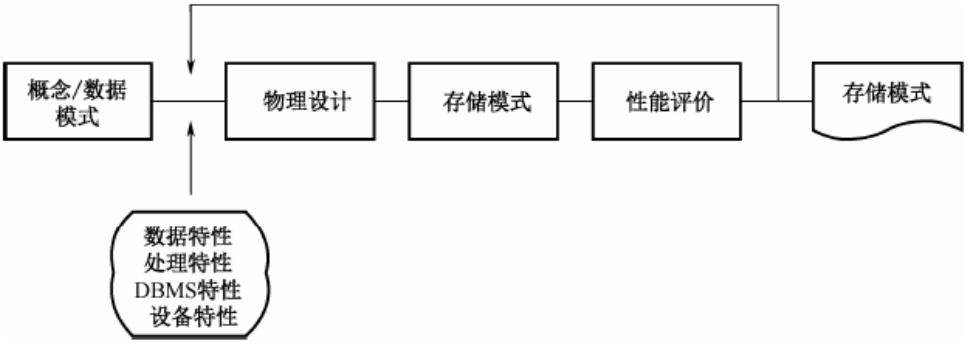


图 3-7 数据库物理设计过程

由于不同的 DBMS 提供的硬件环境和存储结构、存取方法，以及提供给数据库设计者的系统参数、变化范围有所不同，因此，为了设计出一个较好的存储模式，设计者必须了解以下几方面的问题，做到心中有数。

（1）了解并熟悉应用要求，包括各个用户对应的数据视图，即数据库的外模式（子模式），分清哪些是主要的应用，了解各个应用的使用方式、数据量和处理频率等，以便对时间和空间进行平衡，并保证优先满足应用的时间要求。

（2）熟悉使用的 DBMS 的性能，包括 DBMS 的功能，提供的物理环境、存储结构、存取方法和可利用的工具。

（3）了解存放数据的外存设备的特性，如物理存储区域的划分原则，物理块的大小等有关规定及 I/O 特性等。

存储模式和概念模式不一样，它不是面向用户的，一般的用户不一定也不需要了解数据库存储模式的细节。所以数据库存储模式的设计可以不必考虑用户理解的方便，其设计目标主要是提高数据库的性能，其次是节省存储空间。

在进行物理设计时，设计人员可能用到的数据库产品是多种多样的。不同的数据库产品所提供的物理环境、存储结构和存取方法有很大差别，能供设计人员使用的设计变量、参数范围也大不相同，因此没有通用的物理设计方法可遵循，只能给出一般的设计内容和原则。

3.4 事务管理

数据库系统运行的基本工作单位是事务，事务相当于操作系统中的进程，是用户定义的一个数据库操作序列，这些操作序列要么全做要么全不做，是一个不可分割的工作单位。事务具有以下特性：

（1）原子性（Atomicity）：数据库的逻辑工作单位。

（2）一致性（Consistency）：使数据库从一个一致性状态变到另一个一致性状态。

（3）隔离性（Isolation）：不能被其他事务干扰。

（4）持续性（永久性）（Durability）：一旦提交，改变就是永久性的。

事务通常以 BEGIN TRANSACTION（事务开始）语句开始，以 COMMIT 或 ROLLBACK 语句结束。COMMIT 称为“事务提交语句”，表示事务执行成功的结束。ROLLBACK 称为“事务回退语句”，表示事务执行不成功的结束。从终端用户来看，事务是一个原子，是不可分割的操作序列。事务中包括的所有操作要么都做，要么都不做（就效果而言）。事务不应该丢失或被分割地完成。

3.4.1 并发控制

在多用户共享系统中，许多事务可能同时对同一数据进行操作，称为“并发操作”，此时数据库管理系统的并发控制子系统负责协调并发事务的执行，保证数据库的完整性不受破坏，同时避免用户得到不正确的数据。

数据库的并发操作带来的问题有：丢失更新问题、不一致分析问题（读过时的数据）、依赖于未提交更新的问题（读了“脏”数据）。这三个问题需要 DBMS 的并发控制子系统来解决。

处理并发控制的主要方法是采用封锁技术。它有两种类型：排他型封锁（X 封锁）和共享型封锁（S 封锁），分别介绍如下：

（1）排他型封锁（简称 X 封锁）。如果事务 T 对数据 A（可以是数据项、记录、数据集，乃至整个数据库）实现了 X 封锁，那么只允许事务 T 读取和修改数据 A，其他事务要等事务 T 解除 X 封锁以后，才能对数据 A 实现任何类型的封锁。可见 X 封锁只允许一个事务独锁某个数据，具有排他性。

（2）共享型封锁（简称 S 封锁）。X 封锁只允许一个事务独锁和使用数据，要求太严。需要适当放宽，例如可以允许并发读，但不允许修改，这就产生了 S 封锁概念。S 封锁的含义是：如果事务 T 对数据 A 实现了 S 封锁，那么允许事务 T 读取数据 A，但不能修改数据 A，在所有 S 封锁解除之前绝不允许任何事务对数据 A 实现 X 封锁。

数据库是一个共享资源，它允许多个用户程序并行地存取数据库中的数据，但是，如果系统对并行操作不加以控制，就会存取不正确的数据，破坏数据库的完整性。

在多个事务并发执行的系统中，主要采取封锁协议来进行处理。

（1）一级封锁协议。事务 T 在修改数据 R 之前必须先对其加 X 锁，直到事务结束才释放。一级封锁协议可防止丢失修改，并保证事务 T 是可恢复的。但不能保证可重复读和不读“脏”数据。

（2）二级封锁协议。一级封锁协议加上事务 T 在读取数据 R 之前先对其加 S 锁，读完后即可释放 S 锁。二级封锁协议可防止丢失修改，还可防止读“脏”数据，但不能保证可重复读。

（3）三级封锁协议。一级封锁协议加上事务 T 在读取数据 R 之前先对其加 S 锁，直到事务结束才释放。三级封锁协议可防止丢失修改、防止读“脏”数据与防止数据重复读。

（4）两段锁协议。所有事务必须分两个阶段对数据项加锁和解锁。其中扩展阶段是在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；收缩阶段是在释放一

个封锁之后，事务不能再申请和获得任何其他封锁。若并发执行的所有事务均遵守两段封锁协议，则对这些事务的任何并发调度策略都是可串行化的。遵守两段封锁协议的事务可能发生死锁。

下面讨论封锁的粒度。所谓封锁的粒度即是被封锁数据目标的大小。在关系数据库中，封锁粒度有属性值、属性值集、元组、关系、某索引项（或整个索引）、整个关系数据库、物理页（块）等几种。

封锁粒度小则并发性高，但开销大；封锁粒度大则并发性低，但开销小。综合平衡照顾不同需求以合理选取适当的封锁粒度是很重要的。

采用封锁的方法固然可以有效防止数据的不一致性，但封锁本身也会产生一些麻烦，最主要的就是死锁问题。所谓死锁是指多个用户申请不同封锁，由于申请者均拥有一部分封锁权而又需等待另外用户拥有的部分封锁而引起的永无休止的等待。一般来讲，死锁是可以避免的，目前采用的办法有以下几种：

（1）预防法。此种方法是采用一定的操作方式以保证避免死锁的出现，顺序申请法、一次申请法等即是此类方法。所谓顺序申请法是指对封锁对象按序编号，在用户申请封锁时必须按编号顺序（从小到大或反之）申请，这样能避免死锁发生。所谓一次申请法即是指用户在一个完整操作过程中必须一次性申请它所需要的所有封锁，并在操作结束后一次性归还所有封锁，这样也能避免死锁的发生。

（2）死锁的解除法。此种方法允许产生死锁，并在死锁产生后通过解锁程序以解除死锁。这种方法需要有两个程序，一是死锁检测程序，用它测定死锁是否发生，另一是解锁程序，一旦经测定系统已产生死锁则启动解锁程序以解除死锁。有关死锁检测及解锁技术请参阅相应的资料，这里不做进一步讨论。

3.4.2 故障与恢复

数据库的故障可用事务的故障来表示，主要分为四类：

（1）事务故障。事务在运行过程中由于种种原因，如输入数据的错误、运算溢出、违反了某些完整性限制、某些应用程序的错误，以及并发事务发生死锁等，使事务未运行至正常终止点就被撤销，这种情况称为“事务故障”。

（2）系统故障。系统故障是指系统在运行过程中，由于某种原因（如操作系统或数据库管理系统代码错误、操作员操作失误、特定类型的硬件错误（如 CPU 故障）、突然停电

等造成系统停止运行），致使事务在执行过程中以非正常方式终止，这时内存中的信息丢失，但存储在外存储设备上的数据不会受影响。

（3）介质故障。系统在运行过程中，由于某种硬件故障，如磁盘损坏、磁头碰撞或由于操作系统的某种潜在的错误、瞬时强磁场干扰，使存储在外存上的数据部分损失或全部损失，称为“介质故障”。这类故障比前两类故障的可能性虽然小得多，但破坏性却最大。

（4）计算机病毒。计算机病毒是一种人为破坏计算机正常工作的特殊程序。通过读写染有病毒的计算机系统中的程序与数据，这些病毒可以迅速繁殖和传播，危害计算机系统和数据库。目前大多数病毒是在 PC 和其兼容机上传播的。有的病毒一侵入系统就马上摧毁系统，有的病毒有较长的潜伏期，有的病毒则只在特定的日期发生破坏作用，有的病毒感染系统所有的程序和数据，有的只影响特定的程序和数据。

在数据库系统中，恢复的基本含义就是恢复数据库本身。也就是说，在发生某种故障使数据库当前的状态已经不再正确时，把数据库恢复到已知为正确的某一状态。目前数据库系统中最常用的恢复方法是转储和登记日志文件，可根据故障的不同类型，采用不同的恢复策略。

2．故障的恢复

（1）事务故障的恢复。事务故障是指事务未运行至正常终止点前被撤销，这时恢复子系统应对此事务做撤销处理。事务故障的恢复是由系统自动完成的，不需要用户干预，步骤如下：

反向扫描文件日志，查找该事务的更新操作。

对该事务的更新操作执行逆操作。

继续反向扫描日志文件，查找该事务的其他更新操作，并做同样处理。

如此处理下去，直至读到此事务的开始标记，事务故障恢复完成。

（2）系统故障的恢复。系统故障发生时，造成数据库不一致状态的原因有两个：一是由于一些未完成事务对数据库的更新已写入数据库；二是由于一些已提交事务对数据库的更新还留在缓冲区没来得及写入数据库。系统故障的恢复是在重新启动时自动完成的，不需要用户干预，步骤如下：

正向扫描日志文件，找出在故障发生前已经提交的事务，将其事务标识记入重做（Redo）队列。同时找出故障发生时尚未完成的事务，将其事务标识记入撤销（Undo）队列。

对撤销队列中的各个事务进行撤销处理：反向扫描日志文件，对每个 Undo 事务的更新操作执行逆操作。

对重做队列中的各个事务进行重做处理：正向扫描日志文件，对每个 Redo 事务重新执行日志文件登记的操作。

（3）介质故障与病毒破坏的恢复。在发生介质故障和遭病毒破坏时，磁盘上的物理数据库被破坏，这时的恢复操作可分为三步：

装入最新的数据库后备副本，使数据库恢复到最近一次转储时的一致性状态。

从故障点开始反向读日志文件，找出已提交事务标识将其记入重做队列。

从起始点开始正向阅读日志文件，根据重做队列中的记录，重做所有已完成事务，将数据库恢复至故障前某一时刻的一致状态。

（4）具有检查点的恢复技术。检查点记录的内容可包括：

建立检查点时刻所有正在执行的事务清单。

这些事务最近一个日志记录的地址。采用检查点的恢复步骤如下：

从重新开始文件中找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录。

由该检查点记录得到检查点建立时所有正在执行的事务清单队列（A）。

建立重做队列（R）和撤销队列（U），把 A 队列放入 U 队列中，R 队列为空。

从检查点开始正向扫描日志文件，若有新开始的事务 T1，则把 T1 放入 U 队列，若有提交的事务 T2，则把 T2 从 U 队列移到 R 队列，直至日志文件结束。

对 U 队列的每个事务执行 Undo 操作，对 R 队列的每个事务执行 Redo 操作。

DBA 要做的基本操作是：

重装最近转储的后援副本。

运行日志文件，执行系统提供的恢复命令。

数据库安全和恢复是数据库系统正常运行的保证。大型数据库管理系统一般都提供了实现安全机制的保证，即由系统提供了相应的功能，但小型的数据库管理系统并非都具有相应功能，因此有时需要人工的辅助措施，用以保证数据库的安全和恢复。

3.5 备份与恢复

数据库中的数据一般都十分重要，不能丢失，因为各种原因，数据库都有损坏的可能性（虽然很小），所以事先制定一个合适的、可操作的备份和恢复计划至关重要。备份和恢复计划的制订要遵循以下两个原则：

（1）保证数据丢失的情况尽量少或完全不丢失，因为性价比的要求，这要取决于现实系统的具体要求。

（2）备份和恢复时间尽量短，保证系统最大的可用性。数据库备份按照不同方式可分为多种，这里按照备份内容分为物理备份和逻辑备份两类。

物理备份是在操作系统层面上对数据库的数据文件进行备份，物理备份分为冷备份和热备份两种。冷备份是将数据库正常关闭，在停止状态下利用操作系统的 copy、cp、tar、 cpio 等命令将数据库的文件全部备份下来，当数据库发生故障时，将数据文件复制回来，进行恢复。热备份也分为两种，一种是不关闭数据库，将数据库中需要备份的数据文件依次置于备份状态，相对保持静止，然后再利用操作系统的 copy、cp、tar、cpio 等命令将数据库的文件备份下来，备份完毕后再将数据文件恢复为正常状态，当数据库发生故障时，恢复方法同冷备份一样。热备份的另外一种方式是利用备份软件（例如，veritas 公司的 netbackup，legato 公司的 network 等）在数据库正常运行的状态下，将数据库中的数据文件备份出来。

为了提高物理备份的效率，通常将完全、增量、累积三种备份方式相组合。完全备份是将数据库的内容全部备份，作为增量、累积的基础；增量备份是只备份上次完全、增量或累积备份以来修改的数据；累积备份是备份自上次完全或累积备份以来修改过的数据。一个备份周期通常由一个完全备份和多个增量、累积备份组成。由于增量或累计备份导出的数据少，所以其导出的文件较小，所需要的时间较少。利用一个完全备份和多个增量、累积备份恢复数据库的步骤如下：

（1）首先从完全备份恢复数据库。

（2）然后按照时间顺序从早到晚依次导入多个增量和累积备份文件。

逻辑备份是指利用各数据库系统自带的工具软件备份和恢复数据库的内容，例如，Oracle 的导出工具为 exp，导入工具为 imp，可以按照表、表空间、用户、全库等四个层次备份和恢复数据；Sybase 的全库备份命令是 dump database，全库恢复命令是 load database，另外也可利用 BCP 命令来备份和恢复指定表。

在数据库容量不大的情况下逻辑备份是一个非常有效的手段，既简单又方便，但现在随着数据量的越来越大，利用逻辑备份来备份和恢复数据库已力不从心，速度也很慢。针对大型数据库的备份和恢复一般结合磁带库采用物理的完全、增量、累积三种备份方式相组合来进行。但无论任何时候逻辑备份都是一种非常有效的手段，特别适合于日常维护中的部分指定表的备份和恢复。

3.6 分布式数据库系统

近年来，随着计算机技术与网络技术的发展，特别是 Internet 的兴起，分布式数据库系统得到了很快的发展和应用。

3.6.1 分布式数据库的概念

分布式数据库系统是相对于集中式数据库系统而言的，是将数据库技术与网络技术相结合的产物。分布式数据库（Distributed DataBase，DDB）比较确切的定义是：分布式数据库是由一组数据组成的，这组数据分布在计算机网络的不同计算机上，网络中的每个结点具有独立处理的能力，成为场地自治，它可以执行局部应用，同时，每个结点也能通过网络通信子系统执行全局应用。负责分布式数据库的建立、查询、更新、复制、管理和维护的软件，称为分布式数据库管理系统（Distributed DataBase Management System， DDBMS）。分布式数据库管理系统保证分布式数据库中数据的物理分布对用户的透明性。一个计算机网络组成的计算机系统，在配置了分布式数据库管理系统，并在其上建立了分布式数据库和相应的应用程序后，就称其为分布式数据库系统（Distributed DataBase System，DDBS）。分布式数据库管理系统是分布式数据库系统的核心。

1．分布式数据库的特点从上面的定义可以看出分布式数据库系统有以下几个特点：

（1）数据的分布性。分布式数据库中的数据分布于网络中的各个结点，它既不同于传统的集中式数据库，也不同于通过计算机网络共享的集中式数据库系统。

（2）统一性。主要表现在数据在逻辑上的统一性和数据在管理上的统一性两个方面。分布式数据库系统通过网络技术把局部的、分散的数据库构成一个在逻辑上单一的数据库，从而呈现在用户面前的就如同是一个统一的、集中式的数据库。这就是数据在逻辑上的统一性，因此，它不同于由网络互联的多个独立数据库。分布式数据库是由分布式数据库管理系统统一管理和维护的，这种管理上的统一性又使它不同于一般的分布式文件系统。

（3）透明性。用户在使用分布式数据库时，与使用集中式数据库一样，无须知道其所关心的数据存放在哪里，存储了几次。用户需要关心的仅仅是整个数据库的逻辑结构。

与集中式数据库相比，分布式数据库具有下列优点：

（1）坚固性好。由于分布式数据库系统是由多个位置上的多台计算机构成的，在个别结点或个别通信链路发生故障的情况下，它仍然可以降低级别继续工作，如果采用冗余技术，还可以获得一定的容错能力。因此，系统的坚固性好，即系统的可靠性和可用性好。

（2）可扩充性好。可根据发展的需要增减结点，或对系统重新配置，这比用一个更大的系统代替一个已有的集中式数据库要容易得多。

（3）可改善性能。在分布式数据库中可按就近分布，合理地冗余的原则来分布各结点上的数据，构造分布式数据库，使大部分数据可以就近访问，避免了集中式数据库中的瓶颈问题，减少了系统的响应时间，提高了系统的效率，而且也降低了通信费用。

（4）自治性好。数据可以分散管理，统一协调，即系统中各结点的数据操纵和相互作用是高度自治的，不存在主从控制，因此，分布式数据库较好地满足了一个单位中各部门希望拥有自己的数据，管理自己的数据，同时又想共享其他部门有关数据的要求。

虽然分布式数据库系统与集中式数据库相比有不少优点，但同时也需要解决一些集中式数据库所没有的问题。首先，异构数据库的集成问题是一项比较复杂的技术问题，目前还很难用一个通用的分布式数据库管理系统来解决这一问题。其次，如果数据库设计得不好，数据分布不合理，以致远距离访问过多，尤其是分布连接操作过多，不但不能改善性能，反而会使性能降低。

2．分布式数据库的分类

分布式数据库及其分布式数据库管理系统，根据许多因素有不同的分类方法，总的原则是分布式数据库及 DDBMS 必须是其数据和软件必定分布在用计算机网络连接的多个场地上。从应用需要或本身的特征方面考虑可将它从以下几个方面来划分：

（1）按 DDBMS 软件同构度来分。当所有服务器软件（或每个 LDBMS）和所有客户软件均用相同的软件时称为同构型分布式数据库；反之，则称为异构型分布式数据库。

（2）按局部自治度来分。当对 DDBMS 的存取必须通过客户软件，则系统称为无局部自治；当局部事务允许对服务器软件进行直接存取，则系统称为有一定的局部自治。自治的两个分别是无局部自治和联邦型 DDBMS 或称多数据库系统。多数据库系统本质上是集中式与分布式的混合体：对一个局部用户而言，它是自治的，那么是一个集中式 DBS；对一个全局用户而言，则是一个分布式 DBS，但这个 DDBS 没有全局概念模式，只有一个由各局部数据库提供给全局允许共享的有关模式的集成。

（3）按分布透明度来分。分布透明度的另一个概念是模式集成度。若用户可以对集成模式操作不需要涉及任何片段、重复、分布等信息时，则这类 DDBMS 称为有高度分布透明（或高度模式集成）；若用户必须知道所有关于片段、分配、重复等信息时，则这类 DDBMS 没有分布透明，没有模式集成度。当系统不提供分布透明，用户查询时必须指定特定的场地、特定的片段等信息，当然 DDBMS 可以部分分布透明（介于两者之间）。

3．分布式数据库的目标

理想的分布式系统使用时应该精确得像一个非分布式系统。概括起来有以下 12 条具体规则和目标：

（1）局部结点自治性。网络中的每个结点是独立的数据库系统，它有自己的数据库，运行它的局部 DBMS，执行局部应用，具有高度的自治性。

（2）不依赖中心结点。即每个结点具有全局字典管理、查询处理、并发控制和恢复控制等功能。

（3）能连续操作。该目标使中断分布式数据库服务情况减至最少，当一个新场地合并到现有的分布式系统或从分布式系统中撤离一个 场地不会导致任何不必要的服务中断；在分布式系统中可动态地建立和消除片段，而不中止任何组成部分的场地或数据库；应尽可能在不使整个系统停机的情况下对组成分布式系统的场地的 DBMS 进行升级。

（4）具有位置独立性（或称位置透明性）。用户不必知道数据的物理存储地，可工作可像数据全部存储在局部场地一样。一般位置独立性需要有分布式数据命名模式和字典子系统的支持。

（5）分片独立性（或称分片透明性）。分布式系统如果可将给定的关系分成若干块或片，可提高系统的处理性能。利用分片将数据存储在最频繁使用它的位置上，使大部分操作为局部操作，减少网络的信息流量。如果系统支持分片独立性，那么用户工作起来就像数据全然不是分片的一样。

（6）数据复制独立性。是指将给定的关系（或片段）可在物理级用许多不同存储副本或复制品在许多不同场地上存储。支持数据复制的系统应当支持复制独立性，用户工作可像它全然没有存储副本一样地工作。

（7）支持分布式查询处理。在分布式数据库系统中有三类查询：局部查询、远程查询和全局查询。局部查询和远程查询仅涉及单个结点的数据（本地的或远程的），查询优化采用的技术是集中式数据库的查询优化技术。全局查询涉及多个结点上的数据，其查询处理和优化要复杂得多。

（8）支持分布事务管理。事务管理有两个主要方面：恢复控制和并发控制。在分布式系统中，单个事务会涉及多个场地上的代码执行，会涉及多个场地上的更新，可以说每个事务是由多个“代理”组成的，每个代理代表在给定场地上的给定事务上执行的过程。在分布式系统中必须保证事务的代理集或者全部一致交付，或者全部一致回滚。

（9）具有硬件独立性。希望在不同硬件系统上运行同样的 DBMS。

（10）具有操作系统独立性。希望在不同的操作系统上运行 DBMS。

（11）具有网络独立性。如果系统能够支持多个不同的场地，每个场地有不同的硬件和不同的操作系统，则要求该系统能支持各种不同的通信网络。

（12）具有 DBMS 独立性。实现对异构型分布式系统的支持。理想的分布式系统应该提供 DBMS 独立性。

上述的全功能分布式数据库系统的准则和目标起源于：一个分布式数据库系统，对用户来说，应当看上去完全像一个非分布式系统。值得指出的是，现实系统出于对某些方面的特别考虑，对上述各方面做出了种种权衡和选择。

3.6.2 分布式数据库的架构

分布式数据库系统的模式结构有六个层次，如图 3-8 所示，实际的系统并非都具有这种结构。在这种结构中各级模式的层次清晰，可以概括和说明任何分布式数据库系统的概念和结构。

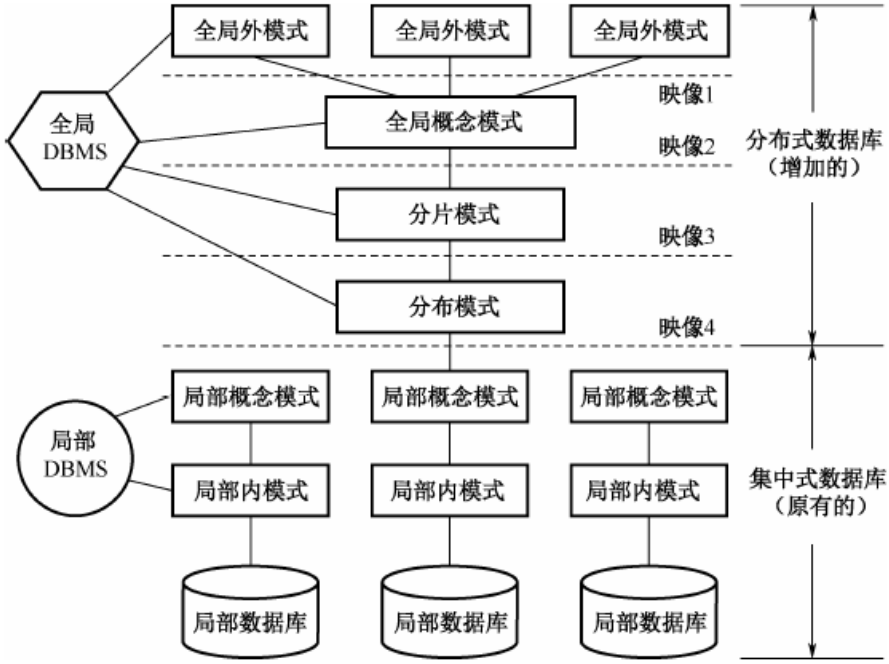


图 3-8　分布式数据库系统的模式结构

图 3-8 的模式结构从整体上可以分为两大部分：下半部分是集中式数据库的模式结构，代表了各局部场地上局部数据库系统的基本结构；上半部分是分布式数据库系统增加的模式级别。

（1）全局外模式。它们是全局应用的用户视图，是全局概念模式的子集。

（2）全局概念模式。它定义分布式数据库中数据的整体逻辑结构，数据就如同根本没有分布一样，可用传统的集中式数据库中所采用的方法定义。全局概念模式中所用的数据模型应该易于向其他层次的模式映像，通常采用关系模型。这样，全局概念模式包括一组全局关系的定义。

（3）分片模式。每一个全局关系可以划分为若干不相交的部分，每一部分称为一个片段，即“数据分片”。分片模式就是定义片段及全局关系到片段的映像。这种映像是一对多的，即每个片段来自一个全局关系，而一个全局关系可对应多个片段。

（4）分布模式。由数据分片得到的片断仍然是 DDB 的全局数据，是全局关系的逻辑部分，每一个片段在物理上可以分配到网络的一个或多个不同结点上。分布模式定义片段的存放结点。分布模式的映像类型确定了分布式数据库是冗余的还是非冗余的。若映像是一对多的，即一个片段分配到多个结点上存放，则是冗余的分布数据库，否则是不冗余的分布数据库。

根据分布模式提供的信息，一个全局查询可分解为若干子查询，每一子查询要访问的数据属于同一场地的局部数据库。由分布模式到各局部数据库的映像（映像 4）把存储在局部场地的全局关系或全局关系的片段映像为各局部概念模式采用局部场地的 DBMS 所支持的数据模型。

分片模式和分布模式均是全局的，分布式数据库系统中增加的这些模式和相应的映像使分布式数据库系统具有了分布透明性。

（5）局部概念模式。一个全局关系经逻辑划分成一个或多个逻辑片断，每个逻辑片断被分配在一个或多个场地上，称为该逻辑片断在某场地上的物理映像或物理片断。分配在同一场地上的同一个全局概念模式的若干片断（物理片断）构成了该全局概念在该场地上的一个物理映像。

一个场地上的局部概念模式是该场地上所有全局概念模式在该场地上物理映像的集合。由此可见，全局概念模式与场地独立，而局部概念模式与场地相关。

（6）局部内模式。局部内模式是 DDB 中关于物理数据库的描述，类似于集中式 DB 中的内模式，但其描述的内容不仅包含局部本场地的数据的存储描述，还包括全局数据在本场地的存储描述。

在图 3-8 的六层模式结构中，全局概念模式、分片模式和分布模式是与场地特征无关的，是全局的，因此它们不依赖于局部 DBMS 的数据模型。在低层次上，需要把物理映像映射成由局部 DBMS 支持的数据模型。这种映像由局部映射模式完成。具体的映射关系，

由局部 DBMS 的类型决定。在异构型系统中，可在不同场地上拥有类型的局部映射模式。

这种分层的模式结构为理解 DDB 提供了一种通用的概念结构。它有三个显著的特征：

（1）数据分片和数据分配概念的分离，形成了“数据分布独立型”概念。

（2）数据冗余的显示控制。数据在各个场地的分配情况在分配模式中一目了然，便于系统管理。

（3）局部 DBMS 的独立性。这个特征也称为“局部映射透明性”。此特征允许在不考虑局部 DBMS 专用数据模型的情况下研究 DDB 管理的有关问题。

1．分布式数据库系统与并行数据库系统的区别

分布式数据库系统与并行数据库系统具有很多相似点：它们都是通过网络连接各个数据处理结点的，整个网络中的所有结点构成一个逻辑上统一的整体，用户可以对各个结点上的数据进行透明存取等。但分布式数据库系统与并行数据库系统之间还是存在着显著的区别的，主要表现在以下几个方面：

（1）应用目标不同。并行数据库系统的目标是充分发挥并行计算机的优势，利用系统中的各个处理机结点并行地完成数据库任务，提高数据库的整体性能。分布式数据库系统主要目的在于实现各个场地自治和数据的全局透明共享，而不要求利用网络中的各个结点来提高系统的整体性能。

（2）实现方式不同。由于应用目标各不相同，在具体实现方法上，并行数据库与分布式数据库之间也有着较大的区别。在并行数据库中，为了充分发挥各个结点的处理能力，各结点间采用高速通信网络互联，结点间数据传输代价相对较低。当负载不均衡时，可以将工作负载过大的结点上的任务通过高速通信网络送给空闲结点处理，从而实现负载平衡。在分布式数据库系统中，各结点（场地）间一般通过局域网或广域网互联，网络带宽比较低，各场地之间的通信开销较大，因此在查询处理时一般应尽量减少结点间的数据传输量。

（3）各结点的地位不同。在并行数据库中，各结点之间不存在全局应用和局部应用的概念。各个结点协同作用，共同处理，而不可能有局部应用。

在分布式数据库系统中，各结点除了能通过网络协同完成全局事务外，还有自己结点场地的自治性。也就是说，分布式数据库系统的每个场地又是一个独立的数据库系统，除了拥有自己的硬件系统（CPU、内存和磁盘等）外，还拥有自己的数据库和自己的客户，可运行自己的 DBMS，执行局部应用，具有高度的自治性。这是并行数据库与分布式数据库之间最主要的区别。

2．数据分片和透明性

将数据分片，使数据存放的单位不是关系而是片段，这既有利于按照用户的需求较好地组织数据的分布，也有利于控制数据的冗余度。分片的方式有多种，水平分片和垂直分片是两种基本的分片方式，混合分片和导出分片是较复杂的分片方式。

分布透明性指用户不必关心数据的逻辑分片，不必关心数据存储的物理位置分配细节，也不必关心局部场地上数据库的数据模型。从图 3-8 的模式结构可以看到分布透明性包括：分片透明性、位置透明性和局部数据模型透明性。

（1）分片透明性是分布透明性的最高层次。所谓分片透明性是指用户或应用程序只对全局关系进行操作而不必考虑数据的分片。当分片模式改变时，只要改变全局模式到分片模式的映像（映像 2），而不影响全局模式和应用程序。全局模式不变，应用程序不必改写，这就是分片透明性。

（2）位置透明性是分布透明性的下一层次。所谓位置透明性是指，用户或应用程序应当了解分片情况，但不必了解片段的存储场地。当存储场地改变时，只要改变分片模式到分配模式的映像（映像 3），而不影响应用程序。同时，若片段的重复副本数目改变了，那么数据的冗余也会改变，但用户不必关心如何保持各副本的一致性，这也提供了重复副本的透明性。

（3）局部数据模型透明性是指用户或应用程序应当了解分片及各片断存储的场地，但不必了解局部场地上使用的是何种数据模型。模型的转换及语言等的转换均由映像 4 来完成。

3．分布式数据库管理系统分布式数据库管理系统的任务，首先就是把用户与分布式数据库隔离开来，使其对用户而言，整个分布式数据库就好像是一个传统的集中式数据库。换句话说，一个分布式数据库管理系统与用户之间的接口，在逻辑上与集中式数据库管理系统是一致的。但是考虑到分布式数据库的特点，其物理实现上又与集中式数据库不同。下面以一种分布式数据库管理。

以系统 DDBMS 的结构为例来分析它的主要成分和功能，如图 3-9 所示。

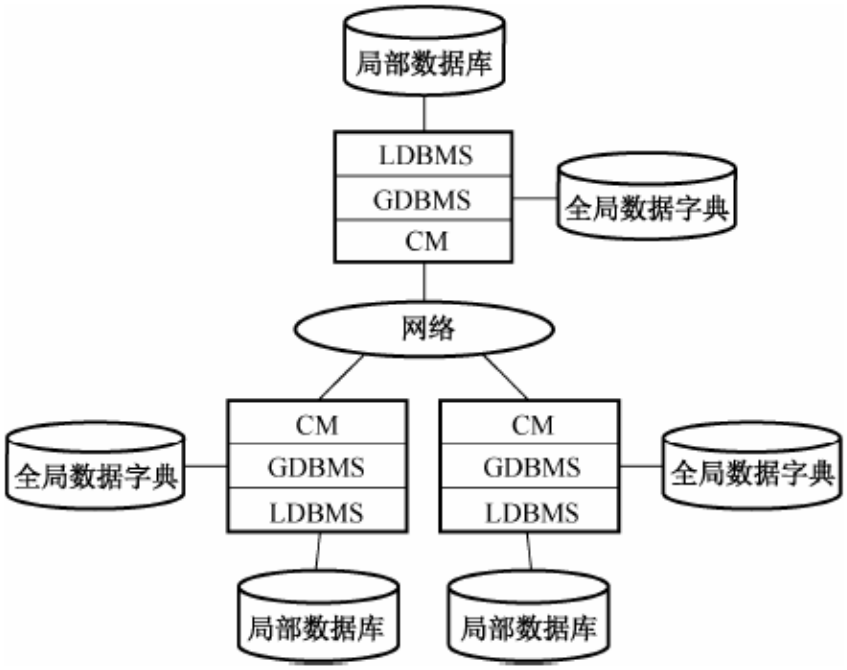


图 3-9　分布式数据库管理系统的结构

由图 3-9 可以看出，DDBMS 由 4 部分组成：

（1）LDBMS（局部 DBMS）。局部场地上的数据库管理系统的功能是建立和管理局部数据库，提供场地自治能力、执行局部应用及全局查询的子查询。

（2）GDBMS（全局 DBMS）。全局数据库管理系统的主要功能是提供分布透明性，协调全局事务的执行，协调各局部 DBMS 以完成全局应用，保证数据库的全局一致性，执行并发控制，实现更新同步，提供全局恢复功能。

（3）全局数据字典。存放全局概念模式、分片模式、分布模式的定义及各模式之间映像的定义；存放有关用户存取权限的定义，以保证全局用户的合法权限和数据库的安全性；存放数据完整性约束条件的定义，其功能与集中式数据库的数据字典类似。

（4）CM（Communication Management，通信管理）。在分布数据库各场地之间传送消息和数据，完成通信功能。

DDBMS 功能的分割和重复及不同的配置策略就导致了各种架构。

（1）全局控制集中的 DDBMS。这种结构的特点是全局控制成分 GDBMS 集中在某一结点上，由该结点完成全局事务的协调和局部数据库转换等一切控制功能，全局数据字典只有一个，也存放在该结点上，它是 GDBMS 执行控制的依据。它的优点是控制简单，易实现更新一致性。但由于控制集中在某一特定的结点上，不仅容易形成瓶颈而且系统较脆弱，一旦该结点出故障，整个系统就会瘫痪。

（2）全局控制分散的 DDBMS。这种结构的特点是全局控制成分 GDBMS 分散在网络的每一个结点上，全局数据字典也在每个结点上有一份，每个结点都能完成全局事务的协调和局部数据库转换，每个结点既是全局事务的参与者又是协调者，一般称这类结构为完全分布的 DDBMS。它的优点是结点独立，自治性强，单个结点退出或进入系统均不会影响整个系统的运行，但是全局控制的协调机制和一致性的维护都比较复杂。

（3）全局控制部分分散的 DDBMS。这种结构是根据应用的需要将 GDBMS 和全局数据字典分散在某些结点上，是介于前两种情况之间的架构。

局部 DBMS 的一个重要性质是：局部 DBMS 是同构的还是异构的。同构和异构的级别可以有三级：硬件、操作系统和局部 DBMS。其中最主要的是局部 DBMS 这一级，因为硬件和操作系统的不同将由通信软件处理和管理。

异构型 DDBMS 的设计和实现比同构型 DDBMS 更加复杂，它要解决不同的 DBMS 之间及不同的数据模型之间的转换。因此在设计和实现 DDBMS 时，若是用自顶向下的方法进行，即并不存在已运行的局部数据库，则采用同构型的结构比较方便。若是采用自底向上设计 DDBMS 的方法，即现已存在的局部数据库，而这些数据库可能采用不同的数据模型（层次、网状或关系），或者虽然模型相同但它们是不同厂商的 DBMS（如 Informix、Sybase、Db2、Oracle），这就必须开发异构型的 DDBMS。要解决异构数据库模型的同种化问题，是研制异构型 DDBMS 的关键所在，所谓同种化就是寻找合适的公共数据模型，采用公共数据模型与异构数据模型（局部）之间的转换，不采用各结点之间的一对一转换。这样可以减少转移次数。设有 N 个结点，用公共数据模型时转换次数为 2N，而各结点之间一对一转换则需 N（N1）次。

3.7 数据仓库

传统的操作型数据库主要是面向业务的，所执行的操作基本上也是联机事务处理，但随着企业规模的增长，历史积累的数据越来越多，如何利用历史数据来为未来决策服务，就显得越来越重要了，而数据仓库就是其中的一种技术。

3.7.1 数据仓库的概念

著名的数据仓库专家 W.H.Inmon 在《Building the Data Warehouse》一书中将数据仓库定义为：数据仓库（Data Warehouse）是一个面向主题的、集成的、相对稳定的、且随时间

变化的数据集合，用于支持管理决策。

1．面向主题的操作型数据库的数据组织面向事务处理任务（面向应用），各个业务系统之间各自分离，而数据仓库中的数据是按照一定的主题域进行组织的。主题是一个抽象的概念，是指用户使用数据仓库进行决策时所关心的重点方面，一个主题通常与多个操作型信息系统相关。例如，一个保险公司所进行的事务处理（应用问题）可能包括汽车保险、人寿保险、健康保险和意外保险等，而公司的主要主题范围可能是顾客、保险单、保险费和索赔等。

2．集成的在数据仓库的所有特性中，这是最重要的。面向事务处理的操作型数据库通常与某些特定的应用相关，数据库之间相互独立，并且往往是异构的。而数据仓库中的数据是在对原有分散的数据库数据抽取、清理的基础上经过系统加工、汇总和整理得到的，必须消除源数据中的不一致性，以保证数据仓库内的信息是关于整个企业的一致的全局信息。

3．相对稳定的操作型数据库中的数据通常实时更新，数据根据需要及时发生变化。数据仓库的数据主要供企业决策分析之用，所涉及的数据操作主要是数据查询，一旦某个数据进入数据仓库以后，一般情况下将被长期保留，也就是数据仓库中一般有大量的查询操作，但修改和删除操作很少，通常只需要定期地加载、刷新。

4．随时间变化的操作型数据库主要关心当前某一个时间段内的数据，而数据仓库中的数据通常包含历史信息，系统记录了企业从过去某一时点（如开始应用数据仓库的时点）到目前的各个阶段的信息，通过这些信息，可以对企业的发展历程和未来趋势做出定量分析和预测。数据仓库反映历史变化的属性主要表现在：

（1）数据仓库中的数据时间期限要远远长于传统操作型数据系统中的数据时间期限，传统操作型数据系统中的数据时间期限可能为数十天或数个月，数据仓库中的数据时间期限往往为数年甚至几十年；

（2）传统操作型数据系统中的数据含有“当前值”的数据，这些数据在访问时是有效的，当然数据的当前值也能被更新，但数据仓库中的数据仅仅是一系列某一时刻（可能是传统操作型数据系统）生成的复杂的快照；

（3）传统操作型数据系统中可能包含也可能不包含时间元素，如年、月、日、时、分、秒等，而数据仓库中一定会包含时间元素。

数据仓库虽然是从传统数据库系统发展而来，但是两者还是存在着诸多差异，如：从数据存储的内容看，数据库只存放当前值，而数据仓库则存放历史值；数据库数据的目标是面向业务操作人员的，为业务处理人员提供数据处理的支持，而数据仓库则是面向中高层管理人员的，为其提供决策支持等。表 3-10 详细说明了数据仓库与传统数据库的区别。

表 3-10　数据仓库与传统数据库的比较

比较项目	传统数据库	数据仓库
数据内容	当前值	历史的、归档的、归纳的、计算的数据（处理过的）
数据目标	面向业务操作程序、重复操作	面向主体域，分析应用
数据特性	动态变化、更新	静态、不能直接更新，只能定时添加、更新
数据结构	高度结构化、复杂，适合操作计算	简单、适合分析
使用频率	高	低
数据访问量	每个事务一般只访问少量记录	每个事务一般访问大量记录
对响应时间的要求	计时单位小，如秒	计时单位相对较大，除了秒，还有分钟、小时

3.7.2　数据仓库的结构

从数据仓库的概念结构看，一般来说，数据仓库系统要包含数据源、数据准备区、数据仓库数据库、数据集市/知识挖掘库及各种管理工具和应用工具，如图 3-10 所示。数据仓库建立之后，首先要从数据源中抽取相关的数据到数据准备区，在数据准备区中经过净化处理后再加载到数据仓库数据库，最后根据用户的需求将数据导入数据集市和知识挖掘库中。当用户使用数据仓库时，可以利用包括 OLAP（On-Line Analysis Processing，联机分析处理）在内的多种数据仓库应用工具向数据集市/知识挖掘库或数据仓库进行决策查询分析或知识挖掘。数据仓库的创建、应用可以利用各种数据仓库管理工具辅助完成。

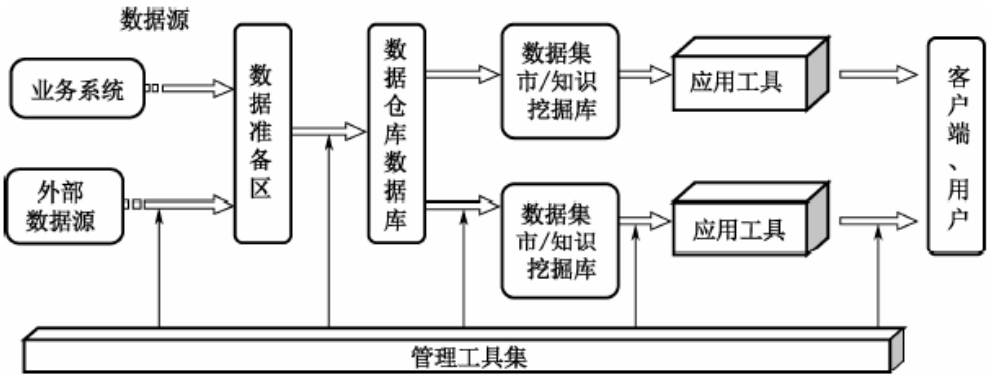


图 3-10　数据仓库的概念结构

1．数据仓库的参考框架

数据仓库的参考框架由数据仓库基本功能层、数据仓库管理层和数据仓库环境支持层组成，如图 3-11 所示。

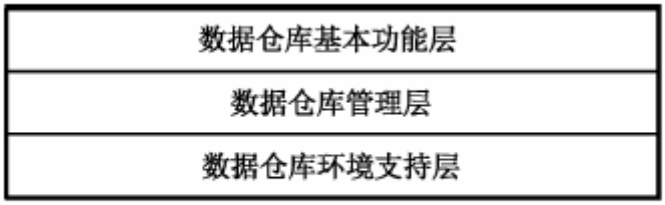


图 3-11　数据仓库的框架结构

（1）数据仓库基本功能层。数据仓库的基本功能层部分包含数据源、数据准备区、数据仓库结构、数据集市或知识挖掘库，以及存取和使用部分。本层的功能是从数据源抽取数据，对所抽取的数据进行筛选、清理，将处理过的数据导入或者说加载到数据仓库中，根据用户的需求设立数据集市，完成数据仓库的复杂查询、决策分析和知识的挖掘等。

（2）数据仓库管理层。数据仓库的正常运行除了需要数据仓库功能层提供的基本功能外，还需要对这些基本功能进行管理与支持的结构框架。数据仓库管理层由数据仓库的数据管理和数据仓库的元数据管理组成。

数据仓库的数据管理层包含数据抽取、新数据需求与查询管理，数据加载、存储、刷新和更新系统，安全性与用户授权管理系统及数据归档、恢复及净化系统等四部分。

（3）数据仓库的环境支持层。数据仓库的环境支持层由数据仓库数据传输层和数据仓库基础层组成。数据仓库中不同结构之间的数据传输需要数据仓库的传输层来完成。

数据仓库的传输层包含数据传输和传送网络、客户/服务器代理和中间件、复制系统及数据传输层的安全保障系统。

2．数据仓库的架构大众观点的数据仓库的架构如图 3-12 所示。

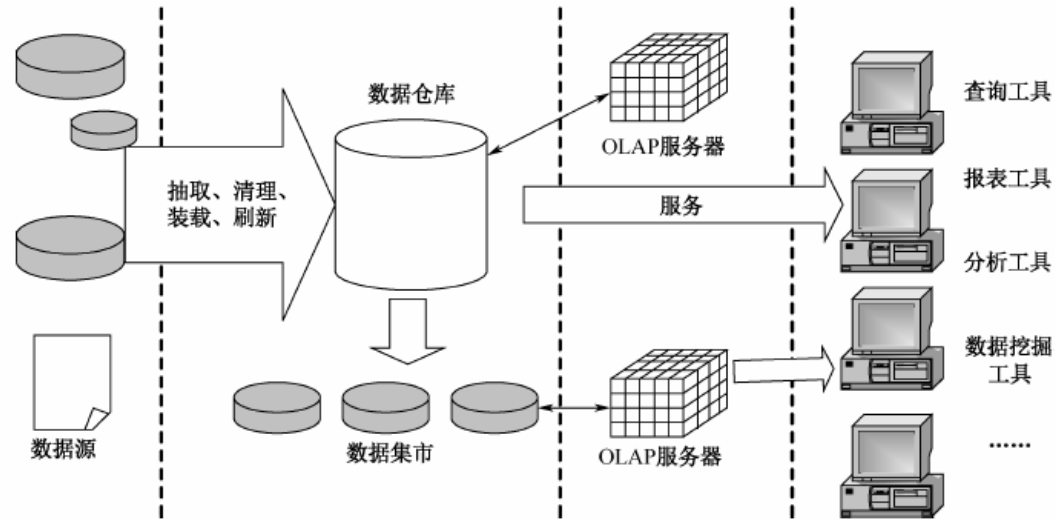


图 3-12　数据仓库架构

（1）数据源。是数据仓库系统的基础，是整个系统的数据源泉。通常包括企业内部信息和外部信息。内部信息包括存放于 RDBMS（关系型 DBMS）中的各种业务处理数据和各类文档数据。外部信息包括各类法律法规、市场信息和竞争对手的信息等。

（2）数据的存储与管理。是整个数据仓库系统的核心。数据仓库的真正关键是数据的存储和管理。数据仓库的组织管理方式决定了它有别于传统数据库，同时也决定了其对外部数据的表现形式。要决定采用什么产品和技术来建立数据仓库的核心，则需要从数据仓库的技术特点着手分析。针对现有各业务系统的数据，进行抽取、清理，并有效集成，按照主题进行组织。数据仓库按照数据的覆盖范围可以分为企业级数据仓库和部门级数据仓库（通常称为数据集市）。

（3）OLAP 服务器。对分析需要的数据进行有效集成，按多维模型予以组织，以便进行多角度、多层次的分析，并发现趋势。其具体实现可以分为：ROLAP、MOLAP 和 HOLAP。ROLAP 基本数据和聚合数据均存放在 RDBMS 之中；MOLAP 基本数据和聚合数据均存放于多维数据库中；HOLAP 基本数据存放于 RDBMS 之中，聚合数据存放于多维数据库中。

（4）前端工具。主要包括各种报表工具、查询工具、数据分析工具、数据挖掘工具及各种基于数据仓库或数据集市的应用开发工具。其中数据分析工具主要针对 OLAP 服务器，报表工具、数据挖掘工具主要针对数据仓库。

3.7.3 数据仓库的实现方法

数据仓库的特性决定了数据仓库的设计不同于传统的数据库设计方法。数据仓库系统的原始需求通常不是很明确，并且需求仍在不断变化、增加，所以，数据仓库的建立是一个过程，从建立简单的基本框架着手，不断丰富和完善整个系统。这一过程将由以下几部分构成：需求分析、概念模型设计、逻辑模型设计、物理模型设计和数据仓库生成。

从整体的角度来看，数据仓库的实现方法主要有自顶向下法、自底向上法和联合方法。

1．自顶向下法

在该方法中，首先应找出数据仓库解决方案所要满足的商业需求，把商业需求视为实现数据仓库的首要任务。数据仓库是一种功能而不是一种特征，数据仓库保存信息，并以外部工具易于显示和操作的方式组织这些信息。因此，如果不借助于可以利用这种功能的外部工具，最终用户就无法将这种功能嵌入数据仓库中。这样，就很难定出该功能的范围，除非用广义上的商业术语，如“数据仓库将包含有关客户、供应商、市场、产品的信息”。自顶向

下方法的优点和缺点如表 3-11 所示。

规划和实现数据仓库的自顶向下方法一般用于以下情况：

表 3-11　自顶向下方法的优缺点

优　点	缺　点
商业需求清楚地描绘出数据仓库实现的范围，因此是实现数据仓库解决方案的有效方法	机会有时超出了当前的业务范围
技术取决于商业	技术可以促进商业和竞争优势，但开始时对商业的促进是不明显的
易于向决策者提供数据仓库的收益情况	一旦数据仓库已经实现，可能就不再要求更高的目标

（1）实现单位比较熟悉技术，并具有根据商业需求采用自顶向下方法开发应用程序的丰富经验。

（2）决策层（总经理、决策者、投资者）完全清楚数据仓库的预测目标。

（3）决策层（总经理、决策者、投资者）完全清楚数据仓库用作哪些机构的决策支持工具。

（4）决策层（总经理、决策者、投资者）完全清楚数据仓库已经是商业过程中的一个子过程。

如果技术是成熟的和众所周知的，或者必须解决的商业问题是显而易见的，那么自顶向下方法是很有用的。采用自顶向下方法可以将技术和商业目标有机地结合起来。

2．自底向上法

自底向上方法一般从实验和基于技术的原形入手。先选择一个特定的、众所周知的商业问题的子集，再为该子集制订方案。实现自底向上一般是比较快的。自底向上可以使一个单位在发展时用尽可能少的经费和时间，就可以在做出有效的投入之前评估技术的收益情况。在数据仓库领域，自底向上方法是快速实现数据集市、部门级数据仓库的有效手段。自底向上方法的优点和缺点如表 3-12 所示。

表 3-12　自底向上方法的优缺点

优　点	缺　点
实现的需求和开始时的需要远远超过自顶向下分析和长期考虑的范围	最初方案实现之后，最好回顾一下方案是如何服务于整个企业的
在企业对数据仓库了解的早期，该方法使企业无须巨大投入就可见到效益	单个自底向上工程项目的失败可能推迟潜在技术的实现
少数人集中工作在一个部门范围，可以加速实现决策过程	早期的小组应不断发展为较大的小组，以扩充最初方案的覆盖范围

规划和实现数据仓库的自底向上方法一般用于以下情况：

（1）企业还没有确实掌握数据仓库技术，希望进行技术评估来决定运行该技术的方式、地点和时间。

（2）企业希望了解实现和运行数据仓库所需要的各种费用情况。

（3）企业在对数据仓库进行投资选择。自底向上方法对于希望从数据仓库投资中快速得到回报的用户是非常有效的。该方法
可以使企业充分利用各种技术，无须冒很大风险。

3．联合方法

在以上两种方法的联合方法中，企业在保持自底向上方法的快速实现和机遇应用的同时，还可以利用自顶向下方法的规划和决策性质。这种方法依赖于以下两个因素：

（1）自顶向下的结构、标准和设计小组，可以从一个项目向另外一个项目传递知识，也可以把战术决策变为战略决策。

（2）自底向上方法的项目小组，它直接负责在短期内实现一个集中的、部门级的商务解决方案。

联合方法具有以上两种方法的优点，但是难以作为一个项目来管理。该方法一般用于：

（1）实现企业拥有经验丰富的设计师，有能力建立、证明、应用和维护数据结构、技术结构及企业模型，可以很容易地从具体（运作系统中的元数据）转移到抽象。

（2）企业拥有固定的项目小组，完全清楚数据仓库技术应用的场所。他们可以清楚地看到当前的商务需求。

联合方法适合数据仓库技术的快速试运行，并且保留了建立长远的决策方案的机会。

3.8 数据挖掘

随着数据库技术的迅速发展及数据库管理系统的广泛应用，人们积累的数据越来越多。激增的数据背后隐藏着许多重要的信息，人们希望能够对其进行更高层次的分析，以便更好地利用这些数据。目前的数据库系统可以高效地实现数据的录入、查询、统计等功能，但无法发现数据中存在的关系和规则，无法根据现有的数据预测未来的发展趋势。缺乏挖掘数据背后隐藏的知识的手段，导致了“数据爆炸但知识贫乏”的现象。

3.8.1 数据挖掘的概念

数据挖掘（Data Mining）技术是人们长期对数据库技术进行研究和开发的结果。起初各种商业数据是存储在计算机的数据库中的，然后发展到可对数据库进行查询和访问，进而发展到对数据库的即时遍历。数据挖掘使数据库技术进入了一个更高级的阶段，它不仅能对过

去的数据进行查询和遍历，并且能够找出过去数据之间的潜在联系，从而促进信息的传递。现在数据挖掘技术在商业应用中已经可以马上投入使用，因为对这种技术进行支持的三种基础技术已经发展成熟，它们是海量数据搜集、强大的多处理器计算机和数据挖掘算法。

从技术角度来看，数据挖掘就是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中，提取隐含在其中的、人们事先不知道的、但又是潜在有用的信息和知识的过程。这个定义包括好几层含义：数据源必须是真实的、大量的、含噪声的；发现的是用户感兴趣的知识；发现的知识要可接受、可理解、可运用；并不要求发现放之四海而皆准的知识，仅支持特定的发现问题。

还有很多和这一术语相近的术语，如从数据库中发现知识、数据分析、数据融合（Data Fusion），以及决策支持等。

何为知识？从广义上理解，数据、信息也是知识的表现形式，但是人们更把概念、规则、模式、规律和约束等看做知识。原始数据可以是结构化的，如关系数据库中的数据；也可以是半结构化的，如文本、图形和图像数据；甚至是分布在网络上的异构型数据。发现知识的方法可以是数学的，也可以是非数学的；可以是演绎的，也可以是归纳的。发现的知识可以被用于信息管理，查询优化，决策支持和过程控制等，还可以用于数据自身的维护。因此，数据挖掘是一门交叉学科，它把人们对数据的应用从低层次的简单查询，提升到从数据中挖掘知识，提供决策支持。在这种需求牵引下，汇聚了不同领域的研究者，尤其是数据库技术、人工智能技术、数理统计、可视化技术、并行计算等方面的学者和工程技术人员，投身到数据挖掘这一新兴的研究领域，形成新的技术热点。

从商业角度来看，数据挖掘是一种新的商业信息处理技术，其主要特点是对商业数据库中的大量业务数据进行抽取、转换、分析和其他模型化处理，从中提取辅助商业决策的关键性数据。

简而言之，数据挖掘其实是一种深层次的数据分析方法。数据分析本身已经有很多年的历史，只不过在过去数据收集和分析的目的是用于科学研究，另外，由于当时计算能力的限制，对大量数据进行分析的复杂数据分析方法受到很大限制。现在，由于各行业业务自动化的实现，商业领域产生了大量的业务数据，这些数据不再是为了分析的目的而收集，而是由于纯机会的商业运作而产生。分析这些数据也不再是单纯为了研究的需要，更主要是为商业决策提供真正有价值的信息，进而获得利润。但所有企业面临的一个共同问题是：企业数据量非常大，而其中真正有价值的信息却很少，因此从大量的数据中通过深层分析，获得有利于商业运作、提高竞争力的信息，就像从矿石中淘金一样，数据挖掘也因此而得名。

因此，数据挖掘可以描述为：按企业既定业务目标，对大量的企业数据进行探索和分析，揭示隐藏的、未知的或验证已知的规律性，并进一步将其模型化的先进有效的方法。

数据挖掘与传统的数据分析（如查询、报表、联机应用分析）的本质区别是数据挖掘是在没有明确假设的前提下去挖掘信息、发现知识。数据挖掘所得到的信息应具有先知，有效和可实用三个特征。

先前未知的信息是指该信息是预先未曾预料到的，即数据挖掘是要发现那些不能靠直觉发现的信息或知识，甚至是违背直觉的信息或知识，挖掘出的信息越是出乎意料，就可能越有价值。在商业应用中最典型的例子就是一家连锁店通过数据挖掘发现了小孩纸尿布和啤酒之间有着惊人的联系。

特别要指出的是，数据挖掘技术从一开始就是面向应用的。它不仅是面向特定数据库的简单检索查询调用，而且要对这些数据进行微观、中观乃至宏观的统计、分析、综合和推理，以指导实际问题的求解，企图发现事件间的相互关联，甚至利用已有的数据对未来的活动进行预测。例如，加拿大 BC 省电话公司要求加拿大 Simon Fraser 大学知识发现研究组，根据其拥有十多年的客户数据，总结、分析并提出新的电话收费和管理办法，制定既有利于公司又有利于客户的优惠政策。这样一来，就把人们对数据的应用，从低层次的末端查询操作，提高到为各级经营决策者提供决策支持。这种需求驱动力比数据库查询更为强大。

3.8.2 数据挖掘的功能

数据挖掘通过预测未来趋势及行为，做出前摄的、基于知识的决策。数据挖掘的目标是从数据库中发现隐含的、有意义的知识，主要有以下五类功能。

1．自动预测趋势和行为数据挖掘自动在大型数据库中寻找预测性信息，以往需要进行大量手工分析的问题如今可以迅速直接由数据本身得出结论。一个典型的例子是市场预测问题，数据挖掘使用过去有关促销的数据来寻找未来投资中回报最大的用户，其他可预测的问题包括预报破产及认定对指定事件最可能做出反应的群体。

2．关联分析数据关联是数据库中存在的一类重要的可被发现的知识。若两个或多个变量的取值之间存在某种规律性，就称为关联。关联可分为简单关联、时序关联、因果关联。关联分析的目的是找出数据库中隐藏的关联网。有时并不知道数据库中数据的关联函数，即使知道也是不确定的，因此关联分析生成的规则带有可信度。

3．聚类数据库中的记录可被划分为一系列有意义的子集，即聚类。聚类增强了人们对

客观现实的认识，是概念描述和偏差分析的先决条件。聚类技术主要包括传统的模式识别方法和数学分类学。20 世纪 80 年代初，Mchalski 提出了概念聚类技术及其要点，即在划分对象时不仅要考虑对象之间的距离，还要求划分出的类具有某种内涵描述，从而避免了传统技术的某些片面性。

4．概念描述概念描述就是对某类对象的内涵进行描述，并概括这类对象的有关特征。概念描述分为特征性描述和区别性描述，前者描述某类对象的共同特征，后者描述不同类对象之间的区别。生成一个类的特征性描述只涉及该类对象中所有对象的共性。生成区别性描述的方法很多，如决策树方法、遗传算法等。

5．偏差检测数据库中的数据常有一些异常记录，从数据库中检测这些偏差很有意义。偏差包括很多潜在的知识，如分类中的反常实例、不满足规则的特例、观测结果与模型预测值的偏差、量值随时间的变化等。偏差检测的基本方法是，寻找观测结果与参照值之间有意义的差别。

3.8.3 数据挖掘常用技术

常用的数据挖掘技术包括关联分析、序列分析、分类、预测、聚类分析及时间序列分析等。

1．关联分析

关联分析主要用于发现不同事件之间的关联性，即一个事件发生的同时，另一个事件也经常发生。关联分析的重点在于快速发现那些有实用价值的关联发生的事件。其主要依据是事件发生的概率和条件概率应该符合一定的统计意义。

对于结构化的数据，以客户的购买习惯数据为例，利用关联分析，可以发现客户的关联购买需要。例如，一个开设储蓄账户的客户很可能同时进行债券交易和股票交易，购买纸尿裤的男顾客经常同时购买啤酒等。利用这种知识可以采取积极的营销策略，扩展客户购买的产品范围，吸引更多的客户。通过调整商品的布局便于顾客买到经常同时购买的商品，或者通过降低一种商品的价格来促进另一种商品的销售等。

对于非结构化的数据，以空间数据为例，利用关联分析，可以发现地理位置的关联性。例如，85%的靠近高速公路的大城镇与水相邻，或者发现通常与高尔夫球场相邻的对象等。

2．序列分析

序列分析技术主要用于发现一定时间间隔内接连发生的事件。这些事件构成一个序列，

发现的序列应该具有普遍意义，其依据除了统计上的概率之外，还要加上时间的约束。

3．分类分析

分类分析通过分析具有类别的样本的特点，得到决定样本属于各种类别的规则或方法。利用这些规则和方法对未知类别的样本分类时应该具有一定的准确度。其主要方法有基于统计学的贝叶斯方法、神经网络方法、决策树方法及支持向量机（support vector machines）等。

利用分类技术，可以根据顾客的消费水平和基本特征对顾客进行分类，找出对商家有较大利益贡献的重要客户的特征，通过对其进行个性化服务，提高他们的忠诚度。

利用分类技术，可以将大量的半结构化的文本数据，如 WEB 页面、电子邮件等进行分类。可以将图片进行分类，例如，根据已有图片的特点和类别，可以判定一幅图片属于何种类型的规则。对于空间数据，也可以进行分类分析，例如，可以根据房屋的地理位置决定房屋的档次。

4．聚类分析

聚类分析是根据物以类聚的原理，将本身没有类别的样本聚集成不同的组，并且对每一个这样的组进行描述的过程。其主要依据是聚到同一个组中的样本应该彼此相似，而属于不同组的样本应该足够不相似。

仍以客户关系管理为例，利用聚类技术，根据客户的个人特征及消费数据，可以将客户群体进行细分。例如，可以得到这样的一个消费群体：女性占 91%，全部无子女、年龄在 31 岁到 40 岁占 70%，高消费级别的占 64%，买过针织品的占 91%，买过厨房用品的占 89%，买过园艺用品的占 79%。针对不同的客户群，可以实施不同的营销和服务方式，从而提高客户的满意度。

对于空间数据，根据地理位置及障碍物的存在情况可以自动进行区域划分。例如，根据分布在不同地理位置的 ATM 机的情况将居民进行区域划分，根据这一信息，可以有效地进行 ATM 机的设置规划，避免浪费，同时也避免失掉每一个商机。

对于文本数据，利用聚类技术可以根据文档的内容自动划分类别，从而便于文本的检索。

5．预测

预测与分类类似，但预测是根据样本的已知特征估算某个连续类型的变量的取值的过程，而分类则只是用于判别样本所属的离散类别而已。预测常用的技术是回归分析。

6．时间序列

分析时间序列分析的是随时间而变化的事件序列，目的是预测未来发展趋势，或者寻找

相似发展模式或者是发现周期性发展规律。

3.8.4 数据挖掘的流程

数据挖掘是指一个完整的过程，该过程从大型数据库中挖掘先前未知的，有效的，可实用的信息，并使用这些信息做出决策或丰富知识。

数据挖掘环境示意图如图 3-13 所示。

图 3-13　数据挖掘环境框图

数据挖掘的流程大致如下：

1．问题定义在开始数据挖掘之前，最先的也是最重要的要求就是熟悉背景知识，弄清用户的需求。缺少了背景知识，就不能明确定义要解决的问题，就不能为挖掘准备优质的数据，也很难正确地解释得到的结果。要想充分发挥数据挖掘的价值，必须对目标有一个清晰明确的定义，即决定到底想干什么。

2．建立数据挖掘库

要进行数据挖掘必须收集要挖掘的数据资源。一般建议把要挖掘的数据都收集到一个数据库中，而不是采用原有的数据库或数据仓库。这是因为大部分情况下需要修改要挖掘的数据，而且还会遇到采用外部数据的情况；另外，数据挖掘还要对数据进行各种纷繁复杂的统计分析，而数据仓库可能不支持这些数据结构。

3．分析数据

分析数据就是通常所进行的对数据深入调查的过程。从数据集中找出规律和趋势，用聚类分析区分类别，最终要达到的目的就是搞清楚多因素相互影响的、十分复杂的关系，发现因素之间的相关性。

4．调整数据

通过上述步骤的操作，对数据的状态和趋势有了进一步的了解，这时要尽可能对问题解决的要求能进一步明确化、进一步量化。针对问题的需求对数据进行增删，按照对整个数据挖掘过程的新认识组合或生成一个新的变量，以体现对状态的有效描述。

5．模型化

在问题进一步明确，数据结构和内容进一步调整的基础上，就可以建立形成知识的模型。这一步是数据挖掘的核心环节，一般运用神经网络、决策树、数理统计、时间序列分析等方法来建立模型。

6．评价和解释

上面得到的模式模型，有可能是没有实际意义或没有实用价值的，也有可能是其不能准确反映数据的真实意义，甚至在某些情况下是与事实相反的，因此需要评估，确定哪些是有效的、有用的模式。评估的一种办法是直接使用原先建立的挖掘数据库中的数据来进行检验，另一种办法是另找一批数据并对其进行检验，再一种办法是在实际运行的环境中取出新鲜数据进行检验。

数据挖掘过程的分步实现，不同的步骤需要不同专长的人员，他们大体可以分为三类。

（1）业务分析人员。要求精通业务，能够解释业务对象，并根据各业务对象确定出用于数据定义和挖掘算法的业务需求。

（2）数据分析人员。精通数据分析技术，并较熟练地掌握统计学，有能力把业务需求转化为数据挖掘的各步操作，并为每步操作选择合适的技术。

（3）数据管理人员。精通数据管理技术，并从数据库或数据仓库中收集数据。

由上可见，数据挖掘是一个多种专家合作的过程，也是一个在资金上和技术上高投入的过程。这一过程要反复进行，在反复过程中，不断地趋近事物的本质，不断地优选问题的解决方案。

3.9 NoSQL

NoSQL 即 Not Only SQL，可直译“不仅仅是 SQL”，这项技术正在掀起一场全新的数据库革命性运动。

在本章 3.2.2 节曾提到数据的模式包括多种类型，如层次模型、网状模型、关系模型等，而在实际应用过程中，几乎都是在用关系模型，主流的数据库系统都是关系型的。但随着互联网 web2.0 网站的兴起，传统的关系数据库在应付 web2.0 网站，特别是超大规模和高并发的 SNS 类型的 web2.0 纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。这也就使得 NoSQL 技术进入了人们的视野。

NoSQL 的出现打破了长久以来关系型数据库与 ACID 理论大一统的局面。NoSQL 数据

存储不需要固定的表结构，通常也不存在连接操作。在大数据存取上具备关系型数据库无法比拟的性能优势。

关系型数据库中的表都是存储一些格式化的数据结构，每个元组字段的组成都一样，即使不是每个元组都需要所有的字段，但数据库会为每个元组分配所有的字段，这样的结构可以便于表与表之间进行连接等操作，但从另一个角度来说它也是关系型数据库性能瓶颈的一个因素。而非关系型数据库以键值对存储，它的结构不固定，每一个元组可以有不一样的字段，每个元组可以根据需要增加一些自己的键值对，这样就不会局限于固定的结构，可以减少一些时间和空间的开销。

与关系型数据库相比，NoSQL 数据库具有以下几个优点：

1．易扩展

NoSQL 数据库种类繁多，但是一个共同的特点都是去掉关系数据库的关系型特性。数据之间无关系，这样就非常容易扩展。无形之间，在架构的层面上带来了可扩展的能力。

2．大数据量，高性能

NoSQL 数据库都具有非常高的读写性能，尤其在大数据量下，同样表现优秀。这得益于它的无关系性，数据库的结构简单。一般 MySQL 使用 Query Cache，每次表一更新 Cache 就失效，它是一种大粒度的 Cache，在针对 web2.0 的交互频繁的应用，Cache 性能不高。而 NoSQL 的 Cache 是记录级的，是一种细粒度的 Cache，所以 NoSQL 在这个层面上来说性能就高很多了。

3．灵活的数据模型

NoSQL 无须事先为要存储的数据建立字段，随时可以存储自定义的数据格式。而在关系数据库里，增删字段是一件非常麻烦的事情。如果是非常大数据量的表，增加字段简直就是一个噩梦。这点在大数据量的 web2.0 时代尤其明显。

4．高可用

NoSQL 在不太影响性能的情况，就可以方便地实现高可用的架构。比如 Cassandra，HBase 模型，通过复制模型也能实现高可用。

当然，NoSQL 也存在很多缺点，例如，并未形成一定标准，各种产品层出不穷，内部混乱，各种项目还需时间来检验，缺乏相关专家技术的支持等。

3.10 大数据

大数据（big data），指无法在一定时间范围内用常规软件工具进行捕捉、管理和处理的数据集合，是需要新处理模式才能具有更强的决策力、洞察发现力和流程优化能力的海量、高增长率和多样化的信息资产。

1. 大数据的特点

业界通常用 4 个 V(即 Volume、Variety、Value、Velocity)来概括大数据的特征。

Volume：指的是数据体量巨大，从 TB 级别跃升到 PB 级别（1PB=1024TB）、EB 级别（1EB=1024PB），甚至于达到 ZB 级别（1ZB=1024EB）。截至目前，人类生产的所有印刷材料的数据量是 200PB，而历史上全人类说过的所有话的数据量大约是 5EB。当前，典型个人计算机硬盘的容量为 TB 量级，而一些大企业的数据量已经接近 EB 量级。

例如，在交通领域，某市交通智能化分析平台数据来自路网摄像头/传感器、公交、轨道交通、出租车以及省际客运、旅游、化危运输、停车、租车等运输行业，还有问卷调查和地理信息系统数据。4 万辆车每天产生 2000 万条记录，交通卡刷卡记录每天 1900 万条，手机定位数据每天 1800 万条，出租车运营数据每天 100 万条，电子停车收费系统数据每天 50 万条，定期调查覆盖 8 万户家庭等，这些数据在体量上就达到了大数据的规模。

Variety：指的是数据类型繁多。这种类型的多样性也让数据被分为结构化数据和非结构化数据。相对于以往便于存储的以文本为主的结构化数据，非结构化数据越来越多，包括网络日志、音频、视频、图片、地理位置信息等，这些多类型的数据对数据的处理能力提出了更高要求。

Value：指的是价值密度低。价值密度的高低与数据总量的大小成反比。以视频为例，一部 1 小时的视频，在连续不间断的监控中，有用数据可能仅有 1-2 秒。如何通过强大的机器算法更迅速地完成数据的价值“提纯”成为目前大数据背景下亟待解决的难题。当然把数据集成在一起，并完成“提纯”是能达到 1+1 大于 2 的效果的，这也正是大数据技术的核心价值之一。

Velocity：指的是处理速度快。这是大数据区分于传统数据挖掘的最显著特征。根据 IDC 的“数字宇宙”的报告，预计到 2020 年，全球数据使用量将达到 35.2ZB。在如此海量的数据面前，处理数据的效率就是企业的生命。

2．传统数据与大数据的比较

传统数据与大数据的差异如表 3-13 所示。

表 3-13　传统数据与大数据的比较

比较维度	传统数据	大数据
数据量	GB 或 TB 级	PB 级或以上
结构化程度	结构化或半结构化数据	所有类型的数据
数据分析需求	现有数据的分析与检测	深度分析（关联分析、回归分析）
硬件平台	高端服务器	集群平台

3．大数据处理关键技术

大数据处理关键技术一般包括：大数据采集、大数据预处理、大数据存储及管理、大数据分析及挖掘、大数据展现和应用（大数据检索、大数据可视化、大数据应用、大数据安全等）。

4．大数据应用

大数据可以在各行各业得以应用，如金融服务、医疗保健、零售业、制造业、政府机构等。

第 4 章 计算机网络

从古代的驿站、八百里快马，到近代的电报、电话，人类对于通信的追求从未间断，信息的处理与通信技术的革新一直伴随社会的发展。而作为 20 世纪人类最伟大、最卓越的发明——个人计算机的出现与发展，使得人们获得了以前无法想象的信息处理能力，为了将这些强大的信息处理设备连接起来，避免出现信息孤岛现象，就催生了“计算机网络”，这一新时代的通信技术。计算机网络使得其功能得到了大大的加强，范围得到了很大的扩展。

从严谨的定义角度说，计算机网络是指由通信线路互相连接的许多独立自主工作的计算机构成的资源共享集合体，它是计算机技术和通信技术相结合的产物。它的出现推动了信息化的发展，从局域网到互联网，都对信息应用产生了深远的影响。因此这就要求系统架构设计师在设计应用系统时必须了解网络的基础知识与特性，掌握一些相关的基础应用知识。

4.1 网络架构与协议

网络架构是指计算机网络的各层及其协议的集合。计算机之间要交换数据，就必须遵守一些事先约定好的规则，用于规定信息的格式及如何发送和接收信息的一套规则就称为网络

协议。为了减少网络协议设计的复杂性，网络设计者并不是设计一个单一、巨大的协议来为所有形式的通信规定完整的细节，而是将庞大而复杂的通信问题转化为若干个小问题，然后为每个小问题设计一个单独的协议。

计算机网络采用分层设计方法，按照信息的传输过程将网络的整体功能分解为一个个的功能层，不同机器上的同等功能层之间采用相同的协议，同一机器上的相邻功能层之间通过接口进行信息传递。

4.1.1 网络互联模型

1977 年，国际标准化组织为适应网络标准化发展的需求，制定了开放系统互联参考模型（Open System Interconnection/Reference Model，OSI/RM），从而形成了网络架构的国际标准。OSI/RM 构造了由下到上的七层模型，分别是物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。

1．OSI/RM 各层的功能

在数据传输过程中，每一层都承担不同的功能和任务，以实现对数据传输过程中的各个阶段的控制。

（1）物理层。物理层的主要功能是透明地完成相邻节点之间原始比特流的传输。其中“透明”的意思是指物理层并不需要关心比特代表的具体含义，而要考虑的是如何发送“0”和“1”，以及接收端如何识别。物理层在传输介质基础上作为系统和通信介质的接口，为数据链路层提供服务。

（2）数据链路层。数据链路层负责在两个相邻节点之间的线路上无差错地传送以帧为单位的数据，通过流量控制和差错控制，将原始不可靠的物理层连接变成无差错的数据通道，并解决多用户竞争问题，使之对网络层显现一条可靠的链路。

（3）网络层。网络层是通信子网的最高层，其主要任务是在数据链路层服务的基础上，实现整个通信子网内的连接，并通过网络连接交换网络服务数据单元（packet）。它主要解决数据传输单元分组在通信子网中的路由选择、拥塞控制和多个网络互联的问题。网络层建立网络连接为传输层提供服务。

（4）传输层。传输层既是负责数据通信的最高层，又是面向网络通信的低三层（物理层、数据链路层和网络层）和面向信息处理的高三层（会话层、表示层和应用层）之间的中间层，是资源子网和通信子网的桥梁，其主要任务是为两台计算机的通信提供可靠的端到端

的数据传输服务。传输层反映并扩展了网络层子系统的服务功能，并通过传输层地址为高层提供传输数据的通信端口，使系统之间高层资源的共享不必考虑数据通信方面的问题。

（5）会话层。会话层利用传输层提供的端到端数据传输服务，具体实施服务请求者与服务提供者之间的通信、组织和同步它们的会话 活动，并管理它们的数据交换过程。会话层提供服务通常需要经过建立连接、数据传输和释放连接三个阶段。会话层是最薄的一层，常被省略。

（6）表示层。表示层处理的是用户信息的表示问题。端用户（应用进程）之间传送的数据包含语义和语法两个方面。语义是数据的内容及其含义，它由应用层负责处理；语法是与数据表示形式有关的方面，例如，数据的格式、编码和压缩等。表示层主要用于处理应用实体面向交换的信息的表示方法，包括用户数据的结构和在传输时的比特流（或字节流）的表示。这样，即使每个应用系统有各自的信息表示法，但被交换的信息类型和数值仍能用一种共同的方法来描述。

（7）应用层。应用层是直接面向用户的一层，是计算机网络与最终用户之间的界面。在实际应用中，通常把会话层和表示层归入到应用层，使 OSI/RM 成为一个简化的五层模型。

2．TCP/IP 结构模型

虽然 OSI/RM 已成为计算机网络架构的标准模型，但因为 OSI/RM 的结构过于复杂，实际系统中采用 OSI/RM 的并不多。目前，使用最广泛的可互操作的网络架构是 TCP/IP（Transmission Control Protocol/ Internet Protocol，传输控制协议/网际协议）结构模型。与 OSI/RM 结构不同，不存在一个正式的 TCP/IP 结构模型，但可根据已开发的协议标准和通信任务将其大致分成四个比较独立的层次，分别是网络接口层、网络互联层、传输层和应用层。

（1）网络接口层。网络接口层大致对应于 OSI/RM 的数据链路层和物理层，TCP/IP 协议不包含具体的物理层和数据链路层，只定义了网络接口层作为物理层的接口规范。网络接口层处在 TCP/IP 结构模型的最底层，主要负责管理为物理网络准备数据所需的全部服务程序和功能。

（2）网络互联层。网络互联层也称为网络层、互联网层或网际层，负责将数据报独立地从信源传送到信宿，主要解决路由选择、阻塞控制和网络互联等问题，在功能上类似于 OSI/RM 中的网络层。

（3）传输层。传输层负责在信源和信宿之间提供端到端的数据传输服务，相当于 OSI/RM 中的传输层。

（4）应用层。应用层直接面向用户应用，为用户方便地提供对各种网络资源的访问服务，包含了 OSI/RM 会话层和表示层中的部分功能。

4.1.2 常见的网络协议

计算机网络的各层中存在着许多协议，它们是定义通过网络进行通信的规则。接收方与发送方同层的协议必须一致，否则，一方将无法识别另一方发出的信息。

1．应用层协议

在应用层中，定义了很多面向应用的协议，应用程序通过本层协议利用网络完成数据交互的任务。这些协议主要有 FTP、TFTP、HTTP、SMTP、DHCP、Telnet、DNS 和 SNMP 等。

FTP（File TransportProtocol，文件传输协议）是网络上两台计算机传送文件的协议，运行在 TCP 之上，是通过 Internet 将文件从一台计算机传输到另一台计算机的一种途径。FTP 的传输模式包括 Bin（二进制）和 ASCII（文本文件）两种，除了文本文件之外，都应该使用二进制模式传输。FTP 在客户机和服务器之间需建立两条 TCP 连接，一条用于传送控制信息（使用 21 号端口），另一条用于传送文件内容（使用 20 号端口）。

TFTP（Trivial FileTransfer Protocol，简单文件传输协议）是用来在客户机与服务器之间进行简单文件传输的协议，提供不复杂、开销不大的文件传输服务。TFTP 建立在 UDP（User Datagram Protocol，用户数据报协议）之上，提供不可靠的数据流传输服务，不提供存取授权与认证机制，使用超时重传方式来保证数据的到达。

HTTP（Hypertext TransferProtocol，超文本传输协议）是用于从 WWW 服务器传输超文本到本地浏览器的传送协议。它可以使浏览器更加高效，使网络传输减少。HTTP 建立在 TCP 之上，它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示等。

SMTP（Simple Mail TransferProtocol，简单邮件传输协议）建立在 TCP 之上，是一种提供可靠且有效的电子邮件传输的协议。SMTP 是建模在 FTP 文件传输服务上的一种邮件服务，主要用于传输系统之间的邮件信息，并提供与电子邮件有关的通知。

DHCP（Dynamic HostConfiguration Protocol，动态主机配置协议）建立在 UDP 之上，是基于客户机/服务器模型设计的。所有的 IP 网络设定数据都由 DHCP 服务器集中管理，并负责处理客户端的 DHCP 要求；而客户端则会使用从服务器分配下来的 IP 环境数据。DHCP 通过租约（默认为 8 天）的概念，有效且动态地分配客户端的 TCP/IP 设定。当租约

过半时，客户机需要向 DHCP 服务器申请续租；当租约超过 87.5%时，如果仍然没有和当初提供 IP 的 DHCP 服务器联系上，则开始联系其他的 DHCP 服务器。DHCP 分配的 IP 地址可以分为三种方式，分别是固定分配、动态分配和自动分配。

Telnet（远程登录协议）是登录和仿真程序，建立在 TCP 之上，它的基本功能是允许用户登录进入远程计算机系统。以前，Telnet 是一个将所有用户输入送到远程计算机进行处理的简单的终端程序。目前，它的一些较新的版本是在本地执行更多的处理，可以提供更好的响应，并且减少了通过链路发送到远程计算机的信息数量。

DNS（Domain NameSystem，域名系统）在 Internet 上域名与 IP 地址之间是一一对应的，域名虽然便于人们记忆，但机器之间只能互相识别 IP 地址，它们之间的转换工作称为域名解析，域名解析需要由专门的域名解析服务器来完成，DNS 就是进行域名解析的服务器。DNS 通过对用户友好的名称查找计算机和服务。当用户在应用程序中输入 DNS 名称时，DNS 服务可以将此名称解析为与之相关的其他信息，例如，IP 地址。

SNMP（Simple NetworkManagement Protocol，简单网络管理协议）是为了解决 Internet 上的路由器管理问题而提出的，它可以在 IP、IPX、AppleTalk 和其他传输协议上使用。SNMP 是指一系列网络管理规范的集合，包括协议本身、数据结构的定义和一些相关概念。目前，SNMP 已成为网络管理领域中事实上的工业标准，并被广泛支持和应用，大多数网络管理系统和平台都是基于 SNMP 的。

2．传输层协议

传输层主要有两个传输协议，分别是 TCP 和 UDP（User Datagram Protocol，用户数据报协议），这些协议负责提供流量控制、错误校验和排序服务。

TCP 是整个 TCP/IP 协议族中最重要的协议之一，它在 IP 协议提供的不可靠数据服务的基础上，采用了重发技术，为应用程序提供了一个可靠的、面向连接的、全双工的数据传输服务。TCP 协议一般用于传输数据量比较少，且对可靠性要求高的场合。

UDP 是一种不可靠的、无连接的协议，可以保证应用程序进程间的通信，与 TCP 相比，UDP 是一种无连接的协议，它的错误检测功能要弱得多。可以这样说，TCP 有助于提供可靠性，而 UDP 则有助于提高传输速率。UDP 协议一般用于传输数据量大，对可靠性要求不是很高，但要求速度快的场合。

3．网络层协议

网络层中的协议主要有 IP、ICMP（Internet Control Message Protocol，网际控制报文协议）、IGMP（Internet Group Management Protocol，网际组管理协议）、ARP（Address Resolution

Protocol，地址解析协议）和 RARP（Reverse Address Resolution Protocol，反向地址解析协议）等，这些协议处理信息的路由和主机地址解析。

IP 所提供的服务通常被认为是无连接的和不可靠的，它将差错检测和流量控制之类的服务授权给了其他的各层协议，这正是 TCP/IP 能够高效率工作的一个重要保证。网络层的功能主要由 IP 来提供，除了提供端到端的分组分发功能外，IP 还提供很多扩充功能。例如，为了克服数据链路层对帧大小的限制，网络层提供了数据分块和重组功能，这使得很大的 IP 数据包能以较小的分组在网络上传输。

ARP 用于动态地完成 IP 地址向物理地址的转换。物理地址通常是指计算机的网卡地址，也称为 MAC（Media Access Control，媒体访问控制）地址，每块网卡都有唯一的地址；RARP 用于动态完成物理地址向 IP 地址的转换。

ICMP 是一个专门用于发送差错报文的协议，由于 IP 协议是一种尽力传送的通信协议，即传送的数据可能丢失、重复、延迟或乱序传递，所以需要一种尽量避免差错并能在发生差错时报告的机制，这就是 ICMP 的功能。

IGMP 允许 Internet 中的计算机参加多播，是计算机用作向相邻多目路由器报告多目组成员的协议。多目路由器是支持组播的路由器，它向本地网络发送 IGMP 查询，计算机通过发送 IGMP 报告来应答查询。多目路由器负责将组播包转发到网络中所有组播成员。

4.1.3 IPv6

互联网络能发展到当前的规模，IPv4 协议的建立功不可没。但同时它的缺点也已经充分显现出来，如地址空间耗尽、路由表急剧膨胀、缺乏对 QoS 的支持、本身并不提供任何安全机制、移动性差等问题。尽管采用了许多新的机制来缓解这些问题，如 DHCP 技术、NAT 技术、CIDR 技术等，但都不可避免地要引入其他新的问题，问题没有得到根本解决。于是 IETF 从 90 年代起就开始积极探讨下一代 IP 网络，经过几年努力，在广泛听取业界和专家意见的基础上，终于在 1995 年 12 月推出了下一代网络的 RFC 文档——IPv6 协议，该协议最早叫做下一代 IP(IP Next Generation，IPng)。现在它的全称是“互联网协议第 6 版”，即下一代的网际协议。

1.IPv6 地址表示

一个 32 位的 IPv4 地址以 8 个位为一段分成 4 段，每段之间用点“.”分开。而 IPv6 地址的 128 位是以 16 位为一段，共分为 8 段，每段的 16 位转换为一个 4 位的 16 进

制数字，每段之间用冒号“：”分开。

如 RFC 2373 所定义，有 3 种格式表示 IPv6 地址。首选格式是最长的表示方法，由所有的 32 个 16 进制字符组成。如，下面这个 128 位的 IPv6 地址用 2 进制表示为：

00100000000000010000110110101000110100000000001000000000000000010000000000000000
0000000000000000000000000000000000001100111011001101

先把这 128 位按照 16 位一段分开：

0010000000000001 0000110110101000 1101000000000001
0000000000000001
0000000000000000 0000000000000000 0000000000000000
1100111011001101

把每 16 位一段转换为 4 个字符表示的 16 进制，然后以冒号隔开，可以得到如下表示形式：

2001：0da8：d001：0001：0000：0000：0000：0001

上面这个地址就是首选格式，是一个适合于计算机“思维”的表示法。

2. IP v6 压缩地址表示

在 IPv6 中，常见到使用包含一长串 0 的地址，为了方便书写，对于每一段中的前导 0 可以进行省略。如前面的首选格式地址经过一次压缩，可以得到：

2001：da8：d001：1：0：0：0：1

对于连续 2 段以上都为 0 的字段，可以使用“::”（两个冒号）来表示，这样再次压缩，变成：

2001：da8：d001：1：：1

这就是 IPv6 地址的压缩表示法。（注意：每个 IPv6 地址只允许有一个“::”）。

3.内嵌 IPv4 地址的 IPv6 地址

还有一种表示法就是在 IPv6 地址中使用内嵌的 IPv4 地址。这种表示法的地址的第一部分使用十六进制表示，而 IPv4 部分采用十进制。这是过渡机制所用的 IPv6 地址特有的表示法。如：fe80：：200：5efe：58.20.27.60，这个 IPv6 地址的后半部分就是一个 IPv4 地址。

4.IPv6 地址类型

IPv4 有单播、广播和组播地址类型，在 IPv6 里面，广播已经不再使用了，这对网络管理员来说，应该是个好消息，因为在传统的 IP 网络中，出现的很多问题都是由于广播引起的。IPv6 仍有 3 种地址类型，分别是单播、多播（也称作组播）、泛播（也称作任意播）。

（1）单播 IPv6 地址：单播地址唯一标识一个 IPv6 节点的接口。发送往单播地址的数据包最终传递给这个地址所标识的接口。为适应负载均衡，IPv6 协议允许多个接口使用相同的 IPv6 地址，只要它们对于主机上的 IPv6 协议表现为一个接口。

（2）多播 IPv6 地址：多播地址标识一组 IPv6 节点的接口。发送往多播地址的数据包会被该多播组所有的成员处理。

（3）泛播 IPv6 地址：泛播地址指派给多个节点的接口。发送往泛播地址的数据包只会传递给其中的一个接口，一般是相隔最近的一个接口。

5.IPv6 的优势

与 IPv4 相比，IPv6 具有以下几点优势：

（1）IPv6 具有更大的地址空间。IPv4 中规定 IP 地址长度为 32 位，而 IPv6 中 IP 地址的长度为 128 位。

（2）IPv6 使用更小的路由表。IPv6 的地址分配一开始就遵循路由汇聚的原则，使路由器能在路由表中用一条记录表示一个子网，大大减小了路由器中路由表的长度，提高了路由器转发数据包的速度。

（3）IPv6 增加了增强的组播支持和对流支持，使网络上的多媒体应用有了长足发展的机会，为服务质量（Quality of Service，QoS）控制提供了良好的网络平台。

（4）IPv6 加入了对自动配置的支持。这是对 DHCP 协议的改进和扩展，使得网络（尤其是局域网）的管理更加方便和快捷。

（5）IPv6 具有更高的安全性。在使用 IPv6 网络时，用户可以对网络层的数据进行加密，并对 IP 报文进行校验，极大地增强了网络的安全性。

6.IPv4 到 IPv6 的过渡技术

IPv4／IPv6 过渡技术有：

（1）双协议栈技术：双栈技术通过节点对 IPv4 和 IPv6 双协议栈的支持，从而支持两种业务的共存。

（2）隧道技术：隧道技术通过在 IPv4 网络中部署隧道，实现在 IPv4 网络上对 IPv6 业务的承载，保证业务的共存和过渡。具体的隧道技术包括：6to4 隧道；6over4 隧道；ISATAP 隧道。

（3）NAT-PT 技术：NAT - PT 使用网关设备连接 IPv6 和 IPv4 网络。当 IPv4 和 IPv6 节点互相访问时，NAT - PT 网关实现两种协议的转换翻译和地址的映射。

4.2 局域网与广域网

局域网（Local Area Network，LAN）是将分散在有限地理范围内的多台计算机通过传输媒体连接起来的通信网络，通过功能完善的网络软件，实现计算机之间的相互通信和资源共享；广域网（Wide Area Network，WAN）是在传输距离较长的前提下所发展的相关技术的集合，用于将大区域范围内的各种计算机设备和通信设备互联在一起，组成一个资源共享的通信网络。

4.2.1 局域网基础知识

当今的计算机网络技术中，局域网已经占据了相当显著的地位。局域网通常具备以下特点：

（1）地理分布范围较小，一般为数百米至数千米的区域范围之内。

（2）数据传输速率高，早期的局域网数据传输速率一般为 10Mbps～100Mbps，目前，1000Mbps 的局域网已经非常普遍，可适用于语音、图像、视频等各种业务数据信息的高速交换。

（3）数据误码率低，这是因为局域网通常采用短距离基带传输，可以使用高质量的传输媒体，从而提高数据传输质量。

（4）一般以 PC 为主体，还包括终端和各种外设，网络中一般不架设主骨干网系统。

（5）协议相对简单、结构灵活，建网成本低、周期短，便于管理和扩充。构成局域网的网络拓扑结构主要有星形结构、总线结构、环形结构和网状结构。

1．星形结构

如图 4-1 所示，星形结构方式的网络在直观上就很容易理解，就像是一张蜘蛛网，中间是一个枢纽（网络交换设备），所有的节点都连接到这个枢纽上，最终组成一个星形的拓扑结构的网络。目前一般办公室的局域网便是该结构。

2．总线结构

如图 4-2 所示，采用总线结构方式的网络，是由一条共享的通信线路将所有节点连接在一起，这条共享的通信线路可以是一根同轴电缆或其他介质。

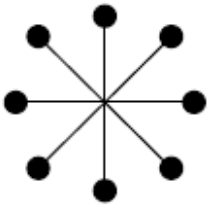


图 4-1　星形结构

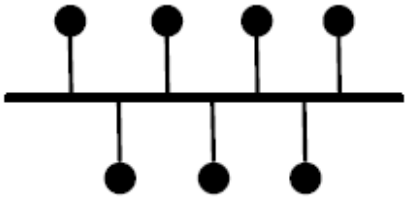


图 4-2　总线结构

3．环形结构

如图 4-3 所示，环形结构方式的网络，与总线结构类似，也是由一条共享的通信线路将所有节点连接在一起。不同的是，环形结构中的共享线路是闭合的，即它将所有的节点排列成一个环，每个节点只与其两个邻居直接相连。若一个节点想要给另一个节点发送信息，消息报文必须经过它们之间的所有节点。

4．网状结构

如图 4-4 所示，网状结构方式的网络就是任何节点彼此之间都会由一根物理通信线路相连，任何节点出现故障都不会影响到其他节点。采用这种拓扑结构方式的网络的布线比较麻烦，而且网络建设的成本也很高，控制方法也很复杂。在实际应用中，一般很少见到这种网络。

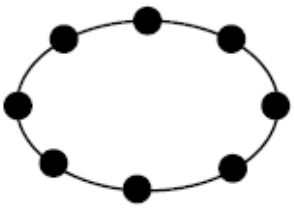


图 4-3　环形结构

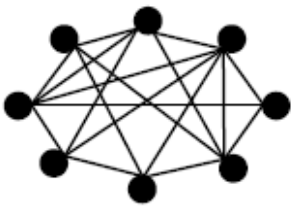


图 4-4　网状结构

4.2.2 无线局域网

无线局域网（Wireless Local Area Networks，WLAN）主要运用射频（Radio Frequency， RF）技术取代原来局域网系统中必不可少的传输介质（例如，同轴电缆、双绞线等）来完成数据的传送任务，有了 WLAN，用户不必因使用有线传输介质而破坏原有的工作环境，可根据需要调整网络节点的位置。同时，便携式计算机更容易接入局域网，这扩大了计算机网络的应用能力和领域。

1．拓扑结构

无线局域网可分为两大类，分别是有接入点模式（基础设施网络）和无接入点模式

（Adhoc 网络）。

（1）基础设施网络。整个网络都使用无线通信的方式，但系统中存在接入点（Access Point，AP），通过接入点将一组节点逻辑上联系在一起，形成一个局域网。AP 的作用与网桥类似，负责在 802.11 和 802.3 的 MAC 协议之间进行转换。一个 AP 覆盖的部分称为一个基本业务域，而 AP 控制的所有节点组成一个基本业务集，由两个以上的基本业务域可以组成一个分布式系统。

（2）Ad hoc 网络。整个网络都使用无线通信的方式，直接通过无线网卡实现点对点连接。与基础设施网络相比，Ad hoc 网络中并没有 AP 这样的设备，可扩展性和灵活性更好，但路由和协调控制等技术都难以解决。

在大多数情况下，无线通信通常是作为有线通信的一种补充和扩展。在这种部署配置下，多个 AP 通过线缆连接在有线网络上，以使无线用户能够访问网络的各部分。

2．IEEE 802.11 标准

IEEE 802 委员会为无线局域网开发了一组标准，即 IEEE 802.11 标准。其中定义了媒体访问控制层（MAC 层）和物理层。物理层定义了工作在 2.4GHz 的 ISM（Industrial Scientific Medical，工业、科学和医学）频段上的扩频通信方式，总数据传输速率设计为 2Mbps。而在 MAC 层采取了载波侦听多路访问/冲突避免协议（Carrier Sense Multiple Access with Collision Avoidance，CSMA/CA），即采用主动避免碰撞而非被动侦测的方式来解决冲突问题。

由于 IEEE 802.11 的业务主要限于数据存取，在速率和传输距离上都不能满足人们的需要，因此，IEEE 在制定更高速度的标准时，就产生了 802.11a 和 802.11b 两个分支，后来又陆续推出了 802.11g、802.11n、802.11ac 的标准，主要是以物理层的不同作为区分，它们的区别直接表现在工作频段和数据传输率、最大传输距离等指标上。而工作在 MAC 层的标准又分为 IEEE 802.11h、IEEE 802.11e 和 IEEE 802.11i 等。802.11h 是 802.11a 的扩展，目的是兼容其他 5GHz 频段的标准（例如，欧盟使用的 HyperLAN2 等）；802.11e 是 IEEE 为满足 QoS 方面的要求而制定的标准；IEEE 802.11i 规定使用 802.1x 认证和密钥管理方式。

3．3G 通信技术

3G 是第三代移动通信及其技术的简称，其主流标准包括：WCDMA、CDMA 2000 和 TD-SCDMA。

WCDMA（Wideband CDMA，宽频 CDMA）的支持者主要是以 GSM（Global System for Mobile Communications，全球移动通信系统）为主的欧洲厂商，日本公司也或多或少参与其中。这套系统能够架设在 GSM 网络上，对于系统提供商而言，可以较轻易地过渡。因此，

WCDMA 具有先天的市场优势。目前，中国联合网络通信集团公司获得基于 WCDMA 技术制式的 3G 业务经营许可。

CDMA 2000 也称为 CDMA Multi-Carrier，以美国高通北美公司为主导提出，摩托罗拉、Lucent 和韩国三星公司都有参与，韩国现在成为该标准的主导者。这套系统是从窄频 CDMA One 数字标准衍生出来的，可以从原有的 CDMA One 结构直接升级到 3G，建设成本低廉。但目前使用 CDMA 的地区只有日本、韩国和北美，所以 CDMA 2000 的支持者不如 WCDMA 多。目前，中国电信集团公司获得基于 CDMA 2000 技术制式的 3G 业务经营许可。

TD-SCDMA 标准是由中国大唐电信制定的 3G 标准，该标准将智能天线、同步 CDMA 和软件无线电等技术融于其中，在频谱利用率、对业务支持具有灵活性、频率灵活性及成本等方面具有独特优势。另外，由于中国庞大的市场，该标准受到各大主要电信设备厂商的重视，全球一半以上的设备厂商都宣布可以支持 TD-SCDMA 标准。目前，中国移动通信集团公司获得基于 TD-SCDMA 技术制式的 3G 业务经营许可。

4．4G 通信技术

4G 是第四代移动通信及其技术的简称，是集 3G 与 WLAN 于一体并能够传输高质量视频图像且图像传输质量与高清晰度电视不相上下的技术产品。4G 系统能够以 100Mbps 的速度下载，比拨号上网快 2000 倍，上传的速度也能达到 20Mbps，并能够满足几乎所有用户对于无线服务的要求。此外，4G 可以在 DSL 和有线电视调制解调器没有覆盖的地方部署，然后再扩展到整个地区。很明显，4G 有着不可比拟的优越性。

4G 标准主要有两大方向，即 LTE（Long Term Evolution）与 WiMAX（无线城域网技术中已介绍），而 LTE 又可进一步分为 TD-LTE 与 FDD-LTE。

（1）TD-LTE（Time Division Long Term Evolution）

TD-LTE 即分时长期演进，是由阿尔卡特-朗讯、诺基亚西门子通信、大唐电信、华为技术、中兴通信、中国移动等业者所共同开发的第四代（4G）移动通信技术与标准。TDD 即时分双工（Time Division Duplexing），是移动通信技术使用的双工技术之一，与 FDD 频分双工相对应。TD-LTE 与 TD-SCDMA 实际上没有关系，TD-LTE 是 TDD 版本的 LTE 的技术，FDD-LTE 的技术是 FDD 版本的 LTE 技术。TD-SCDMA 是 CDMA（码分多址）技术，TD-LTE 是 OFDM（正交频分复用）技术。两者从编解码、帧格式、空口、信令，到网络架构，都不一样。

（2）FDD-LTE（Frequency Division Duplexing Long Term Evolution）

FDD-LTE 即频分双工长期演进，目前该标准的产业发展领先于 TD-LTE。FDD-LTE 已成为当前世界上采用的国家及地区最广泛的终端种类最丰富的一种 4G 标准。其演进路线与速率为： GSM(9K) --> GPRS(42K) --> EDGE(172K) --> WCDMA(364K) -->HSDPA/HSUPA(14.4M) --> HSDPA+/HSUPA+(42M) --> FDD-LTE(300M)。

（3）WiMAX

WiMAX：Worldwide Interoperabilityfor Microwave Access，即全球微波互联接入， WiMAX 的另一个名字是 IEEE 802.16。WiMAX 技术既被纳入到了 3G 技术，也被纳入到了 4G 技术。

802.16 工作的频段采用的是无需授权频段，范围在 2GHz 至 66GHz 之间，而 802.16a 则是一种采用 2G 至 11GHz 无需授权频段的宽带无线接入系统，其频道带宽可根据需求在 1.5M 至 20MHz 范围进行调整，具有更好高速移动下无缝切换的 IEEE 802.16m 的技术正在研发。因此，802.16 所使用的频谱可能比其他任何无线技术更丰富，WiMAX 具有以下优点：

<1>对于已知的干扰，窄的信道带宽有利于避开干扰，而且有利于节省频谱资源。

<2>灵活的带宽调整能力，有利于运营商或用户协调频谱资源。

<3>WiMAX 所能实现的 50 公里的无线信号传输距离是无线局域网所不能比拟的，网络覆盖面积是 3G 发射塔的 10 倍，只要少数基站建设就能实现全城覆盖，能够使无线网络的覆盖面积大大提升。

不过 WiMAX 网络在网络覆盖面积和网络的带宽上优势巨大，但是其移动性却有着先天的缺陷，无法满足高速(≧50km/h)下的网络的无缝链接。

4.2.3 广域网技术

广域网主要提供面向通信的服务，支持用户使用计算机进行远距离的信息交换，与局域网相比，其覆盖范围广、通信的距离远、需要考虑的因素增多，例如，线路的冗余、带宽的利用和差错处理等。广域网一般由电信部门负责组建、管理和维护，并向全社会提供面向通信的有偿服务、流量统计和计费问题。由于在架构师考试中考广域网的概率极低，所以在此不详细叙述。

4.2.4 网络接入技术

目前，接入 Internet 的主要方式有 PSTN、ISDN、ADSL、FTTx+LAN 和 HFC 接入等五种。

1．PSTN 接入

PSTN（Public SwitchingTelephone Network，公用交换电话网络）是指利用电话线拨号接入 Internet，通常计算机需要安装一个 Modem（调制解调器），将电话线插入到 Modem 上，在计算机上利用拨号程序输入接入号码进行接入。PSTN 的速度较低，一般低于 64Kbps。

2．ISDN 接入

ISDN（IntegratedServices Digital Network，综合业务数字网）俗称“一线通”，是在电话网络的基础上构造的纯数字方式的综合业务数字网，能为用户提供包括语音、数据、图像和传真等在内的各类综合业务。ISDN 的基本速率接口为 2B+D 信道，共 144Kbps 带宽，一般使用 RJ45 接口。

3．ADSL 接入

ADSL（AsymmetricalDigital Subscriber Loop，非对称数字用户线路）的服务端设备和用户端设备之间通过普通的电话线连接，无须对入户线缆进行改造，就可以为现有的大量电话用户提供 ADSL 宽带接入。随着标准和技术的成熟及成本的不断降低，ADSL 日益受到电信运营商和用户的欢迎，成为接入 Internet 的主要方式之一。ADSL 的特点是上行速度和下行速度不一样，并且往往是下行速度大于上行速度。目前，比较成熟的 ADSL 标准主要有两种，分别是 G.DMT 和 G.Lite。G.DMT 是全速率的 ADSL 标准，提供 8Mbps 的下行速率和 1.5Mbps 的上行速率，但要求用户安装分离器，而 G.Lite 是一种速率较慢的 ADSL，它不需要在用户端进行线路的分离。G.Lite 标准的最大下行速率为 1.5Mbps，最大上行速率为 512Kbps。

4．FTTx+LAN 接入

光纤通信是指利用光导纤维（简称为光纤）传输光波信号的一种通信方法，相对于以电为媒介的通信方式而言，光纤通信的主要优点有传输频带宽，通信容量大；传输损耗小；抗电磁干扰能力强；线径细、重量轻；资源丰富等。

（1）FTTx 技术。随着光纤通信技术的平民化，以及高速以太网的发展，现在许多宽带智能小区就是采用以千兆以太网技术为主干，充分利用光纤通信技术完成接入的。实现高速以太网的宽带技术常用的方式是 FTTx+LAN（光纤+局域网），根据光纤深入用户的程度，可以分为五种，分别是 FTTC（Fiber To The Curb，光纤到路边）、FTTZ（Fiber To The Zone，光纤到小区）、FTTB（Fiber To The Building，光纤到楼）、FTTF（Fiber To The Floor，光纤到楼层）和 FTTH（Fiber To The Home，光纤到户）。

（2）无源光纤网络（Passive Optical Network，PON）技术。PON 是实现 FFTB 的关键

技术，在光分支点不需要节点设备，只需安装一个简单的光分支器即可，因此，具有节省光缆资源、带宽资源共享、节省机房投资、设备安全性高、建网速度快和综合建网成本低等优点。目前，PON 主要有 APON（ATM PON）和 EPON（Ethernet PON）两种。APON 选择 ATM 和 PON 作为网络协议和平台，其上下行方向的信息传输都采用 ATM 传输方案，下行速率为 622Mbps 或 155Mbps，上行速率为 155Mbps。光节点到前端的距离可长达 10～ 20km，或者更长。采用无源双星形拓扑结构，使用时分复用和时分多址技术，可以实现信元中继、局域网互联、电路仿真、普通电话业务等；EPON 是以太网技术发展的新趋势，其下行速率为 100Mbps 或者 1000Mbps，上行为 100Mbps。在 EPON 中，传送的是可变长度的数据包，最长可为 65535 个字节，简化了网络结构、提高了网络速度。

5．同轴+光纤接入

同轴光纤技术（Hybrid Fiber-Coaxial，HFC）是将光缆敷设到小区，然后通过光电转换节点，利用有线电视（Community Antenna Television，CATV）的总线式同轴电缆连接到用户，提供综合电信业务的技术。这种方式可以充分利用 CATV 原有的网络，由于具有建网快、造价低等特点，使其逐渐成为最佳的接入方式之一。HFC 是由光纤干线网和同轴分配网通过光节点结合而成，一般光纤干线网采用星形结构，同轴电缆分配网采用树形结构。

HFC 的用户端需要使用一个称为 CableModem（电缆调制解调器）的设备，它不单纯是一个调制解调器，还集成了调谐器、加/解密设备、桥接器、网络接口卡、虚拟专网代理和以太网集线器的功能于一身，它无须拨号，可提供随时在线的永远连接。HFC 采用频分复用技术和 64QAM 调制，其上行速率已达 10Mbps 以上，下行速率更高。

4.3 网络互连与常用设备

网络互连是为了将两个以上具有独立自治能力、同构或异构的计算机网络连接起来，实现数据流通，扩大资源共享的范围，或者容纳更多的用户。网络互连包括局域网与局域网的互连、局域网与广域网的互连、广域网与广域网的互连，这可以扩大资源共享的范围，使更多的资源可以被更多的用户共享。

1．网络互连设备

在网络互连时，各节点一般不能简单地直接相连，而是需要通过一个中间设备来实现。按照 OSI/RM 的分层原则，这个中间设备要实现不同网络之间的协议转换功能，根据它们工作的协议层不同进行分类，网络互连设备有中继器（实现物理层协议转换，在电缆间转换

二进制信号）、网桥（实现物理层和和数据链路层协议转换）、路由器（实现网络层和以下各层协议转换）、网关（提供从最底层到传输层或以上各层的协议转换）和交换机等。在实际应用中，各厂商提供的设备都是多功能组合，向下兼容的。表 4-1 则是对以上设备的一个总结。

表 4-1　网络互连设备

互联设备	工作层次	主要功能
中继器	物理层	对接收信号进行再生和发送，只起到扩展传输距离的作用，对高层协议是透明的，但使用个数有限（例如，在以太网中只能使用 4 个）
网桥	数据链路层	根据帧物理地址进行网络之间的信息转发，可缓解网络通信繁忙度，提高效率。只能够连接相同 MAC 层的网络
路由器	网络层	通过逻辑地址进行网络之间的信息转发，可完成异构网络之间的互联互通，只能连接使用相同网络层协议的子网
网关	高层（第 4～7 层）	最复杂的网络互联设备，用于连接网络层以上执行不同协议的子网
集线器	物理层	多端口中继器
二层交换机	数据链路层	是指传统意义上的交换机，多端口网桥
三层交换机	网络层	带路由功能的二层交换机
多层交换机	高层（第 4～7 层）	带协议转换的交换机

随着无线技术运用的日益广泛，目前，市面上基于无线网络的产品非常多，主要有无线网卡、无线 AP、无线网桥和无线路由器等。

2．交换技术

在计算机网络中，当用户较多而传输的距离较远时，通常不采用两点固定连接的专用线路，而是采用交换技术，使通信传输线路为各个用户公用，以提高传输设备的利用率，降低系统费用。

按照实际的数据传送技术，交换技术又可分为电路交换、报文交换和分组交换，它们的主要特点如下：

（1）电路交换。在数据传送之前必须先设置一条通路。在线路释放之前，该通路将由一对用户独占。

（2）报文交换。报文从源点传送到目的地采用存储转发的方式，在传送报文时，同时只占用一段通道。在交换节点中需要缓冲存储，报文需要排队。因此，报文交换不能满足实时通信的要求。

（3）分组交换。交换方式和报文交换方式类似，但报文被分成分组传送，并规定了最

大的分组长度。在数据报分组交换中，目的地需要重新组装报文；在虚电路分组交换中，在数据传送之前必须通过虚呼叫设置一条虚电路。分组交换技术是在数据网络中使用最广泛的一种交换技术。

根据各自的特点，不同的交换技术适用于不同的场合。例如，对于交互式通信来说，报文交换肯定是不适合的；对于较轻和间歇式负载来说，电路交换是最合适的，因此，可以通过电话拨号线路来实行通信；对于较重或持续的负载来说，使用租用的线路以电路交换方式通信是合适的；对必须交换中等数据到大量的数据时，可用分组交换方法。

3．路由技术

路由器是工作在网络层的重要网络互连设备，构成了基于 TCP/IP 协议的 Internet 的主体脉络，工作在 Internet 上的路由器也称为 IP 网关。

路由器的主要功能就是进行路由选择。当一个网络中的计算机要给另一个网络中的计算机发送分组时，它首先将分组送给同一个网络中用于网络之间连接的路由器，路由器根据目的地址信息，选择合适的路由，将该分组传递到目的网络用于网络之间连接的路由器中，然后通过目的网络中内部使用的路由选择协议，该分组最后被递交给目的计算机。

根据路由选择协议的应用范围，可以将其分为内部网关协议（ Interior Gateway Protocol，IGP）、外部网关协议（Exterior Gateway Protocol，EGP）和核心网关协议（Gateway Gateway Protocol，GGP）三大类。

（1）内部网关协议。内部网关协议是指在一个自治系统（Autonomous System，AS）内运行的路由选择协议，主要包括 RIP（Routing Information Protocol，路由信息协议）、 OSPF（Open Shortest Path First，开放式最短路径先先）、
IGRP（Interior Gateway Routing Protocol，内部网关路由协议）和 EIGRP（Enhanced IGRP，增强型 IGRP）等。其中 AS 是指同构型的网关连接的互连网络，通常是由一个网络管理中心控制的。

（2）外部网关协议。外部网关协议是指在两个 AS 之间使用的路由选择协议，最新的 EGP 主要有 BGP（Border Gateway Protocol，边界网关协议），其主要功能是控制路由策略。

（3）核心网关协议。Internet 中有个主干网，所有的 AS 都连接到主干网上，主干网中的网关称为核心网关，核心网关之间交换路由信息时使用的是 GGP。

从路由协议使用的算法来看，所有的路由协议可以分为以下三类：

（1）距离向量协议。计算网络中所有链路的矢量和距离，并以此为依据来确定最佳路径。这类协议会定期向相邻的路由器发送全部或部分路由表。

（2）链路状态协议。使用为每个路由器创建的拓扑数据库来创建路由表，通过计算最短路径来形成路由表。这类协议会定期向相邻路由器发送网络链路状态信息。

（3）平衡型协议。结合了距离向量协议和链路状态协议的优点。

4.4 网络工程

网络工程的建设是一个极其复杂的系统工程，是对计算机网络、信息系统建设和项目管理等领域知识的综合利用的过程，系统分析师必须根据用户单位的需求和具体情况，结合当前网络技术的发展和产品化程度，经过充分的需求分析和市场调研，确定网络建设方案，依据方案有计划、分步骤地实施。按照实施过程的先后，网络工程可分为网络规划、网络设计和网络实施三个阶段。

4.4.1 网络规划

网络规划是网络建设过程中非常重要的环节，同时也是一个系统性的过程。网络规划应该以需求为基础，同时考虑技术和工程的可行性。具体来说，网络规划包括网络需求分析、可行性分析和对现有网络的分析与描述。

1.网络需求分析在网络组建之前，首先要进行需求分析，根据用户提出的要求，进行网络的设计，网络建设的成败很大程度取决于网络实施前的规划工作。

需求分析的基本任务是深入调查用户网络建设的背景、必要性、上网的人数和信息量等，然后进行纵向的、更加深入细致的需求分析和调研，在确定地理布局、设备类型、网络服务、通信类型和通信量、网络容量和性能，以及网络现状等与网络建设目标相关的几个主要方面情况的基础上形成分析报告，为网络设计提供依据。需求分析通常采用自顶向下的结构化方法，从以下几个方面着手，逐一深入，在调研的基础上进行充分的分解，从而为网络设计提供基础。

（1）功能需求。功能需求是指用户希望利用网络来完成什么功能，然后依据使用需求、实现成本、未来发展和总预算投资等因素对网络的组建方案进行认真的设计和推敲。

（2）通信需求。通信需求是指了解用户需要的通信类型、通信频度、通信时间和通信量等。

（3）性能需求。性能需求包括容量（带宽）、利用率、最优利用率、吞吐量、可提供负载、精确度、效率、延迟（等待时间）、延时变化量、响应时间、最优网络利用率、端到端

的差错率、精确度和网络效率等。

（4）可靠性需求。可靠性需求主要包括精确度、错误率、稳定性、无故障时间、数据备份等几个方面。

（5）安全需求。衡量网络安全的指标是可用性、完整性（信息的完整、精确和有效，不因人为或非人为的原因而改变信息内容）和保密性（信息只能通过一定方式向有权知道其内容的人员透露）。

（6）运行与维护需求。运行与维护需求是指网络运行和维护费用方面的需求。

（7）管理需求。管理需求主要包括用户管理（创建和维护用户账户及其访问权限）、资源管理、配置管理、性能管理（监视和跟踪网络活动，维护和增强系统性能）和网络维护（防止、检查和解决网络故障问题）。

除此之外，系统分析师还应该了解网络的地理位置，以及对运行环境的要求（包括网络操作系统、数据库和应用软件等相关的需求）。

2．可行性研究

在网络规划阶段，有一个很重要的活动，那就是系统可行性研究，通常从技术可行性、经济可行性、法律可行性和用户使用可行性等方面进行论证。

3．对现有网络的分析与描述

如果是在现有网络系统的基础上进行升级，那么，网络规划阶段的一项重要工作就是对现有网络进行分析，并系统化地描述出来。对现有网络系统进行调研，主要从以下几个方面进行：

（1）服务器的数量和位置。通常服务器所在的中心机房就是网络瓶颈所在，因此，服务器的数量和位置是确定网络瓶颈、解决网络拥塞的前提。

（2）客户机的数量和位置。对客户机的数量和位置进行分析，便于发现在客户机相对集中的地方是否存在瓶颈，结合地理位置确认客户机的网络接入位置是否合理，当存在拥堵现象时，可以重新设计该区域及周边区域的网络结构，均衡网络负载。

（3）同时访问的数量。了解网络中并发访问的情况，并发访问的最大值也就是网络的峰值，是考验网络负载能力的重要参数。通常该值超过网络负载能力时，就会出现问题，需要采取相应措施。可以借助一些工具（例如，网络分析仪）进行连续多天 24 小时全天候跟踪以进行分析。

（4）每天的用户数。每天的用户数可以从一个侧面反映网络的负载和流量。

（5）每次使用的时间。每次网络访问的持续时间将影响到整个模型的建立，对并发的

流量预计有很大的影响，因为其必将对并发人数有影响。

（6）每次数据传输的数据量，即每笔业务所产生的数据流量。

（7）网络拥塞的时间段。可以针对网络拥塞的时间段所发生的数据流、用户数、业务类型进行重点分析，从而找到导致网络拥塞的症结所在。

（8）采用的协议。不同的协议对网络的传输介质和使用的设备，及应用的规划会有不同方面的影响因素。

（9）通信模式。对通信模式的分析，包括双工模式或单工模式、速度和通信地域范围等。

结合对现有网络系统的调研与分析，并在其基础上进行新的网络规划，能够通过以下措施更有效地保证用户的原始投资：

（1）不要推倒重来，要基于现有设备的基础上进行升级和改造。

（2）将现有的设备降级使用（例如，将原有核心层设备降级为分层级使用等），并新增更先进的设备，以提高网络的性能。

4.4.2 网络设计

网络设计的工作是在网络规划的基础上，设计一个能够解决用户问题的方案。在整个设计过程中，首先要确定网络总体目标和设计原则，然后设计网络的逻辑结构，再设计网络的物理结构。

1．网络逻辑结构设计

网络逻辑结构设计是体现网络设计核心思想的关键阶段，在这一阶段根据需求规范和通信规范，选择一种比较适宜的网络逻辑结构，并基于该逻辑结构实施后续的资源分配规划、安全规划等内容。

在逻辑网络设计阶段，需要描述满足用户需求的网络行为及性能，详细说明数据是如何在网络上传输的，此阶段不涉及网络元素的具体物理位置。

网络设计者利用需求分析和现有网络体系分析的结果来设计逻辑网络结构。如果现有的软件、硬件不能满足新网络的需求，现有系统就必须升级。如果现有系统能继续运行使用，可以将它们集成到新设计中来。如果不集成旧系统，网络设计小组可以找一个新系统，对它进行测试，确定是否符合用户的需求。

此阶段最后应该得到一份逻辑网络设计文档，输出的内容包括以下几点：

① 逻辑网络设计图；

② IP 地址方案；

③ 安全方案；

④ 具体的软件、硬件、广域网连接设备和基本的服务；

⑤ 雇佣和培训新网络员工的具体说明；

⑥ 初步对软件、硬件、服务、网络雇佣员工和培训的费用估计。

2．网络物理结构设计

物理网络设计是对逻辑网络设计的物理实现，通过对设备的具体物理分布、运行环境等的确定，确保网络的物理连接符合逻辑连接的要求。在这一阶段，网络设计者需要确定具体的软硬件、连接设备、布线和服务。

如何购买和安装设备，由网络物理结构这一阶段的输出作指导，所以网络物理设计文档必须尽可能详细、清晰，输出的内容如下：

① 物理网络图和布线方案；

② 设备和部件的详细列表清单；

③ 软件、硬件和安装费用的估计；

④ 安装日程表，用以详细说明实际和服务中断的时间及期限；

⑤ 安装后的测试计划；

⑥ 用户培训计划。

3．分层设计

为了能够更好地分析与设计复杂的大型互连网络，在计算机网络设计中，主要采用分层（分级）设计模型，它类似于软件工程中的结构化设计。通过一些通用规则来设计网络，就可以简化设计、优化带宽的分配和规划。在分层设计中，引入了三个关键层的概念，分别是核心层、汇聚层和接入层。

通常将网络中直接面向用户连接或访问网络的部分称为接入层，将位于接入层和核心层之间的部分称为分布层或汇聚层。接入层的目的是允许终端用户连接到网络，因此，接入层交换机具有低成本和高端口密度特性。

汇聚层是核心层和接入层的分界面，完成网络访问策略控制、数据包处理、过滤、寻址，以及其他数据处理的任务。汇聚层交换机是多台接入层交换机的汇聚点，它必须能够处理来自接入层设备的所有通信量，并提供到核心层的上行链路，因此，汇聚层交换机与接入层交换机比较，需要更高的性能，更少的接口和更高的交换速率。

网络主干部分称为核心层，核心层的主要目的在于通过高速转发通信，提供优化、可靠的骨干传输结构，因此，核心层交换机应拥有更高的可靠性，性能和吞吐量。核心层为网络提供了骨干组件或高速交换组件，在纯粹的分层设计中，核心层只完成数据交换的特殊任务。需要根据网络需求的地理距离、信息流量和数据负载的轻重来选择核心层技术，常用的技术包括 ATM、100Base-Fx 和千兆以太网等。在主干网中，考虑到高可用性的需求，通常会使用双星（树）结构，即采用两台同样的交换机，与汇聚层交换机分别连接，并使用链路聚合技术实现双机互联。

4.4.3 网络实施

网络实施是在网络设计的基础上进行设备的购买、安装、调试和系统切换工作。主要包括以下步骤：

（1）工程实施计划。在网络设备安装前，需要编制工程实施计划，列出需实施的项目、费用和负责人等，以便控制投资，按进度要求完成实施任务。工程计划必须包括在网络实施阶段的设备验收、人员培训、系统测试和网络运行维护等具体事务的处理，必须控制和处理所有可预知的事件，并调动有关人员的积极性。

（2）网络设备到货验收。系统中要用到的网络设备到货后，在安装调试前，必须先进行严格的功能和性能测试，以保证购买的产品能很好地满足用户需要。在到货验收的过程中，要做好记录，包括对规格、数量和质量进行核实，以及检查合格证、出厂证、供应商保证书和各种证明文件是否齐全。在必要时利用测试工具进行评估和测试，评估设备能否满足网络建设的需求。如果发现短缺或破损，要求设备提供商补发或免费更换。

（3）设备安装。网络系统的安装和调试需要由专门的技术人员负责。安装项目一般分为综合布线系统、机房工程、网络设备、服务器、系统软件和应用软件等几个部分，不同的部分应分别由专门的工程师进行安装和调试。在这些安装项目中，尤其要注意综合布线系统的质量，因为综合布线一般会涉及隐蔽工程，一旦覆盖后发生故障，查找错误源和恢复故障的代价比较高。

（4）系统测试。系统安装完毕，就要进行系统测试。系统测试是保证网络安全可靠运行的基础。网络测试包括网络设备测试、网络系统测试和网络应用测试三个层次。网络设备测试主要是针对交换机、路由器、防火墙和线缆等传输介质和设备的测试，网络系统测试主要是针对系统的连通性、链路传输率、吞吐率、传输时延和丢包率、链路利用率、错误率、

广播帧和组播帧和冲突率等方面的测试，网络应用测试主要针对 DHCP、DNS、Web、 Email 和 FTP 等服务性能进行测试。

（5）系统试运行。系统调试完毕后，进入试运行阶段。这一阶段是验证系统在功能和性能上是否达到预期目标的重要阶段，也是对系统进行不断调整，直至达到用户要求的重要时刻。

（6）用户培训。一个规模庞大、结构复杂的网络系统往往需要网络管理员来维护，并协调网络资源的使用。对有关人员的培训是网络建设的重要一环，也是保证系统正常运行的重要因素之一。

（7）系统转换。经过一段时间的试运行，系统达到稳定、可靠的水平，就可以进行系统转换工作。系统转换可以采用三种方法，分别是直接转换、并行转换和分段转换，这三种方法的可靠性和成本各不相同，应视具体情况而定。

4.5 网络存储技术

目前，主流的网络存储技术主要有三种，分别是直接附加存储（Direct Attached Storage，DAS）、网络附加存储（Network Attached Storage，NAS）和存储区域网络（Storage Area Network，SAN）。

1．直接附加存储

DAS 是将存储设备通过 SCSI（Small Computer System Interface，小型计算机系统接口）电缆直接连到服务器，其本身是硬件的堆叠，存储操作依赖于服务器，不带有任何存储操作系统。因此，有些文献也把 DAS 称为 SAS（Server Attached Storage，服务器附加存储）。

DAS 的适用环境为：

（1）服务器在地理分布上很分散，通过 SAN 或 NAS 在它们之间进行互连非常困难时；

（2）存储系统必须被直接连接到应用服务器（例如，Microsoft Cluster Server 或某些数据库使用的“原始分区”）上时；

（3）包括许多数据库应用和应用服务器在内的应用，它们需要直接连接到存储器上时。

由于 DAS 直接将存储设备连接到服务器上，这导致它在传递距离、连接数量、传输速率等方面都受到限制。因此，当存储容量增加时，DAS 方式很难扩展，这对存储容量的升级是一个巨大的瓶颈；另一方面，由于数据的读取都要通过服务器来处理，必然导致服务器的处理压力增加，数据处理和传输能力将大大降低；此外，当服务器出现宕机等异常状况时，

也会波及存储数据，使其无法使用。目前 DAS 基本被 NAS 所代替。

2．网络附加存储

采用 NAS 技术的存储设备不再通过 I/O 总线附属于某个特定的服务器，而是通过网络接口与网络直接相连，由用户通过网络访问。NAS 存储系统的结构如图 4-5 所示。

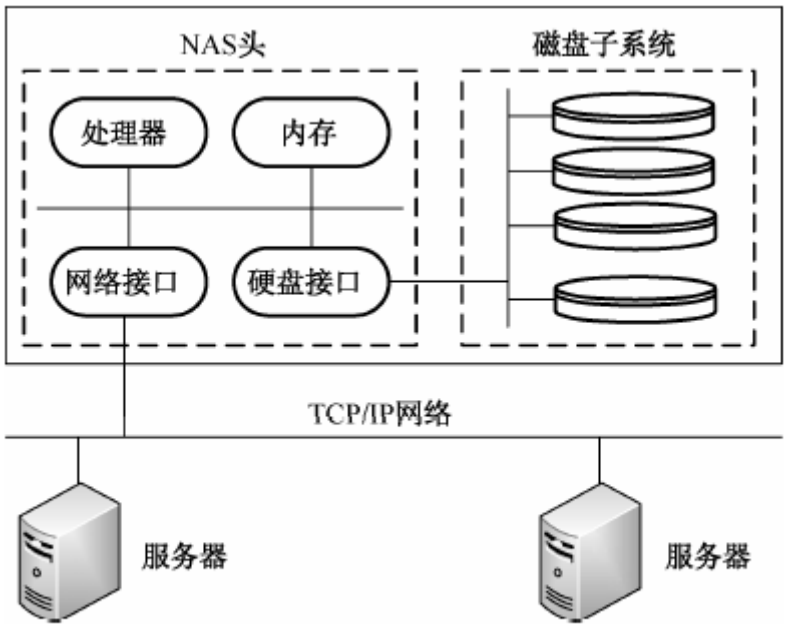


图 4-5　NAS 存储系统的结构

NAS 存储设备类似于一个专用的文件服务器，它去掉了通用服务器的大多数计算功能，而仅仅提供文件系统功能，从而降低了设备的成本。并且为方便存储设备到网络之间能以最有效的方式发送数据，它专门优化了系统硬件与软件架构。NAS 以数据为中心，将存储设备与服务器分离，其存储设备在功能上完全独立于网络中的主服务器，客户机与存储设备之间的数据访问不再需要文件服务器的干预，同时它允许客户机与存储设备之间进行直接的数据访问，所以不仅响应速度快，而且数据传输速率也很高。

NAS 技术支持多种 TCP/IP 网络协议，主要是 NFS（Net File System，网络文件系统）和 CIFS（Common Internet File System，通用 Internet 文件系统）来进行文件访问，所以 NAS 的性能特点是进行小文件级的共享存取。在具体使用时，NAS 设备通常配置为文件服务器，通过使用基于 Web 的管理界面来实现系统资源的配置、用户配置管理和用户访问登录等。

NAS 存储支持即插即用，可以在网络的任一位置建立存储。基于 Web 管理，从而使设备的安装、使用和管理更加容易。NAS 可以很经济地解决存储容量不足的问题，但难以获得满意的性能。

3．存储区域网络

SAN 是通过专用交换机将磁盘阵列与服务器连接起来的高速专用子网。它没有采用文件共享存取方式，而是采用块（block）级别存储。SAN 是通过专用高速网将一个或多个网络存储设备和服务器连接起来的专用存储系统，其最大特点是将存储设备从传统的以太网中分离出来，成为独立的存储区域网络，SAN 的系统结构如图 4-6 所示。

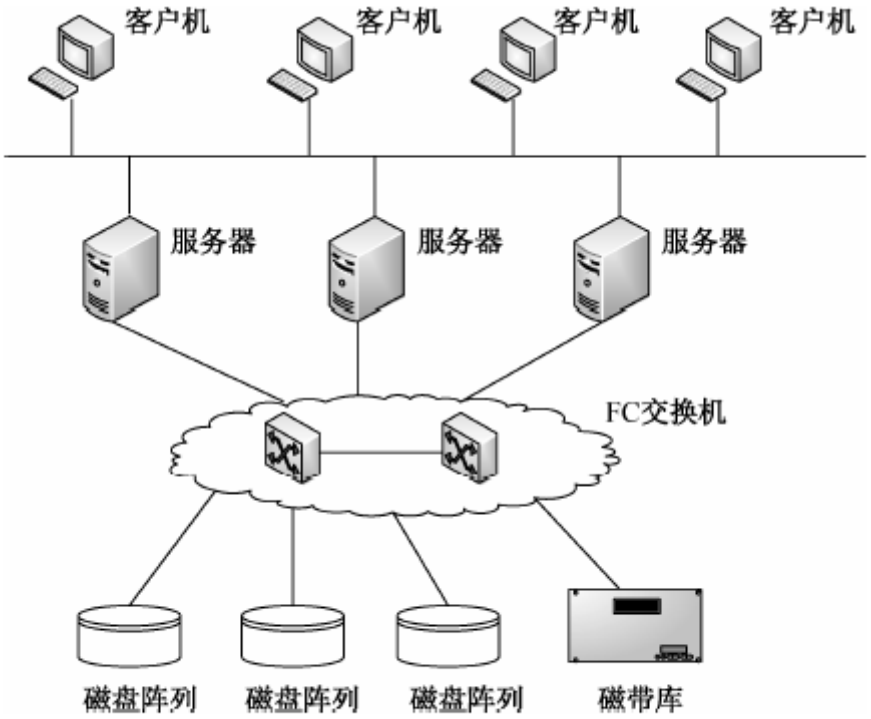


图 4-6　SAN 存储系统的结构

根据数据传输过程采用的协议，其技术划分为 FC SAN 和 IP SAN。另外，还有一种新兴的 IB SAN 技术。

（1）FC SAN。FC（Fiber Channel，光纤通道）和 SCSI 接口一样，最初也不是为硬盘设计开发的接口技术，而是专门为网络系统设计的，随着存储系统对速度的需求，才逐渐应用到硬盘系统中。光纤通道的主要特性有：热插拔性、高速带宽、远程连接、连接设备数量大等。它是当今最昂贵和最复杂的存储架构，需要在硬件、软件和人员培训方面进行大量投资。

FC SAN 由三个基本的组件构成，分别是接口（SCSI、FC)、连接设备（交换机、路由器）和协议（IP、SCSI）。这三个组件再加上附加的存储设备和服务器就构成一个 SAN 系统。它是专用、高速、高可靠的网络，允许独立、动态地增加存储设备，使得管理和集中控制更加简化。

FC SAN 有两个较大的缺陷，分别是成本高和复杂性，其原因就是因为使用了 FC。在光纤通道上部署 SAN，需要每个服务器上都要有 FC 适配器、专用的 FC 交换机和独立的布

线基础架构。这些设施使成本大幅增加，更不用说精通 FC 协议的人员培训成本。

（2）IP SAN。IP SAN 是基于 IP 网络实现数据块级别存储方式的存储网络。由于设备成本低，配置技术简单，可共享和使用大容量的存储空间，因而逐渐获得广泛的应用。

在具体应用上，IP 存储主要是指 ISCSI（Internet SCSI）。作为一种新兴的存储技术，ISCSI 基于 IP 网络实现 SAN 架构，既具备了 IP 网络配置和管理简单的优势，又提供了 SAN 架构所拥有的强大功能和扩展性。ISCSI 是连接到一个 TCP/IP 网络的直接寻址的存储库，通过使用 TCP/IP 协议对 SCSI 指令进行封装，可以使指令能够通过 IP 网络进行传输，而过程完全不依赖于地点。

ISCSI 优势的主要表现在于，首先，建立在 SCSI、TCP/IP 这些稳定和熟悉的标准上，因此安装成本和维护费用都很低；其次，ISCSI 支持一般的以太网交换机而不是特殊的光纤通道交换机，从而减少了异构网络和电缆；最后，ISCSI 通过 IP 传输存储命令，因此可以在整个 Internet 上传输，没有距离限制。

ISCSI 的缺点在于，存储和网络是同一个物理接口，同时协议本身的开销较大，协议本身需要频繁地将 SCSI 命令封装到 IP 包中及从 IP 包中将 SCSI 命令解析出来，这两个因素都造成了带宽的占用和主处理器的负担。但是，随着专门处理 ISCSI 指令的芯片的开发（解决主处理器的负担问题），以及 10G 以太网的普及（解决带宽问题），ISCSI 将有着更好的发展。

4.6 综合布线

综合布线是一种模块化的、灵活性极高的建筑物内或建筑群之间的信息传输通道。通过它可使话音设备、数据设备、交换设备及各种控制设备与信息管理系统连接起来，同时也使这些设备与外部通信网络相连。它还包括建筑物外部网络或电信线路的连接点与应用系统设备之间的所有线缆及相关的连接部件。综合布线由不同系列和规格的部件组成，其中包括：传输介质、相关连接硬件（如配线架、连接器、插座、插头、适配器）及电气保护设备等。

根据《综合布线系统工程设计规范》（GB50311-2007），综合布线系统可分为 7 个部分：工作区、配线子系统、干线子系统、建筑群子系统、设备间、进线间、管理。

（1）工作区：一个独立的需要设置终端设备的区域宜划分为一个工作区。工作区应由配线子系统的信息插座模块延伸到终端设备处的连接缆线及适配器组成。

（2）配线子系统：配线子系统应由工作区的信息插座模块、信息插座模块至电信间配

线设备的配线电缆和光缆、电信间的配线设备及设备缆线和跳线等组成。

（3）干线子系统：干线子系统应由设备间至电信间的干线电缆和光缆，安装在设备间的建筑物配线设备及设备缆线和跳线组成。

（4）建筑群子系统：建筑群子系统应由连接多个建筑物之间的主干电缆和光缆、建筑群配线设备及设备缆线和跳线组成。

（5）设备间：设备间是在每幢建筑物的适当地点进行网络管理和信息交换的场地。对于综合布线系统工程设计，设备间主要是用来安装建筑物配线设备。电话交换机、计算机主机设备及入口设施也可与配线设备安装在一起。

（6）进线间：进线间是建筑物外部通信和信息管线的入口部位，并可作为入口设施和建筑群配线设备的安装场地。

（7）管理：管理应对工作区、电信间、设备间、进线间的配线设备、缆线、信息插座模块等设施，按一定的模式进行标识和记录。

第 5 章 系统性能评价

系统性能是一个系统提供给用户的众多性能指标的混合体。它既包括硬件性能，也包括软件性能。随着计算机技术的不断发展，有关性 能的描述也越来越细化，根据不同的应用需要产生了各种各样的性能指标，如整数运算性能、浮点运算性能、响应时间、网络带宽、稳定性、I/O 吞吐量、SPEC-Int、SPEC-Fp、TPC、Gibson mix 等。有了这些性能指标之后，如何来衡量这些性能指标呢？这就涉及了性能计算。同时用户对性能需求的多样性和广泛性也更进一步加快了计算机技术的发展，并由此出现了一个新的分支：性能设计。性能设计主要包含两方面的内容：一是作为未来计算机技术发展的参考和规划；另一个则是对现有系统进行性能上的调整以达到最优化。在系统性能指标的不断增多和完善过程中，许多公司和个人投身于系统性能的挖掘和实践中，并由此产生了一系列有效的系统性能评价体系。如前面提到 SPEC，已经成为测试 CPU 的最权威的性能测试标准。

本章将就系统性能的 4 个方面进行阐述：

（1）性能指标：描述当前流行系统主要涉及的性能指标；

（2）性能计算：描述当前使用到的主要性能指标的计算方法；

（3）性能设计：描述如何对现有系统进行性能上的调整优化，并介绍几个已经成熟的

设计规则和解决方案；

（4）性能评估：描述如何对当前取得的性能指标进行评价和改进。

5.1 性能指标

在计算机刚刚诞生时，所谓的系统仅仅指的是计算机本身，随着网络的出现和发展，诸如路由器、交换机设备，TCP/IP、SPX/IPX、以太网、光纤网络等网络技术如雨后春笋般涌现。系统的概念也不再局限于单台计算机，而是成为一个集各种通信设备于一体的集成装置。因此，这里所提到的性能指标，既包括软件，也包括硬件。在硬件中，既包括计算机，也包括各种通信交换设备，以及其他网络硬件；在软件中，既包括操作系统和各种通信协议，也包括各种参与到通信中的应用程序，如数据库系统、Web 服务器等。因此，本节要提到的系统性能指标实际上就是这些软硬件的性能指标的集成。

5.1.1 计算机

对计算机评价的主要性能指标如下：

1．时钟频率（主频）

主频是计算机的主要性能指标之一，在很大程度上决定了计算机的运算速度。CPU 的工作节拍是由主时钟来控制的，主时钟不断产生固定频率的时钟脉冲，这个主时钟的频率即是 CPU 的主频。主频越高，意味着 CPU 的工作节拍就越快，运算速度也就越快。但从 2000 年 IBM 发布第一款双核处理器开始，多核心已经成为 CPU 发展的一个重要方向。原来单以时钟频率来计算性能指标的方式已经不合适了，还得看单个 CPU 中的内核数。现在主流的服务器 CPU 大都为八核或十二核，未来更可能发展到 32 核，96 核甚至更多。

2．高速缓存

高速缓存可以提高 CPU 的运行效率。目前一般采用两级高速缓存技术，有些使用三层。高速缓冲存储器均由静态 RAM（Random Access Memory，随机存取存储器）组成，结构较复杂，在 CPU 管芯面积不能太大的情况下，L1 级高速缓存的容量不可能做得太大。采用回写（WriteBack）结构的高速缓存。它对读和写操作均有可提供缓存。而采用写通（Write-through）结构的高速缓存，仅对读操作有效。L2 及 L3 高速缓存容量也会影响 CPU 的性能，原则是越大越好。

3．运算速度

运算速度是计算机工作能力和生产效率的主要表征，它取决于给定时间内 CPU 所能处理的数据量和 CPU 的主频。其单位一般用 MIPS（百万条指令/秒）和 MFLOPS（百万次浮点运算/秒）。MIPS 用于描述计算机的定点运算能力；MFLOPS 则用来表示计算机的浮点运算能力。

4．运算精度

即计算机处理信息时能直接处理的二进制数据的位数，位数越多，精度就越高。参与运算的数据的基本位数通常用基本字长来表示。PC（Personal Computer，个人计算机）机的字长，已由 8088 的准 16 位（运算用 16 位，I/O 用 8 位）发展到现在的 32 位、64 位。大中型计算机一般为 32 位和 64 位。巨型机一般为 64 位。在单片机中，目前主要使用的是 8 位和 16 位字长。

5．内存的存储容量

内存用来存储数据和程序，直接与 CPU 进行信息交换。内存的容量越大，可存储的数据和程序就越多，从而减少与磁盘信息交换的次数，使运行效率得到提高。存储容量一般用字节（Byte）数来度量。PC 的内存已由 286 机配置的 1MB，发展到现在主流的 1G 以上。而在服务器领域中，一般的都在 2~8G，多的如银行系统中省级结算中心使用的大型机，内存高达上百 GB。内存容量的加大，对于运行大型软件十分必要，尤其是对于大型数据库应用。内存数据库的出现更是将内存的使用发挥到了极致。

6．存储器的存取周期

内存完成一次读（取）或写（存）操作所需的时间称为存储器的存取时间或者访问时间。而连续两次读（或写）所需的最短时间称为存储周期。存储周期越短，表示从内存存取信息的时间越短，系统的性能也就越好。目前内存的存取周期约为几到几十 ns（10-9 秒）。

存储器的 I/O 的速度、主机 I/O 的速度，取决于 I/O 总线的设计。这对于慢速设备（例如键盘、打印机）关系不大，但对于高速设备则效果十分明显。例如对于当前的硬盘，它的外部传输率已可达 100MBps、133MBps 以上。

7．数据处理速率

数据处理速率（Processing Data Rate，PDR）的计算公式是：PDR=L/R。其中： L=0.85G＋0.15H＋0.4J＋0.15K；R=0.85M＋0.09N＋0.06P

其中：G 是每条定点指令的位数；　　M 是平均定点加法时间；

H 是每条浮点指令的位数；　　N 是平均浮点加法时间；

J 是定点操作数的位数；　　P 是平均浮点乘法时间；

K 是浮点操作数的位数；

另外还规定：G>20 位，H>30 位；从主存取一条指令的时间等于取一个字的时间；指令和操作数都存放在同一个主存，无变址或间址操作；允许有先行或并行取指令功能，此时选用平均取指令时间。

PDR 主要用来度量 CPU 和主存储器的速度，它没有涉及高速缓存和多功能等。因此，PDR 不能度量机器的整体速度。

8．响应时间

某一事件从发生到结束的这段时间。其含义将根据应用的不同而变化。响应时间既可以是原子的，也可以是由几个响应时间复合而成的。在计算机技术的发展中，早在 1968 年，米勒先生就已给出 3 个经典的有关响应时间的建议。

0.1 秒：用户感觉不到任何延迟。

1.0 秒：用户愿意接受的系统立即响应的时间极限。即当执行一项任务的有效反馈时间在 0.1～1 秒之内时，用户是愿意接受的。超过此数据值，则意味着用户会感觉到有延迟，但只要不超过 10 秒，用户还是可以接受的。

10 秒：用户保持注意力执行本次任务的极限，如果超过此数值时仍然得不到有效的反馈，客户会在等待计算机完成当前操作时转向其他的任务。

9．RASIS 特性

RASIS 特性是可靠性（Reliability）、可用性（Availability）、可维护性（Serviceability）、完整性（Integraity）和安全性（Security）五者的统称。可靠性是指计算机系统在规定的工作条件下和规定的工作时间内持续正确运行的概率。可靠性一般是用平均无故障时间（Mean Time To Failure，MTTF）或平均故障间隔时间（Mean Time Between Failure，MTBF）来衡量。

可维护性是指系统发生故障后能尽快修复的能力，一般用平均故障修复时间（Mean Time To Repair，MTTR）来表示。取决于维护人员的技术水平和对系统的熟悉程度，同时和系统的可维护性也密切相关。

有关这些特性的详细知识，将在 17.5 节介绍。

10．平均故障响应时间

平均故障响应时间（TAT）即从出现故障到该故障得到确认修复前的这段时间。该指标反应的是服务水平。平均故障响应时间越短，对用户系统的影响越小。

11．兼容性

兼容性是指一个系统的硬件或软件与另一个系统或多种操作系统的硬件或软件的兼容

能力，是指系统间某些方面具有的并存性，即两个系统之间存在一定程度的通用性。兼容是一个广泛的概念，它包括数据和文件的兼容、程序和语言级的兼容、系统程序的兼容、设备的兼容以及向上兼容和向后兼容等。

除了上述性能指标之外，还有其他性能指标，例如综合性能指标如吞吐率、利用率；定性指标如保密性、可扩充性；功能特性指标如文字处理能力、联机事务处理能力、I/O 总线特性、网络特性等。

5.1.2 网络

网络是一个是由多种设备组成的集合体。其性能指标也名目繁多。一般可以将这些性能指标分为以下几类：

（1）设备级性能指标。网络设备提供的通信量的特征，是确定网络性能的一个重要因素。计算机网络设备（主要指路由器）的标准性能指标主要包括吞吐量（信道的最大吞吐量为“信道容量”）、延迟、丢包率和转发速度等。

（2）网络级性能指标。可达性、网络系统的吞吐量、传输速率、信道利用率、信道容量、带宽利用率、丢包率、平均传输延迟、平均延迟抖动、延迟/吞吐量的关系、延迟抖动/吞吐量的关系、丢包率/吞吐量的关系等。

（3）应用级性能指标。QOS、网络对语言应用的支持程度、网络对视频应用的支持程度、延迟/服务质量的关系、丢包率/服务质量的关系、延迟抖动/服务质量的关系等。

（4）用户级性能指标。计算机网络是一种长周期运行的系统。可靠性和可用性是长周期运行系统非常重要的服务性能，是决定系统是否有实际使用价值的重要参数。

（5）吞吐量。在没有帧丢失的情况下，设备能够接受的最大速率。网络吞吐量可以帮助寻找网络路径中的瓶颈。例如，即使客户端和服务器都被分别连接到各自的 100Mbps 以太网上，但是如果这两个 100Mbps 以太网被 10Mbps 的以太网连接起来，那么 10Mbps 的以太网就是网络的瓶颈。

网络吞吐量非常依赖于当前的网络负载情况。因此，为了得到正确的网络吞吐量，最好在不同时间（一天中的不同时刻，或者一周中不同的天）分别进行测试，只有这样才能得到对网络吞吐量的全面认识。

有些网络应用程序在开发过程的测试中能够正常运行，但是到实际的网络环境中却无法正常工作（由于没有足够的网络吞吐量）。这是因为测试只是在空闲的网络环境中，没有考

虑到实际的网络环境中还存在着其他的各种网络流量。所以，网络吞吐量定义为剩余带宽是有实际意义的。

5.1.3 操作系统

现代操作系统的基本功能是管理计算机系统的硬件、软件资源，这些管理工作分为处理机管理、存储器管理、设备管理、文件管理、作业管理和通信事务管理。

操作系统的性能与计算机系统工作的优劣有着密切的联系。评价操作系统的性能指标一般有：

（1）系统的可靠性。

（2）系统的吞吐量，是指系统在单位时间内所处理的信息量，以每小时或每天所处理的各类作业的数量来度量。

（3）系统响应时间，是指用户从提交作业到得到计算结果这段时间，又称周转时间；

（4）系统资源利用率，指系统中各个部件、各种设备的使用程度。它用在给定时间内，某一设备实际使用时间所占的比例来度量。

（5）可移植性。

5.1.4 数据库管理系统

数据库为了保证存储在其中的数据的安全和一致，必须有一组软件来完成相应的管理任务，这组软件就是 DBMS，DBMS 随系统的不同而不同，但是一般来说，它应该包括以下几方面的内容：

（1）数据库描述功能。定义数据库的全局逻辑结构，局部逻辑结构和其他各种数据库对象。

（2）数据库管理功能。包括系统配置与管理，数据存取与更新管理，数据完整性管理和数据安全性管理。

（3）数据库的查询和操纵功能。该功能包括数据库检索和修改。

（4）数据库维护功能。包括数据引入引出管理，数据库结构维护，数据恢复功能和性能监测。为了提高数据库系统的开发效率，现代数据库系统除了 DBMS 之外，还提供了各种支持应用开发的工具。

因此，衡量数据库管理系统的主要性能指标包括数据库本身和管理系统两部分。

数据库和数据库管理系统的性能指标包括数据库的大小、单个数据库文件的大小、数据库中表的数量、单个表的大小、表中允许的记录（行）数量、单个记录（行）的大小、表上所允许的索引数量、数据库所允许的索引数量、最大并发事务处理能力、负载均衡能力、最大连接数。

5.1.5 Web 服务器

Web 服务器也称为 WWW 服务器，主要功能是提供网上信息浏览服务。

在 UNIX 和 Linux 平台下使用最广泛的 HTTP 服务器是 W3C、NCSA 和 Apache 服务器，而 Windows 平台使用 IIS 的 Web 服务器。跨平台的 Web 服务器有 IBM WebSphere、BEA WebLogic、Tomcat 等。在选择使用 Web 服务器时，应考虑的本身特性因素有性能、安全性、日志和统计、虚拟主机、代理服务器、缓冲服务和集成应用程序等。

Web 服务器的主要性能指标包括最大并发连接数、响应延迟、吞吐量（每秒处理的请求数）、成功请求数、失败请求数、每秒点击次数、每秒成功点击次数、每秒失败点击次数、尝试连接数、用户连接数等。

5.2 性能计算

随着计算机系统复杂度的不断增长，性能指标也在不断地增长，这也增加了衡量计算机系统性能的难度。如何在众多指标中选取合适的性能指标，以及选择何种衡量方法都成为一项重要的课题，因此也衍生了各种性能评估体系。由于性能指标种类繁多，不可能一一列举，本节主要介绍一些常用性能指标的计算方法。在实际应用时，往往是对这些常用性能指标的复合计算，然后通过算法加权处理得到最终结果。

性能指标计算的主要方法有：定义法、公式法、程序检测法、仪器检测法。定义法主要根据其定义直接获取其理想数据，公式法则一般适用于根据基本定义所衍生出的复合性能指标的计算，而程序检测法和仪器检测法则是通过实际的测试来得到其实际值（由于测试的环境和条件不定，其结果也可能相差比较大）。

有些性能指标，在不同的环境中，其名字相同，但计算方式和结果可能相差甚远，例如，吞吐量、带宽等，在计算机、路由器、交换机和网络中多处出现了有关吞吐量的定义，但其具体的含义不尽相同。

1．MIPS 的计算方法

$$\mathrm{MIPS}=\frac{\text{指令条数}}{\text{执行时间}\times 10^{6}}=\frac{Fz}{\mathrm{CPI}}=\mathrm{IPC}\times Fz$$

公式中，Fz 为处理机的工作主频，CPI（Cycles Per Instruction）为每条指令所需的平均时钟周期数，IPC 为每个时钟周期平均执行的指令条数。

例如，如果要计算 Pentium IV/2.4E 处理机的运算速度，因为 Pentium IV/2.4E 处理机的 IPC=2（或 CPI=0.5），Fz=2400MHz，所以 MIPSP4/2.4E = IPC′ Fz=2′ 2400 = 4800MIPS。

2．峰值计算

衡量计算机性能的一个重要指标就是计算峰值或者浮点计算峰值，它是指计算机每秒钟能完成的浮点计算最大次数。包括理论浮点峰值和实测浮点峰值。

理论浮点峰值是该计算机理论上能达到的每秒钟能完成浮点计算最大次数，它主要是由 CPU 的主频决定。

理论浮点峰值 = CPU 主频′ CPU 每个时钟周期执行浮点运算的次数′ 系统中 CPU 数

希赛教育专家提示：CPU 每个时钟周期执行浮点运算的次数是由处理器中浮点运算单元的个数及每个浮点运算单元在每个时钟周期能处理几条浮点运算来决定的。

3．等效指令速度

静态指令使用频度指的是在程序中直接统计的计算机速度。动态指令使用频度指的是在程序执行过程中统计的指令速度。在计算机发展的早期，用加法指令的运算速度来衡量计算机的速度。后来发展成为等效指令速度法或吉普森（Gibson）法，在这种方法中，通常加、减法指令占 50%，乘法指令占 15%，除法指令占 5%，程序控制指令占 15%，其他指令占 15%。

例如，我国最早研制的小型计算机 DJS-130，定点 16 位，加法速度每秒 50 万次，但没有硬件乘法和除法等指令。用软件实现乘法和除法，速度降低 100 倍左右，则其等效指令速度为

$$1/\left(\frac{0.80}{0.5}+\frac{0.20}{0.5/100}\right)=0.02\mathrm{MIPS}$$

即每秒 2 万次，由于乘法和除法用软件实现，等效速度降低了 25 倍。

又如，如果浮点开平方操作 FPSQR 的比例为 2%，它的 CPI 为 100，其他浮点操作的比例为 23%，它的 CPI 4.0，其余指令的 CPI 1.33，则该处理机的等效 CPI 为：100 ′

2%+4′ 23%+ 1.33′ 75%= 3.92

如果 FPSQR 操作的 CPI 也为 4.0，则其等效 CPI 为：4 ′ 25%+1.33′ 75% =2.00

由于改进了仅占 2%的 FPSQR 操作的 CPI，使等效速度提高了近一倍。

5.3 性能设计

本节主要讨论如何利用阿姆达尔解决方案进行系统性能优化及负载均衡方面的知识。

5.3.1 阿姆达尔解决方案

阿姆达尔定律是这样的：系统中对某部件采用某种更快的执行方式，所获得的系统性能的改变程度，取决于这种方式被使用的频率，或所占总执行时间的比例。

阿姆达尔定律定义了采用特定部件所取得的加速比。假定使用某种增强部件，计算机的性能就会得到提高，那么加速比就是如下公式所定义的比率：

$$\text{加速比} = \frac{\text{不使用增强部件时完成整个任务的时间}}{\text{使用增强部门时完成整个任务的时间}}$$

加速比反映了使用增强部件后完成一个任务比不使用增强部件完成同一任务加快了多少。阿姆达尔定律为计算某些情况下的加速比提供了一种便捷的方法。加速比主要取决于两个因素：

（1）在原有的计算机上，能被改进并增强的部分在总执行时间中所占的比例。这个值称之为增强比例，它永远小于等于 1。

（2）通过增强的执行方式所取得的改进，即如果整个程序使用了增强的执行方式，那么这个任务的执行速度会有多少提高，这个值是在原来条件下程序的执行时间与使用增强功能后程序的执行时间之比。

原来的机器使用了增强功能后，执行时间等于未改进部分的执行时间加上改进部分的执行时间：

$$\text{新的执行时间} = \text{原来的执行时间} \times \left((1 - \text{增强比例}) + \frac{\text{增强比例}}{\text{增强加速比}} \right)$$

总的加速比等于两种执行时间之比：

$$\text{总加速比} = \frac{\text{原来的执行时间}}{\text{新的执行时间}} = \frac{1}{\left((1 - \text{增强比例}) + \frac{\text{增强比例}}{\text{增强加速比}} \right)}$$

5.3.2 负载均衡

负载均衡是由多台服务器以对称的方式组成一个服务器集合，每台服务器都具有等价的地位，都可以单独对外提供服务而无须其他服务器的辅助。通过某种负载分担技术，将外部发送来的请求均匀地分配到对称结构中的某一台服务器上，而接收到请求的服务器独立地回应客户的请求。

当用户发现 Web 站点负载量非常大时，应当考虑使用负载均衡技术来将负载平均分摊到多个内部服务器上。如果有多个服务器同时执行某一个任务时，这些服务器就构成一个集群。使用集群技术可以用最少的投资获得接近于大型主机的性能。

1．负载均衡技术的类型

目前，比较常用的负载均衡技术主要有以下几种：

（1）基于特定服务器软件的负载均衡。很多网络协议都支持“重定向”功能，例如，在 HTTP 协议中支持 Location 指令，接收到这个指令的浏览器将自动重定向到 Location 指明的另一个 URL 上。由于发送 Location 指令比起执行服务请求，对 Web 服务器的负载要小得多，因此可以根据这个功能来设计一种负载均衡的服务器。当 Web 服务器认为自己负载较大的时候，它就不再直接发送回浏览器请求的网页，而是送回一个 Location 指令，让浏览器在服务器集群中的其他服务器上获得所需要的网页。

在这种方式下，服务器本身必须支持这种功能，然而具体实现起来却有很多困难。例如，一台服务器如何能保证它重定向过的服务器是比较空闲的，并且不会再次发送 Location 指令？Location 指令和浏览器都没有这方面的支持能力，这样很容易在浏览器上形成一种死循环。因此这种方式实际应用当中并不多见，使用这种方式实现的服务器集群软件也较少。有些特定情况下可以使用 CGI（包括使用 FastCGI 或 mod_perl 扩展来改善性能）来模拟这种方式去分担负载，而 Web 服务器仍然保持简洁、高效的特性。此时，避免 Location 循环的任务将由用户的 CGI 程序来承担。

（2）基于 DNS（Domain Name Server，域名服务器）的负载均衡。通过 DNS 服务中的随机名字解析来实现负载均衡，在 DNS 服务器中，可以为多个不同的地址配置同一个名字，而最终查询这个名字的客户机将在解析这个名字时得到其中一个地址。因此，对于同一个名字，不同的客户机会得到不同的地址，它们也就访问不同地址上的 Web 服务器，从而达到负载均衡的目的。

DNS 负载均衡的优点是简单易行，并且服务器可以位于互联网的任意位置上，当前使用在

包括 Yahoo 在内的 Web 站点上。然而它也存在不少缺点，一个缺点是为了保证 DNS 数据及时更新，一般都要将 DNS 的刷新时间设置得较小，但太小就会造成太大的额外网络流量，并且更改了 DNS 数据之后也不能立即生效；另一个缺点是 DNS 负载均衡无法得知服务器之间的差异，它不能做到为性能较好的服务器多分配请求，也不能了解到服务器的当前状态，甚至会出现客户请求集中在某一台服务器上的偶然情况。

（3）反向代理负载均衡。使用代理服务器可以将请求转发给内部的 Web 服务器，使用这种加速模式显然可以提升静态网页的访问速度。因此也可以考虑使用这种技术，让代理服务器将请求均匀地转发给多台内部 Web 服务器，从而达到负载均衡的目的。这种代理方式与普通的代理方式有所不同，标准代理方式是客户使用代理访问多个外部 Web 服务器，而这种代理方式是多个客户使用它访问内部 Web 服务器，因此也被称为反向代理模式。

实现这个反向代理能力并不能算是一个特别复杂的任务，但是在负载均衡中要求特别高的效率，这样实现起来就不是十分简单的事了。每针对一次代理，代理服务器就必须打开两个连接，一个为对外的连接，一个为对内的连接。因此，当连接请求数量非常大的时候，代理服务器的负载也非常大，最后，反向代理服务器会成为服务的瓶颈。例如，使用 Apache 的 mod_rproxy 模块来实现负载均衡功能时，提供的并发连接数量受 Apache 本身的并发连接数量的限制。一般来讲，可以使用它来连接数量不是特别大、但每次连接都需要消耗大量处理资源的站点来进行负载均衡，例如搜寻。

使用反向代理的好处是，可以将负载均衡和代理服务器的高速缓存技术结合在一起，以提供有益的性能；其具备额外的安全性，外部客户不能直接访问真实的服务器。并且实现起来可以采用较好的负载均衡策略，将负载非常均衡地分给内部服务器，不会出现负载集中到某个服务器的偶然现象。

（4）基于 NAT（Network Address Translation，网络地址转换）的负载均衡技术。网络地址转换指的是在内部地址和外部地址之间进行转换，以便具备内部地址的计算机能访问外部网络，而当外部网络中的计算机访问地址转换网关拥有的某一外部地址时，地址转换网关能将其转发到一个映射的内部地址上。因此如果地址转换网关能将每个连接均匀转换为不同的内部服务器地址，此后，外部网络中的计算机就各自与自己转换得到的地址上的服务器进行通信，从而达到负载分担的目的。

地址转换可以通过软件方式来实现，也可以通过硬件方式来实现。使用硬件方式进行操作一般称为交换，而当交换必须保存 TCP 连接信息的时候，这种针对 OSI/RM 网络层的操作就被称为第四层交换。支持负载均衡的网络地址转换为第四层交换机的一种重要功能，由

于它基于定制的硬件芯片，因此其性能非常优秀，很多交换机声称具备 400MB～ 800MB 的第四层交换能力；然而也有一些资料表明，在如此快的速度下，大部分交换机就不再具备第四层交换能力了，而仅仅支持第三层甚至第二层交换。

使用软件方式来实现基于网络地址转换的负载均衡则要实际得多，除了一些厂商提供的解决方法之外，更有效的方法是使用免费的自由软件来完成这项任务。其中包括 Linux Virtual Server Project 中的 NAT 实现方式。一般来讲，使用这种软件方式来实现地址转换，中心负载均衡器存在带宽限制，在 100MBps 的快速以太网条件下，能得到最高达 80MBps 的带宽，然而在实际应用中，可能只有 40MBps～60MBps 的可用带宽。

（5）扩展的负载均衡技术。上面使用网络地址转换来实现负载分担，毫无疑问所有的网络连接都必须通过中心负载均衡器，那么如果负载特别大，以至于后台的服务器的数量不再在是几台、十几台，而是上百台甚至更多，这时，即便是使用性能优秀的硬件交换机也会遇到瓶颈。此时问题将转变为，如何将那么多台服务器分布到各个互联网的多个位置，分散网络负担。当然这可以通过综合使用 DNS 和 NAT 两种方法来实现，然而更好的方式是使用一种半中心的负载均衡方式。

在这种半中心的负载均衡方式下，即当客户请求发送给负载均衡器的时候，中心负载均衡器将请求打包并发送给某个服务器，而服务器的回应请求不再返回给中心负载均衡器，而是直接返回给客户，因此中心负载均衡器只负责接受并转发请求，其网络负担就较小了。

2．服务器负载均衡

服务器负载均衡一般用于提高服务器的整体处理能力，并提高可靠性、可用性和可维护性，最终目的是加快服务器的响应速度，从而提高用户的体验度。

负载均衡从结构上分为本地负载均衡（Local Server Load Balance）和全域负载均衡（Global Server Load Balance），前者是指对本地的服务器群做负载均衡，后者是指对分别放置在不同的地理位置、有不同的网络及服务器群之间做负载均衡。

全域负载均衡有以下特点：

（1）解决网络拥塞问题，服务就近提供，实现地理位置无关性；

（2）对用户提供更好的访问质量；

（3）提高服务器响应速度；

（4）提高服务器及其他资源的利用效率；

（5）避免了数据中心单点失效。

5.4 性能评估

性能评估是对一个系统进行各项检测，并形成一份直观的文档，因此性能评估是通过各项测试来完成的。评估的一个目的是为性能的优化提供参考，而性能优化涉及的面很广，也很复杂，而且永无止境。对于不同的应用程序，优化的方法会有一些区别。

5.4.1 基准测试程序

把应用程序中用得最多、最频繁的那部分核心程序作为评价计算机性能的标准程序。称为基准测试程序（benchmark）。

（1）整数测试程序：Dhrystone。用 C 语言编写，100 条语句。包括：各种赋值语句，各种数据类型和数据区，各种控制语句，过程调用和参数传送，整数运算和逻辑操作。

VAX-11/780z 的测试结果为每秒 1757 个 Dhrystones，即：

1VAXMIPS=1757 Dhrystones/s

（2）浮点测试程序：Linpack。用 FORTRAN 语言编写，主要是浮点加法和浮点乘法操作。用 MFPOPS（Million Floating Point Operations Per Second）表示 GFLOPS、 TFLOPS。

（3）Whetstone 基准测试程序。用 FORTRAN 语言编写的综合性测试程序，主要包括：浮点运算、整数算术运算、功能调用、数组变址、条件转移、超越函数。测试结果用 Kwips 表示。

（4）SPEC 基准测试程序。SPEC 基准测试程序（System Performance Evaluation Cooperative，系统性能评估联盟）由 30 个左右世界知名计算机大厂商所支持的非盈利的合作组织，包括 IBM、AT&T、BULL、Compaq、CDC、DG、DEC、Fujitsu、HP、Intel、 MIPS、Motolola、SGI、SUN、Unisys 等；SPEC 能够全面反映机器的性能，具有很高的参考价值。SPEC 以 AX-11/780 的测试结果作为基数，当前主要的基准测试程序有 SPEC int_base_rate 2000、SPEC fp_base_rate 2000 和 SPEC JBB 2000 等。还有基于某种数据库运行环境下的测试，也是可以参考的数值。在采用通用基准测试程序时，要注意真实的业务流程和使用环境与通用测试基准的业务流程和使用环境的异同，这样，基准测试值才有参考价值。

（5）TPC 基准程序。TPC（Transaction Processing Council，事务处理委员会）成立于 1988 年，已有 40 多个成员，用于评测计算机的事务处理、数据库处理、企业管理与决策支持等方面的性能。1989 年以来相继发表的 TPC 基准测试程序包括 TPC-A、TPC-B、TPC-C、TPC-D、TPC-H 和 TPC-W 等。其中 TPC-A 用于在线联机事务处理下更新密集的数据库环境下的性

能测试，TPC-B 用于数据库系统及运行它的操作系统的核心性能测试， TPC-C 则用于在线联机事务处理测试，TPC-D 用于决策支持系统测试，TPC-H 是基于 TPC-D 基础上决策支持基准测试，还有 TPC-W 是用于电子商务应用软件测试。

TPC-C 是衡量 OLTP 系统的工业标准。它测试广泛的数据库功能，包括查询、更新和排队袖珍型批处理（mini-batch）事务。这一规范在关键领域十分严格，如数据库透明性和事务处理隔离性。许多 IT 专家把 TPC-C 作为“真实世界”OLTP 系统性能的一个很好的指示器。独立审核员认证基准测试（benchmark）的结果，TPC 还有全套的公开报告。

（6）Linpack 测试。Linpack 是国际上最流行的用于测试高性能计算机系统浮点性能的测试。通过对高性能计算机采用高斯消元法求解一元 N 次稠密线性代数方程组的测试，评价高性能计算机的浮点性能。

Linpack 测试包括三类，Linpack100、Linpack1000 和 HPL。Linpack100 求解规模为 100 阶的稠密线性代数方程组，它只允许采用编译优化选项进行优化，不得更改代码，甚至代码中的注释也不得修改。Linpack1000 要求求解 1000 阶的线性代数方程组，达到指定的精度要求，可以在不改变计算量的前提下做算法和代码上的优化。HPL 即 High Performance Linpack，也叫高度并行计算基准测试，它对数组大小 N 没有限制，求解问题的规模可以改变，除基本算法（计算量）不可改变外，可以采用其他任何优化方法。前两种测试运行规模较小，已不太适合现代计算机的发展。

HPL 是针对现代并行计算机提出的测试方式。用户在不修改任意测试程序的基础上，可以调节问题规模的大小（矩阵大小）、使用 CPU 数目、使用各种优化方法等来执行该测试程序，以获取最佳的性能。HPL 采用高斯消元法求解线性方程组。求解问题规模为 N 时，浮点运算次数为 2/3′ N3 -2N2。

因此，只要给出问题规模 N，测得系统计算时间 T，峰值=计算量（2/3′ N3- 2N2)/ 计算时间 T，测试结果以浮点运算每秒（Flops）给出。HPL 测试结果是 TOP500 排名的重要依据。

5.4.2 Web 服务器的性能评估

在 Web 服务器的测试中，能够反映其性能的主要包括最大并发连接数、响应延迟和吞吐量（每秒处理的请求数）几个参数。

现在常见的 Web 服务器性能评测方法有基准性能测试、压力测试和可靠性测试。基准

测试即采用前面所提到的各种基准程序对其进行测试；压力测试则是采用一些测试工具（这些测试工具的主要特征就是能够模拟足够数量的并发操作）来测试 Web 服务器的一些性能指标，如最大并发连接数，间接测试响应时间，以及每秒钟可以处理的请求数目。通过这种压力测试，不但可以考察 Web 服务器的各项性能指标，而且可以找到服务器的瓶颈所在，然后通过参数调整，让服务器运行得更高效。

IxWeb 是美国 IXIA 公司的一个有关 Web 测试的解决方案，它是一个高性能业务负 载生成与分析应用系统，可在 TCP 和应用层模拟现实世界的业务负载方案，能够对设备 进行强度测试、检验转发策略和验证 4～7 层的性能。IxWeb 通过模拟用户（客户端）来测试 Web 服务器。每一个 IXIA 测试仪端口都有独立的 CPU 和内存，运行 Linux 操作系统，具备完整的 TCP/IP 协议栈，每个端口可以模拟大量的 Web 客户，每个客户能够产生大量的并发连接。它还可以通过同时模拟客户端和服务器端，对内容交换机等设备进行测试。

IxWeb 能够配置会话，以模拟使用路由、交换和 NAT 环境中的用户。为了进一步测试实际应用方案，IxWeb 能够修改 TCP 参数，并提供对用户通过诸如拨号调制解调器和 DSL 等多种接入机制联网进行仿真的功能。通过配置“思考时间”（Think Times，用户操作之间的时间间隔）来对用户的行为进行仿真，还可进一步增加测试的真实性。

IxWeb 中 HTTP 1.0/1.1 支持的方法包括：GET、POST、PUT、HEAD、DELETE，可以修改 HTTP 客户端与服务器包头规格。IxWeb 对 SSL 的支持使用户能够模拟访问安全网站的大量 SSL 客户端会话。IxWeb 具有生成真正的 SSL 握手和 HTTPS 功能。

IxWeb 能够仿真成千上万个下载大文件的 FTP 用户及提供大量所需数据的相应设备。IxWeb 提供了灵活的用户配置和全面的统计数字，以测定同步 FTP 用户的最大数量，以及内容交换机、负载均衡器、服务器与防火墙的吞吐量。

IxWeb 提供了广泛翔实的实时统计数字及记录日志，并可将其导出到标准文件格式，以便定制报告生成。测试一旦完成，即可以用包括 HTML 在内的多种文件格式生成内容丰富的报告。

5.4.3 系统监视

系统监视的目标是为了评估系统性能。要监视系统性能，需要收集某个时间段内的 3 种不同类型的性能数据：

（1）常规性能数据。该信息可帮助识别短期趋势（如内存泄漏）。经过一两个月的数据

收集后，可以求出结果的平均值并用更紧凑的格式保存这些结果。这种存档数据可帮助人们在业务增长时作出容量规划，并有助于在日后评估上述规划的效果。

（2）比较基准的性能数据。该信息可帮助人们发现缓慢、历经长时间才发生的变化。通过将系统的当前状态与历史记录数据相比较，可以排除系统问题并调整系统。由于该信息只是定期收集的，所以不必对其进行压缩存储。

（3）服务水平报告数据。该信息可帮助人们确保系统能满足一定的服务或性能水平，也可能会将该信息提供给并不是性能分析人员的决策者。收集和维护该数据的频率取决于特定的业务需要。

进行系统监视通常有 3 种方式。一是通过系统本身提供的命令，如 UNIX/Liunx 中的 w、ps、last，Windows 中的 netstat 等；二是通过系统记录文件查阅系统在特定时间内的运行状态；三是集成命令、文件记录和可视化技术，提供直观的界面，操作人员只需要进行一些可视化的设置，而不需要记忆繁杂的命令行参数，即可完成监视操作，如 Windows 的 Perfmon 应用程序。

目前，已经有些厂商提供专业化的监视平台，将上面 3 种方式集成到一个统一的监控平台，进行统一监控，并提供各类分析数据和分析报表，帮助用户进行性能的评估和诊断。如 IBM 公司提供的 Tivoli、HP 公司提供的 Sitescope 等。

第 6 章：开发方法

软件开发方法是软件开发的方法学。自从“软件危机”爆发以来，软件研究人员就在对开发方法进行不断地研究，以期能够提高软件的质量、降低软件的成本。经过 40 多年的研究，人们提出了很多开发方法，如最初的结构化开发到现在非常流行的面向对象的开发方法等。本章将介绍软件生命周期、软件开发模型、软件重用技术、逆向工程及形式化开发方法。

6.1 软件生命周期

软件生命周期也就是软件生存的周期。同万物一样，软件也有诞生和消亡，软件生命周期就是指软件自开始构思与研发到不再使用而消亡的过程。有关软件生命周期的阶段划分，不同的标准有不同的规定。在 GB8566-88（《软件工程国家标准——计算机软件开发规范》）中将软件生命周期划分为 8 个阶段：可行性研究与计划、需求分析、概要设计、详细设计、

实现、集成测试、确认测试、使用和维护。

（1）可行性研究与计划：在决定是否开发软件之前，首先需要进行可行性研究。通过可行性研究，来确定开发此软件的必要性，并根据可行性研究的结果初步确定软件的目标、范围、风险、开发成本等内容。从而制定出初步的软件开发计划。通过可行性研究，如果确定该软件具有研发的必要，则将产生《可行性研究报告》和《软件开发计划》，并进入需求分析的阶段。

（2）需求分析：需求分析是软件开发的重要阶段。经过可行性研究后，初步确定了软件开发的目标和范围，之后则需要对软件的需求进行细致的分析，来确定软件要做成什么样的。需求分析是软件开发过程中极其重要的一环，如果需求分析出现了重大偏差，那么软件开发必然会偏离正确的道路，越走越远。尤其是需求分析的错误如果在软件开发后期才被发现，修正的代价是非常大的。

（3）概要设计：概要设计确定整个软件的技术蓝图，负责将需求分析的结果转化为技术层面的设计方案。在概要设计中，需要确定系统架构、各子系统间的关系、接口规约、数据库模型、编码规范等内容。概要设计的结果将作为程序员的工作指南，供程序员了解系统的内部原理，并在其基础上进行详细设计和编码工作。

（4）详细设计：详细设计完成编码前最后的设计，详细设计在概要设计的基础上，进行细化，如类设计。详细设计不是开发过程中必需的阶段，在一些规模较小、结构简单的系统中，详细设计往往被省略。同样，在某一次软件开发中，可能只会对部分关键模块进行详细设计。

（5）实现：实现过程包括编码和单元测试。单元测试指的是对刚刚编写出的一个小的程序单元进行测试，如某一个过程、方法或函数。因为单元测试的对象是小的程序单元，而不是完整的程序，因此往往需要编写一些测试程序来进行测试。有效的单元测试可以大大提高编码的质量，降低软件系统的缺陷率。

（6）集成测试：集成测试又称为组装测试。通过单元测试的程序并不意味着没有缺陷，当程序单元被集成到一起进行交互的时候，往往会出现单元测试中不能发现的问题。同单元测试不同，集成测试必须经过精心的组织，指定集成测试计划，确定如何将这些程序单元集成到一起，按照什么样的顺序进行测试，使用哪些测试数据等问题。

（7）确认测试：当完成集成测试后，软件之间的接口方面的错误已经排除，这时需要验证软件是否同需求一致，是否达到了预期目标。同集成测试一样，确认测试也需要进行计划和组织，逐步地验证软件系统同需要的一致性。经过确认测试的软件将投入正常使用，并

进入维护期。

（8）使用和维护：即使通过了单元测试、集成测试和确认测试，也不可能发现软件系统中的全部缺陷；软件系统的需求也会根据业务的发展变化而变化。因此，在软件使用过程中，必须不断地对软件进行维护，修正软件中的缺陷，修改软件中已经不能适应最新情况的功能或者增加新的功能。软件维护的过程会贯穿整个软件的使用过程。当使用和维护阶段结束后，软件系统也就自然消亡，软件系统的生命周期结束。

6.2 软件开发模型

在计算机刚刚诞生的年代，计算机是一种只有天才才能掌握的工具。人们对软件的认知仅仅停留在程序的层面上，所谓的软件开发就是那些能够掌握计算机的天才们写的一些只有计算机才能理解的二进制序列。但随着技术的发展，软件的复杂度不断提高，人们进入了大规模软件开发的时代。这时，人们发现，软件系统已经变得非常复杂，需要遵循一定的开发方法才能取得成功，于是称这些模式化的开发方法为开发模型。

6.2.1 瀑布模型

顾名思义，瀑布模型就如同瀑布一样，从一个特定的阶段流向下一个阶段，如图 6-1 所示。

1.瀑布模型的核心思想

瀑布模型认为，软件开发是一个阶段化的精确的过程。就像要制造一艘航空母舰，首先需要知道航空母舰的参数（长、宽、高、排水量、航速等）。在这些参数的技术上需要对航空母舰进行设计，设计包括总体设计和详细设计。只有设计得一清二楚的图纸才能交付施工，否则造出的零件肯定拼装不到一起。制造完毕后，要把这些零件一个一个地拼装起来，拼装成发动机、船舱等部分，并检查这些部分是否符合设计标准，这就是集成测试。最后，把各个部分组合在一起，造出一艘巨大的航母。这个过程正如图 5-1 中的描述，软件要经过需求分析、总体设计、详细设计、编码、调试、集成测试和系统测试阶段才能够被准确地实现。在图 6-1 中，每一阶段都有回到前一阶段的反馈线，这指的是，在软件开发中当在后续阶段发现缺陷的时候，可以把这个缺陷反馈到上一阶段进行修正。

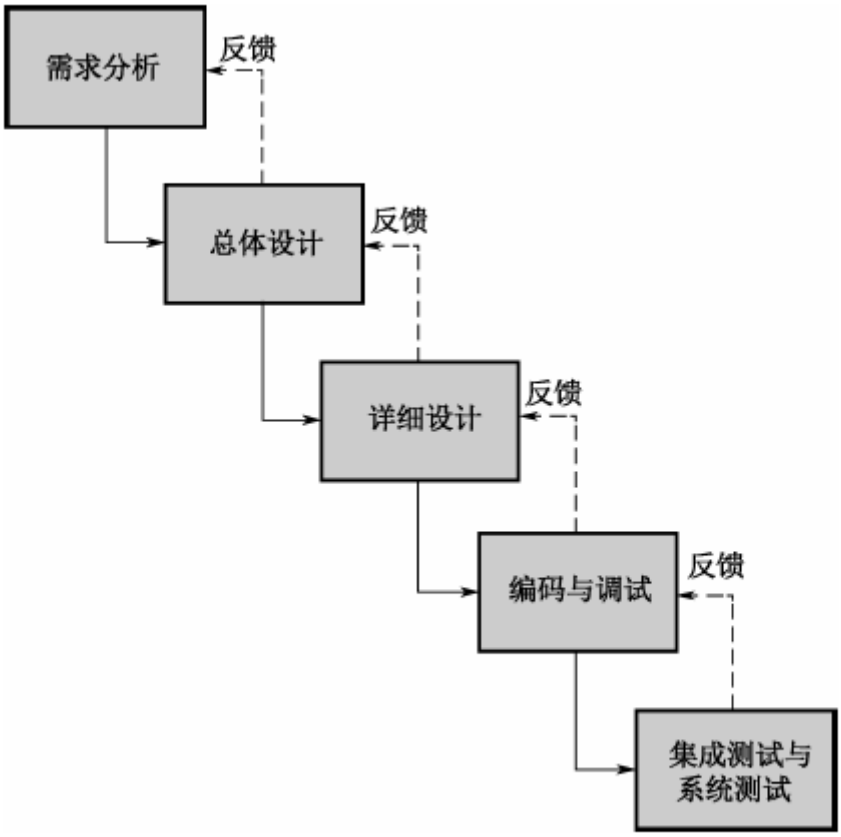


图 6-1　瀑布模型

从图 6-1 中可以看出瀑布模型的一个重要特点：软件开发的阶段划分是明确的，一个阶段到下一个阶段有明显的界线。在每个阶段结束后，都会有固定的文档或源程序流入下一阶段。在需求分析阶段结束后，需要有明确的描述软件需求的文档；总体设计结束后，需要有描述软件总体结构的文档；详细设计结束后，需要有可以用来编码的详细设计文档；而编码结束后，代码本身被作为文档流到下一个阶段。因此也称瀑布模型是面向文档的软件开发模型。

当软件需求明确、稳定时，可以采用瀑布模型按部就班地开发软件，当软件需求不明确或变动剧烈时，瀑布模型中往往要到测试阶段才会暴露出需求的缺陷，造成后期修改代价太大，难以控制开发的风险。

2．瀑布 V 模型

瀑布 V 模型是瀑布模型的一种变体。随着对瀑布模型的应用，人们发现，缺陷是无法避免的，任何一个阶段都会在软件中引入缺陷，而最后的测试也不能保证软件完全没有缺陷，只能争取在交付前发现更多的缺陷。测试成为软件开发中非常重要的环节，测试的质量直接影响到软件的质量。因此，人们对瀑布模型进行了小小的更改，提出了更强调测试的瀑布 V

模型，如图 6-2 所示。

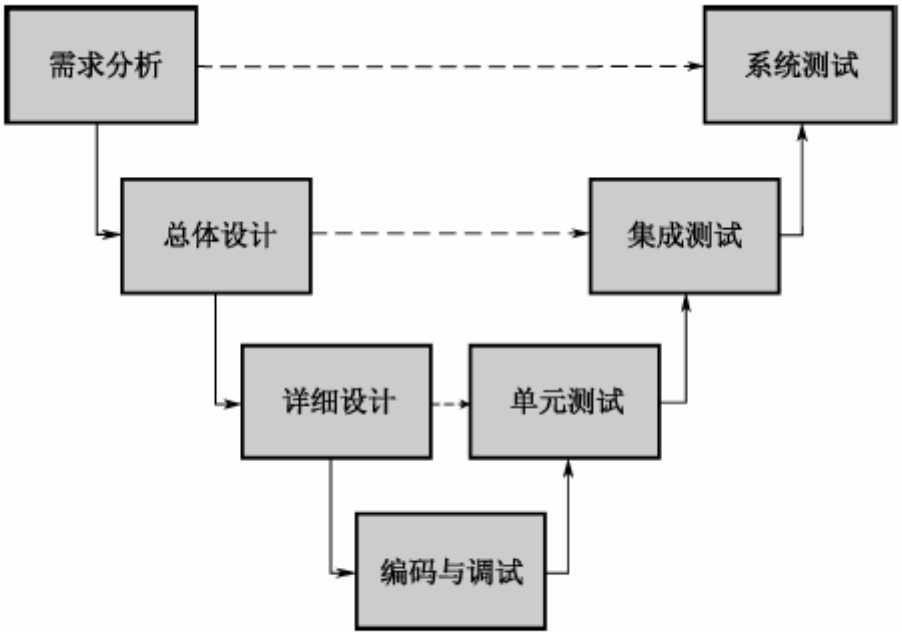


图 6-2　瀑布 V 模型

整个瀑布模型在编码与调试阶段转了个弯，形成了一个对称的 V 字。瀑布 V 模型同标准瀑布模型一样，在进行完需求分析后就将进入总体设计阶段，但是除总体设计外，需求分析还有一条虚线指向系统测试。这指的是，需求分析的结果将作为系统测试的准则，即需求分析阶段也将产生同软件需求一致的系统测试；同时软件产品是否符合最初的需求将在系统测试阶段得到验证。以此类推，总体设计对应了集成测试，详细设计对应了单元测试。瀑布 V 模型不但保持了瀑布模型的阶段式文档驱动的特点，而且更强调了软件产品的验证工作。

3．瀑布模型的缺点

虽然是经典的开发模型，但瀑布模型中仍存在一些难以克服的缺陷，即使是在改进的瀑布 V 模型中还是会存在。

首先，在瀑布模型中，需求分析阶段是一切活动的基础，设计、实现和验证活动都是从需求分析阶段的结果导出的。一旦需求分析的结果不完全正确，存在偏差，那么后续的活动只能放大这个偏差，在错误的道路上越走越远。事实上，由于用户和开发者的立场、经验、知识域都不相同，不同的人对同一件事物的表述也不同，这就造成需求分析的结果不可能精确、完整地描述整个软件系统。所以瀑布模型后期的维护工作相当繁重，而这些维护工作大多都是修正在需求分析阶段引入的缺陷。这个问题是瀑布模型难以克服的。

其次，瀑布模型难以适应变化。在瀑布模型中精确地定义了每一个阶段的活动和活动结果，而每一阶段都紧密依赖于上一阶段的结果。如果在软件的后期出现了需求的变化，整个系统又要从头开始。

再次，使用瀑布模型意味着当所有阶段都结束才能最终交付软件产品，所以在提出需求后需要相当长一段时间的等待才能够看到最终结果，才能发现软件产品究竟能不能够满足客户的需求。

最后，文档驱动型的瀑布模型除了制造出软件产品外还将产生一大堆的文档，大部分的文档对客户没有任何意义，但完成这些对客户没有意义的文档却需要花费大量的人力。所以瀑布模型也是一种重载过程。

6.2.2 演化模型

瀑布模型看起来很好，随着一个又一个阶段的流过，软件系统就被建立起来了。可是在应用软件开发的过程中，人们发现很难一次性完全理解用户的需求、设计出完美的架构，开发出可用的系统，这是由于人的认知本身就是一个过程，这个过程是渐进的、不断深化的。对于复杂问题，“做两次”肯定能够做得更好。那么，对于软件开发这个复杂而且与人的认知过程紧密相关的事也应该是一个渐进的过程。

演化模型正是基于这个观点提出的。一般情况下，一个演化模型可以看做若干次瀑布模型的迭代，当完成一个瀑布模型后，重新进入下一个迭代周期，软件在这样的迭代过程中得以演化、完善。根据不同的迭代特点，演化模型可以演变为螺旋模型、增量模型和原型法开发。

6.2.3 螺旋模型

螺旋模型将瀑布模型和演化模型结合起来，不仅体现了两个模型的优点，而且还强调了其他模型均忽略了的风险分析。螺旋模型的每一周期都包括需求定义、风险分析、工程实现和评审 4 个阶段，由这 4 个阶段进行迭代，软件开发过程每迭代一次，软件开发就前进一个层次。采用螺旋模型的软件过程如图 6-3 所示。

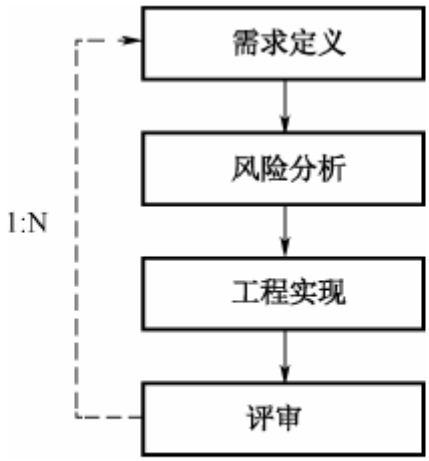


图 6-3　采用螺旋模型的软件过程

螺旋模型的基本做法是在“瀑布模型”的每一个开发阶段前，引入一个非常严格的风险识别、风险分析和风险控制。它把软件项目分解成一个个小项目，每个小项目都标识一个或多个主要风险，直到所有的主要风险因素都被确定。

螺旋模型强调风险分析，使得开发人员和用户对每个演化层出现的风险都有所了解，继而做出应有的反应。因此，螺旋模型特别适用于庞大而复杂、具有高风险的系统，对于这些系统，风险是软件开发潜在的、不可忽视的不利因素，它可能在不同程度上损害软件开发过程，影响软件产品的质量。减小软件风险的目标是在造成危害之前，及时对风险进行识别、分析，决定采取何种对策，进而消除或减少风险的损害。

与瀑布模型相比，螺旋模型支持用户需求的动态变化，为用户参与软件开发的所有关键决策提供了方便，有助于提高目标软件的适应能力，为项目管理人员及时调整管理决策提供了便利，从而降低了软件开发风险。

但是，不能说螺旋模型绝对比其他模型优越，事实上，螺旋模型也有其自身的缺点：

（1）采用螺旋模型，需要具有相当丰富的风险评估经验和专业知识。在风险较大的项目开发中，如果未能及时标识风险，势必会造成重大损失。

（2）过多的迭代次数会增加开发成本，延迟提交时间。

6.2.4 增量模型

演化模型的另一种形式是增量模型。在系统的技术架构成熟、风险较低的时候，可以采用增量的方式进行系统开发，这样可以提前进行集成测试和系统测试，缩短初始版本的发布周期，提高用户对系统的可见度。

对于增量模型，通常有两种策略。一是增量发布的办法。即首先做好系统的分析和设计工作，然后将系统划分为若干不同的版本，每一个版本都是一个完整的系统，后一版本以前一版本为基础进行开发，扩充前一版本的功能。在这种策略中，第一版本往往是系统的核心功能，可以满足用户最基本的需求，随着增量的发布，系统的功能逐步地丰富、完善起来。用户在很短的时间内就可以得到系统的初始版本并进行试用。试用中的问题可以很快地反馈到后续开发中，从而降低了系统的风险。在应用增量模型中需要注意：

（1）每一个版本都是一个完整的版本。虽然最初的几个增量不能完全地实现用户需求，但这些版本都是完整的、可用的。

（2）版本间的增量要均匀，这一点是很重要的。如果第一个版本花费一个月的时间，而第二个版本需要花费 6 个月的时间，这种不均匀的分配会降低增量发布的意义，需要重新调整。

另一种策略是原型法。同增量发布不同，原型法的每一次迭代都经过一个完整的生命周期。当用户需求很不明确或技术架构中存在很多不可知因素的时候，可以采用原型法。在初始的原型中，针对一般性的用户需求进行快速实现，并不考虑算法的合理性或系统的稳定性。这个原型的主要目的是获得精确的用户需求，或验证架构的可用性。一般情况下，会在后面的开发中抛弃这个原型，重新实现完整的系统。

6.2.5 构件组装模型

随着软构件技术的发展，人们开始尝试利用软构件进行搭积木式的开发，即构件组装模型。在构建组装模型中，当经过需求分析定义出软件功能后，将对构件的组装结构进行设计，将系统划分成一组构件的集合，明确构件之间的关系。在确定了系统构件后，则将独立完成每一个构件，这时既可以开发软件构件，也可以重用已有的构件，当然也可以购买或选用第三方的构件。构件是独立的、自包容的，因此架构的开发也是独立的，构件之间通过接口相互协作。

构件组装模型的一般开发过程如图 6-4 所示。

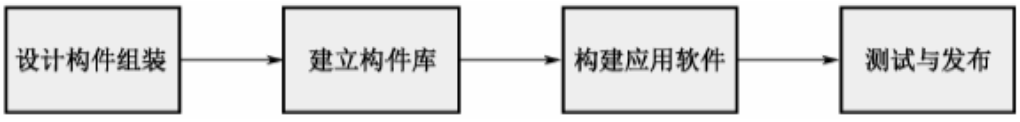


图 6-4　构件组装模型

构件组装模型的优点如下：

（1）构件的自包容性让系统的扩展变得更加容易

（2）设计良好的构件更容易被重用，降低软件开发成本

（3）构件的粒度较整个系统更小，因此安排开发任务更加灵活，可以将开发团队分成若干组，并行地独立开发构件。

鱼与熊掌不可兼得，构件组装模型也有明显的缺点：

（1)对构件的设计需要经验丰富的架构设计师,设计不良的构件难以实现构件的优点，降低构件组装模型的重用度。

（2）在考虑软件的重用度时，往往会对其他方面做出让步，如性能等。

（3）使用构件组装应用程序时，要求程序员熟练地掌握构件，增加了研发人员的学习成本。

（4）第三方构件库的质量会最终影响到软件的质量，而第三方构件库的质量往往是开发团队难以控制的。

6.3 统一过程

统一过程（Unified Process，UP）是由 Rational 公司开发的一种迭代的软件过程，是一个优秀的软件开发模型，它提供了完整的开发过程解决方案，可以有效地降低软件开发过程的风险，经过裁剪的 UP 可以适应各种规模的团队和系统。

1．UP 的二维模型

UP 是一个很有特色的模型，它本身是一个二维的结构，如图 6-5 所示。对于 UP 而言，时间主线就是横轴的阶段，随着时间的流逝，软件开发活动总要经过初始、细化、构建和交付这 4 个阶段方能完成。而纵轴的工作流程则描述了在不同的阶段需要进行的主要工作。例如在初始阶段，软件组织需要进行大量的调研，对软件进行业务建模、需求，同时进行一些设计以验证建模的合理性，还要进行一些实施甚至测试和部署的工作，用以验证需求和设计的工作及开发系统原型，当然配置与变更管理、项目管理和环境是在任何阶段都是不能缺少的。

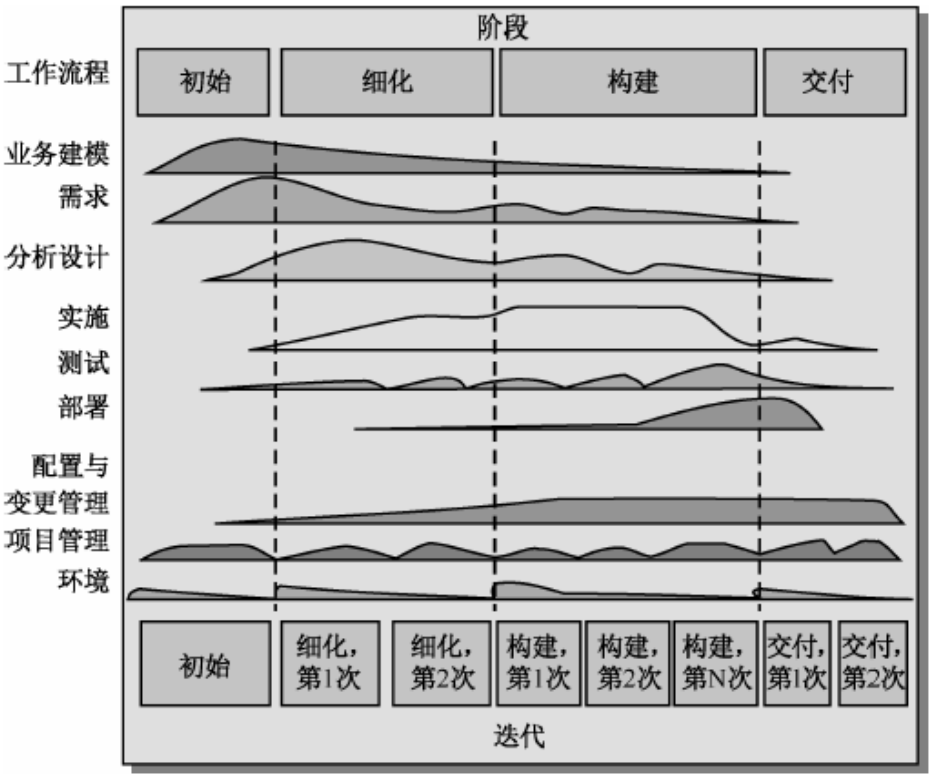


图 6-5　UP 二维模型图

从这个模型中可以看出 UP 迭代的特点。任何一个阶段的工作都不是绝对的，都是相互交叠配合的。但每一个阶段都有其侧重点：

在初始阶段，开发者刚刚接入系统，此时最重要的工作是界定系统范围，明确系统目的。在这一阶段，业务建模和需求工作成了重头戏。

在细化阶段，开发者需要抽象出软件的逻辑模型，设计出软件的架构，在这一阶段，分析设计工作是最主要的工程活动。

在构建阶段，开发者需要基本完成系统的构建，使之成为一个完整的实体，并进行测试和部署，在这一阶段，实施和测试是最主要的活动。

当进入交付阶段（该阶段也经常被称为转移阶段），软件系统需求已经完全成熟或产品化，或进入下一个版本。在这一阶段不可避免地要对软件系统进行重构、修改、测试和部署。

在这 4 个阶段中，各有侧重点，但也不是像瀑布模型那样完全不允许其他活动的存在。在初始阶段，为了验证开发者的想法，就需要进行一部分的实施和测试；而即使到了交付阶段，需要也可能会发生变化，仍然需要进行部分业务建模、需求和设计的活动。

在每个阶段中，系统推进不是一蹴而就的。在图中将细化阶段划分为第 1 次细化和第 2 次细化，将构建阶段也划分为 3 个小阶段。在实际开发中，可以根据实际的需要划分为更多的小阶段来完成。

对于纵轴而言，业务建模、需求、分析设计、实施、测试、部署、配置与变更管理、项目管理、环境称为 UP 的 9 个核心工作流。可以把这 9 个工作流进行简单的分类以帮助理解，业务建模、需求、分析设计、实施、测试和部署是工程活动，而配置与变更管理、项目管理和环境是管理活动。

在这 9 个工作流中，前 8 个可以说是绝大多数人都耳熟能详的东西，而“环境”工作流则相对难以理解。“环境”工作流很重要，也可以称之为“环境管理”。俗语说，“巧妇难为无米之炊”，“环境”工作流就是为软件开发准备“米”的活动。在软件开发中，需要为各种工作准备相应的工作环境，在工作环境中需要包含必需的工具、活动的指南、活动的流程规范、工作产品的模板、基本的开发设施等。在很多组织中，“环境”工作流没有得到应有的重视，或者完全被忽视，以为为开发者提供了工作台和计算机就万事大吉了，其实这种做法是错误的。每一个开发团体都有自己特定的活动准则和规范，这些准则和规范是团体协作的基础，万万少不得。没有合理的工具配备，没有充分的指南、规范和模板，软件开发的活动肯定是放羊式的管理，管理者除了一些“羊毛”外什么也收获不到。观察 UP 模型就可以发现，在每一阶段的最开始，“环境”工作流都有一个小小的波峰。在这里面，开发团队需要为开发环境进行相应的准备并在后续活动中为开发环境提供支持。

2．UP 的生命周期

前面已经提到，UP 模型的时间主线是阶段，UP 的生命周期也是与阶段一一对应的。在 UP 的生命周期中共有 4 个里程碑：

（1）目标里程碑。目标里程碑对应着先启阶段的结束，当开发者可以明确软件系统的目标和范围时即达到了该里程碑。

（2）架构里程碑。架构里程碑是 UP 生命周期中的第二个里程碑，在这个里程碑前，开发者需要确定稳定的系统架构。

（3）能力里程碑。当系统已经足够的稳定和成熟并完成 Alpha 测试后，认为达到了第 3 个里程碑。

（4）发布里程碑。在达到发布里程碑前，需要完成系统的测试、完成系统发布和用户培训等工作。

在经过这 4 个里程碑后，即为一个完整的生命周期，开发出一个新的版本。此时可以关闭该产品的开发，也可以迭代进入下一版本。

3．UP 的特点

UP 是一个特点鲜明的开发模型，下面列出 UP 的一些特点：

（1）UP 是一个迭代的二维开发模型，在生命周期的每一阶段都可以进行需求、设计等活动。UP 不但给出了迭代的生命周期，还给出了生命周期每一阶段的迭代指南。

（2）采用不同迭代方式的 UP 可以演变为演化模型或增量模型。

（3）UP 的迭代特点使得更容易控制软件开发的风险。

（4）虽然 UP 是一个迭代的开发模型，但 UP 本身并不属于敏捷方法。相反，一般认为，未经裁减的 UP 是一个重载过程。

（5）在实际应用中可以根据具体问题对 UP 进行裁减，从而使其可以适应各种规模的软件和开发团队。

4．架构设计师在 UP 中的活动

架构设计师在 UP 活动中承担着非常重要的角色。在 UP 中，架构设计师除了需要建立系统架构模型外，还需要：

（1）同需求人员和项目管理人员密切协作。

（2）细化软件架构。

（3）保持整个架构的概念完整性。具体地说，架构设计师不但需要设计系统架构，还需要定义设计方法、设计指南、编
码指南、评审设计等工作。因此，有人也称 UP 是一个以架构为中心的开发模型。

6.4 敏捷方法

2001 年 2 月，在美国的犹他州，17 位“无政府主义者”共同发表了《敏捷软件开发宣言》，在宣言中指出：

尽早地、持续地向客户交付有价值的软件对开发人员来说是最重要的。

拥抱变化，即使在开发的后期。敏捷过程能够驾驭变化，保持客户的竞争力。

经常交付可工作的软件，从几周到几个月，时间范围越小越好。

在整个项目中，业务人员和开发者紧密合作。

围绕士气高昂的团队进行开发，为团队成员提供适宜的环境，满足他们的需要，并给予足够的信任。

在团队中，最有效率的、也是效果最好的沟通方式是面对面地交流。

可以工作的软件是进度首要的度量方式。

可持续地开发。投资人、开发团队和用户应该保持固定的节奏。

不断追求优秀的技术和良好的设计有助于提高敏捷性。

要简单，尽可能减少工作量。减少工作量的艺术是非常重要的。

最好的架构、需求和设计都来自于一个自我组织的团队。

团队要定期地总结如何能够更有效率，然后相应地自我调整。

至此，敏捷软件联盟建立起来，敏捷软件开发方法进入了大发展的时代。这份宣言也就是敏捷方法的灯塔，所有的敏捷方法都在向这个方向努力。目前已形成多种敏捷方法，其中 XP 传播最为广泛，为此，本节主要介绍 XP，其次简单介绍另外几种很有特色的开发模型。

6.4.1 极限编程

XP 方法可以说是敏捷联盟中最鲜艳的一面旗帜，也是相对来说最成熟的一种。XP 方法的雏形最初形成于 1996—1999 年间，Kent Beck、Ward Cunningham、Ron Jeffery 夫妇在开发 C3 项目（Chrysler Comprehensive Compensation）的实践中总结出了 XP 的基本元素。在此之后，Kent Beck 和他的一些好朋友们一起在实践中完善提高，终于形成了极限编程方法。

XP 是一种轻量（敏捷）、高效、低风险、柔性、可预测、科学而且充满乐趣的软件开发方式。与其他方法论相比，其最大的不同在于：

（1）在更短的周期内，更早地提供具体、持续的反馈信息。

（2）迭代地进行计划编制，首先在最开始迅速生成一个总体计划，然后在整个项目开发过程中不断地发展它。

（3）依赖于自动测试程序来监控开发进度，并及早地捕获缺陷。

（4）依赖于口头交流、测试和源程序进行沟通。

（5）倡导持续的、演化式的设计。

（6）依赖于开发团队内部的紧密协作。

（7）尽可能达到程序员短期利益和项目长期利益的平衡。

XP 由价值观、原则、实践和行为四个部分组成，它们彼此相互依赖、关联，并通过行为贯穿于整个生命周期。

1．四大价值观

XP 的核心是其总结的沟通、简单、反馈、勇气四大价值观，它们是 XP 的基础，也是 XP 的灵魂。

（1）沟通。通常，程序员给人留下的印象就是“内向、不善言谈”，项目中的许多问题就出在这些缺乏沟通的开发人员身上。由于某个程序员做出了一个设计决定，但是却不能够及时地通知团队中的其他成员，结果使得团队在协作与配合上出现很多麻烦。而在传统的开发方法中，并不在意这种口头沟通不畅的问题，而是希望借助于完善的流程和面面俱到的文档、报表、计划来替代，但是，这又引入了效率不高的新问题。

XP 方法认为，如果小组成员之间无法做到持续的、无间断的交流，那么协作就无从谈起。从这个角度来看，通过文档、报表等人工制品进行交流，具有很大的局限性。因此， XP 组合了诸如结对编程这样的最佳实践，鼓励大家进行口头交流、通过交流解决问题，提高效率。

（2）简单。XP 方法在工作中秉承“够用即好”的思路，也就是尽量地简单化，只要今天够用就行，不考虑明天会出现的新问题。这一点看上去十分容易，但要真正做到保持简单的工作其实是很难的，因为在传统的开发方法中，都要求开发人员对未来做一些预先规划，以便对今后可能发生的变化预留一些扩展空间。

沟通和简单之间还有一种相当微妙的互相支持关系。一方面，团队成员之间沟通得越多，就越容易明白哪些工作需要做，哪些工作不需要做；另一方面，系统越简单，需要沟通的内容也就越少，沟通也将更加全面。

（3）反馈。是什么原因使得客户、管理层这么不理解开发团队？究其症结，就是开发的过程中缺乏必要的反馈。在很多项目中，当开发团队经历过了需求分析阶段之后，在一个相当长的时间段中，是没有任何反馈信息的。整个开发过程对于客户和管理层而言就像一个黑盒子，进度完全看不到。而且，在项目开发过程中，这样的现象不仅出现在开发团队与客户、管理层之间，还包括在开发团队内部。因此，开发团队需要更加注重反馈。反馈对于任何软件项目的成功都是至关重要的，而在 XP 方法论中则更进一步，通过持续、明确的反馈来暴露软件状态的问题。

反馈与沟通有着良好的配合，及时和良好的反馈有助于沟通。而简单的系统，更有利于测试和反馈。

（4）勇气。在应用 XP 方法时，每时每刻都在应对变化：由于沟通良好，会有更多需求变更的机会；由于时刻保持系统的简单，新的变化会带来一些重新开发的需要；由于反馈及时，会有更多中间打断思路的新需求。总之，这一切使得开发团队处于变化之中，因此，这时就需要有勇气来面对快速开发，面对可能的重新开发。勇气可以来源于沟通，因为它使得高风险、高回报的试验成为可能；勇气可以来源于简单，因为面对简单的系统，更容易鼓

起勇气；勇气可以来源于反馈，因为可以及时获得每一步前进的状态（自动测试），会让人更勇于重构代码。

希赛教育专家提示：在 XP 的四大价值观之下，隐藏着一种更深刻的东西，那就是尊重。因为这一切都建立在团队成员之间相互关心、相互理解的基础之上。

2．十二个最佳实践

在 XP 中，集成了 12 个最佳实践，有趣的是，它们没有一个是创新的概念，大多数概念和编程一样老。其主要的创新点在于提供一种良好的思路将这些最佳实践结合在一起，并且确保尽可能彻底地执行，使得它们能够在最大程度上互相支持。

（1）计划游戏。计划游戏的主要思想就是先快速地制定一份概要的计划，然后，随着项目细节的不断清晰，再逐步完善这份计划。计划游戏产生的结果是一套用户故事及后续的一两次迭代的概要计划。

（2）小型发布。XP 方法秉承的是“持续集成、小步快走”的哲学思维，也就是说每一次发布的版本应该尽可能地小，当然前提条件是每个版本有足够的商业价值，值得发布。由于小型发布可以使得集成更频繁，客户获得的中间结果越频繁，反馈也就越频繁，客户就能够实时地了解项目的进展情况，从而提出更多的意见，以便在下一次迭代中计划进去，以实现更高的客户满意度。

（3）隐喻。相对而言，隐喻比较令人费解。根据词典中的解释是：“一种语言的表达手段，它用来暗示字面意义不相似的事物之间的相似之处”。隐喻常用于四个方面：寻求共识、发明共享语汇、创新的武器、描述架构。

希赛教育专家提示：如果能够找到合适的隐喻是十分快乐的，但并不是每一种情况都可以找到恰当的隐喻，因此，没有必要去强求，而是顺其自然。

（4）简单设计。强调简单的价值观，引出了简单性假设原则，落到实处就是“简单设计”实践。这个实践看上去似乎很容易理解，但却又经常被误解，许多批评者就指责 XP 忽略设计是不正确的。其实，XP 的简单设计实践并不是要忽略设计，而是认为设计不应该在编码之前一次性完成，因为那样只能建立在“情况不会发生变化”或者“我们可以预见所有的变化”之类的谎言的基础上。

（5）测试先行。对于有些团队而言，有时候程序员会以“开发工作太紧张”为理由，继而忽略测试工作。这样，就导致了一个恶性循环，越是没空编写测试程序，代码的效率与质量越差，花在找缺陷、解决缺陷的时间也越来越多，实际产能大大降低。由于产能降低，因此时间更紧张，压力就更大。

（6）重构。重构是一种对代码进行改进而不影响功能实现的技术，XP 需要开发人员在“闻到代码的坏味道”时，就有重构代码的勇气。重构的目的是降低变化引发的风险、使得代码优化更加容易。

（7）结对编程。从 20 世纪 60 年代开始，就有类似的实践在进行，长年以来的研究结果给出的结论是，结对编程的效率反而比单独编程更高。一开始虽然会牺牲一些速度，但慢慢地，开发速度会逐渐加快。究其原因，主要是结对编程大大降低了沟通的成本，提高了工作的质量。结对编程技术被誉于 XP 保证工作质量、强调人文主义的一个最典型的实践，应用得当还能够使开发团队协作更加顺畅、知识交流与共享更加频繁、团队稳定性也会更加牢固。

（8）集体代码所有制。由于 XP 方法鼓励团队进行结对编程，而且认为结对编程的组合应该动态地搭配，根据任务的不同、专业技能的不同进行最优组合。因此，每一个人都会遇到不同的代码，代码的所有制就不再适合于私有，因为那样会给修改工作带来巨大的不便。所谓集体代码所有制，就是团队中的每个成员都拥有对代码进行改进的权利，每个人都拥有全部代码，也都需要对全部代码负责。同时，XP 强调代码是谁破坏的（修改后出现问题），就应该由谁来修复。

希赛教育专家提示：集体代码所有制是 XP 与其他敏捷方法的一个较大不同，也是从另一个侧面体现了 XP 中蕴含的很深厚的编码情节。

（9）持续集成。在前面谈到小型发布、重构、结对编程、集体代码所有制等最佳实践的时候，多次提到“持续集成”，可以说持续集成是这些最佳实践的基本支撑条件。

（10）每周工作 40 小时。这是最让开发人员开心、管理者反对的一个最佳实践了，加班、再加班早已成为开发人员的家常便饭，也是管理者最常使用的一种策略。而 XP 方法认为，加班最终会扼杀团队的积极性，最终导致项目的失败，这也充分体现了 XP 方法关注人的因素比关注过程的因素更多一些。不过，有一点是需要解释的，“每周工作 40 小时”中的“40”不是一个绝对数，它所代表的意思是团队应该保证按照“正常的时间” 进行工作。

（11）现场客户。为了保证开发出来的结果与客户的预想接近，XP 方法认为最重要的是需要将客户请到开发现场。就像计划游戏中提到过的，在 XP 项目中，应该时刻保证客户负责业务决策，开发团队负责技术决策。因此，在项目中有客户在现场明确用户故事，并做出相应的业务决策，对于 XP 项目而言有着十分重要的意义。

（12）编码标准。拥有编码标准可以避免团队在一些与开发进度无关的细枝末节问题上发生争论，而且会给重构、结对编程带来很大的麻烦。不过，XP 方法的编码标准的目的不

是创建一个事无巨细的规则列表，而是要能够提供一个确保代码清晰，便于交流的指导方针。

有句经典名言“1+1>2”最适合表达 XP 的观点，Kent Beck 认为，XP 方法的最大价值在于，在项目中融会贯通地运用这 12 个最佳实践，而非单独使用。当然，可以使用其中的一些实践，但这并不意味着就应用了 XP 方法。XP 方法真正能够发挥其效能，就必须完整地运用 12 个实践。

6.4.2 特征驱动开发

FDD 方法来自于一个大型的新加坡银行项目。FDD 的创立者 Jeff De Luca 和 Peter Coad 分别是这个项目的项目经理和首席架构设计师。在 Jeff 和 Peter 接手项目时，客户已经经历了一次项目的失败，从用户到高层都对这个项目持怀疑的态度，项目组士气低落。随后， Jeff 和 Peter 采用了特征驱动、彩色建模等方法，最终获得了巨大成功。

FDD 是也是一个迭代的开发模型。FDD 的每一步都强调质量，不断地交付可运行的软件，并以很小的开发提供精确的项目进度报告和状态信息。同敏捷方法一样，FDD 弱化了过程在软件开发中的地位。虽然 FDD 中也定义了开发的过程，不过一个几页纸就能完全描述的过程深受开发者的喜爱。

1．FDD 角色定义

FDD 认为，有效的软件开发不可缺少的三个要素是：人、过程和技术。软件开发不能没有过程，也不能没有技术，但软件开发中最重要的是人。个人的生产率和人的技能将会决定项目的成败。为了让项目团队能够紧密地工作在一起，FDD 定义了 6 种关键的项目角色：

（1）项目经理。项目经理是开发的组织者，但项目经理不是开发的主宰。对于项目团队来说，项目经理应该是团队的保护屏障。他将同团队外界（如高层领导、人事甚至写字楼的物业管理员）进行沟通，努力为团队提供一个适宜的开发环境。

（2）首席架构设计师。不难理解，首席架构设计师负责系统架构的设计。

（3）开发经理。开发经理负责团队日常的开发，解决开发中出现的技术问题与资源冲突。

（4）主程序员。主程序员将带领一个小组完成特征的详细设计和构建的工作，一般要求主程序员具有一定的工作经验，并能够带动小组的工作。

（5）程序员。若干个程序员在主程序员的带领下形成一个开发小组，按照特征开发计划完成开发。

（6）领域专家。领域专家是对业务领域精通的人，一般由客户、系统分析员等担当。领域专家作为关键的项目角色正是敏捷宣言中“业务人员同开发人员紧密合作”的体现。

根据项目规模的大小，有些角色是可以重复的。例如在一个小规模项目中，项目经理自身的能力很强，他就可以同时担当项目经理、首席架构设计师和开发经理的角色。

2．核心过程

FDD 共有 5 个核心过程，如图 6-6 所示。

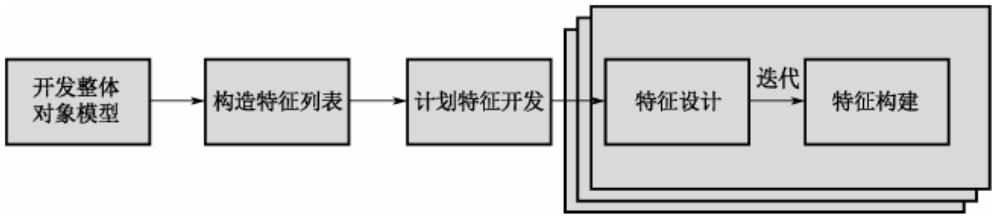


图 6-6　FDD 的核心过程

（1）开发整体对象模型。开发整体对象模型也就是业务建模的阶段。不过 FDD 强调的是系统地完整地面向对象建模，这种做法有助于把握整个系统，而不仅仅关注系统中的若干个点。在这一阶段，领域专家和首席架构设计师相互配合，完成整体对象模型。

（2）构造特征列表。完成系统建模后，需要构造一个完整的特征列表。所谓特征指的是一个小的、对客户有价值的功能。采用动作、结果和目标来描述特征，特征的粒度最好可以在两周之内实现。在这一阶段中，可以整理出系统的需求。

（3）计划特征开发。很少看到有哪个软件在开发过程中明确包含计划过程，其实任何一个软件项目都必须有计划——无论是重载方法还是敏捷方法。在这一阶段中，项目经理根据构造出的特征列表、特征间的依赖关系进行计划，安排开发任务。

（4）特征设计。在这一阶段，主程序员将带领特征小组对特征进行详细设计，为后面的构建做准备。

（5）特征构建。特征构建和特征设计这两个阶段合并起来可以看做特征的实现阶段，这两个阶段反复地迭代，直到完成全部的开发。

3．最佳实践

组成 FDD 的最佳实践包括：领域对象建模、根据特征进行开发、类的个体所有、组成特征小组、审查、定期构造、配置管理、结果的可见性。

其中，最有特色的莫过于类的个体所有。几乎所有的开发模型都是代码共有，程序员们负责开发系统中的全部代码，并通过配置管理和变更控制来保持代码的一致性。在 FDD 中，将类分配给特定的任何小组，分配给 A 成员的代码将全部由 A 来维护，除 A 外的角色都

不能修改它，只能使用它。这样做当然有它的优点：个人对所分配的类很容易保持概念的完整性；开发类代码的人肯定是最熟悉这个类的主人；而对这个类的支配感会促使开发人员产生自豪感，从而更出色地完成任务。不过 FDD 也提到了类个体所有的缺陷：项目中的依赖关系增强、当 A 需要 B 修改他自己的类时，必须等待 B 完成修改才能使用；类的个体所有增加了员工离职的损失。面对这些优点和缺陷，显然 FDD 认为类的个体所有对系统开发更有帮助。

除类的个体所有外，审查也是 FDD 中很具特色的一项实践。不少人都认为审查是非常严格的软件过程所特有的，因为进行审查不但要花费不少的人力和时间，对审查者本身的素质也有要求。然而在 FDD 中，明确地将审查作为一项最佳实践提出。审查是一种很有效的发现缺陷的手段，但经常被忽视，国内的软件组织中很少有严格审查制度保证软件质量。有效的审查可以发现很多潜在的问题，而这些问题往往是无法通过测试发现的，例如建模、需求和设计期的缺陷。这些潜在的缺陷大多要到系统测试甚至发布后才能发现，修正这些缺陷的代价是很大的。

6.4.3 Scrum

Scrum 是一个用于开发和维持复杂产品的框架，是一个增量的、迭代的开发过程。在这个框架中，整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个 Sprint，每个 Sprint 的建议长度是 2 到 4 周(互联网产品研发可以使用 1 周的 Sprint)。在 Scrum 中，使用产品 Backlog 来管理产品的需求，产品 Backlog 是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。Scrum 团队总是先开发对客户具有较高价值的需求。在 Sprint 中，Scrum 团队从产品 Backlog 中挑选最高优先级的需求进行开发。挑选的需求在 Sprint 计划会议上经过讨论、分析和估算得到相应的任务列表，我们称它为 Sprint backlog。在每个迭代结束时，Scrum 团队将递交潜在可交付的产品增量。 Scrum 起源于软件开发项目，但它适用于任何复杂的或是创新性的项目。Scrum 的基本流程如图 6-7 所示。

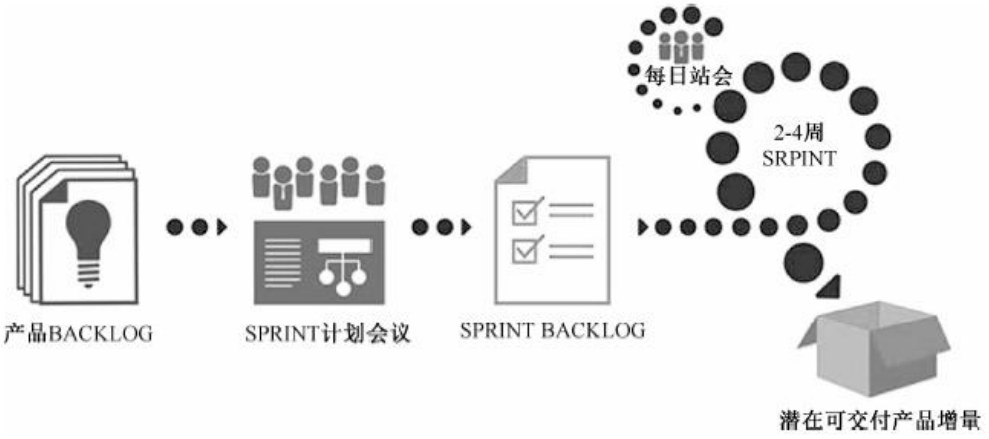


图 6-7　Scrum 流程图

1．Scrum 的五个活动

Scrum 主要包括：产品待办事项列表梳理、Sprint 计划会议、每日 Scrum 会议、Sprint 评审会议、Sprint 回顾会议等五个活动。

（1）产品待办事项列表梳理

产品待办事项通常会很大，也很宽泛，而且想法会变来变去、优先级也会变化，所以产品待办事项列表梳理是一个始终贯穿整个 Scrum 项目的活动。该活动包含但不限于以下的内容：保持产品待办事项列表有序、把看起来不再重要的事项移除或者降级、增加或提升涌现出来的或变得更重要的事项、将事项分解成更小的事项、将事项归并为更大的事项、对事项进行估算。

产品待办事项列表梳理的一个最大好处是为即将到来的几个 Sprint 做准备。为此，梳理时会特别关注那些即将被实现的事项。需要考虑不少因素，这包括但不限于以下的内容：

理想情况下，下一个 Sprint 的备选事项都应该提升“商业价值”。开发团队需要能够在一个 Sprint 内完成每一个事项。每个人都需要清楚预期产出是什么。

产品开发决定了，有可能需要其他的技能和输入。因此，产品待办事项列表梳理最好是所有团队成员都参与的活动，而不单单是产品负责人。

（2）Sprint 计划会议

每个 Sprint 都以 Sprint 计划会议作为开始，这是一个固定时长的会议，在这个会议中，Scrum 团队共同选择和理解在即将到来的 Sprint 中要完成的工作。

整个团队都要参加 Sprint 计划会议。针对排好序的产品待办事项列表（Product Backlog），产品负责人和开发团队成员讨论每个事项，并对该事项达成共识，包括根据当前的“完成的定义”，为了完成该事项所需要完成的所有事情。所有的 Scrum 会议都是限定时长的。Sprint

计划会议推荐时长是 Sprint 中的每周对应两小时或者更少（例如，一个 Sprint 包含 2 个星期，则 Sprint 计划会议时长应为 4 个小时或者更少）。因为会议是限制时长的,Sprint 计划会议的成功十分依赖于产品待办事项列表的质量。这就是产品待办事项列表梳理十分重要的原因。

在 Scrum 中，Sprint 计划会议有两部分：

 决定在 Sprint 中需要完成哪些工作

 决定这些工作如何完成

第一部分：需要完成哪些工作?

在会议的第一部分，产品负责人向开发团队介绍排好序的产品待办事项，整个 Scrum 团队共同理解这些工作。

Sprint 中需要完成的产品待办事项数目完全由开发团队决定。为了决定做多少，开发团队需要考虑当前产品增量的状态，团队过去的工作情况，团队当前的生产能力，以及排好序的产品待办事项列表。做多少工作只能由开发团队决定。产品负责人或任何其他人，都不能给开发团队强加更多的工作量。

通常 Sprint 都有个目标，称作 Sprint 目标。这将十分有效地帮助大家更加专注于需要完成的工作的本质，而不必花太多精力去关注那些对于我们需要完成的工作并不重要的小细节。

第二部分：如何完成工作？

在会议的第二部分里，开发团队需要根据当前的“完成的定义”一起决定如何实现下一个产品增量。他们进行足够的设计和计划，从而有信心可以在 Sprint 中完成所有工作。前几天的工作会被分解成小的单元，每个工作单元不超过一天。之后要完成的工作可以稍大些，以后再对它们进行分解。

决定如何完成工作是开发团队的职责，决定做什么则是产品负责人的职责。在计划会议的第二部分，产品负责人可以继续留下来回答问题，以及澄清一些误解。

不管怎样，团队应该很容易找到产品负责人。

Sprint 计划会议的产出。Sprint 计划会议最终需要 Scrum 团队对 Sprint 需要完成工作的数量和复杂度达成共识，并预期在一个合理的条件范围内完成它们。开发团队预测并共同承诺他们要完成的工作量。总而言之：在 Sprint 计划会议中，开发团队和产品负责人一起考虑并讨论产品待办事项，确保他们对这些事项的理解，选择一些他们预测能完成的事项，创建足够详细的计划来确保他们能够完成这些事项。

最终产生的待办事项列表就是“Sprint 待办事项列表（Sprint Backlog）”。

（3）每日 Scrum 会议

开发团队是自组织的。开发团队通过每日 Scrum 会议来确认他们仍然可以实现 Sprint 的目标。这个会议每天在同样的时间和同样的地点召开。每一个开发团队成员需要提供以下三点信息：

从上一个每日 Scrum 到现在，我完成了什么；从现在到下一个每日 Scrum，我计划完成什么；有什么阻碍了我的进展。

每日 Scrum 中可能有简要的问题澄清和回答，但是不应该有任何话题的讨论。通常，许多团队会在每日 Scrum 之后马上开会处理他们遇到的任何问题。

每日 Scrum 既不是向管理层汇报，也不是向产品负责人或者 ScrumMaster 汇报。它是一个开发团队内部的沟通会议，来保证他们对现状有一致的了解。只有 Scrum 团队的成员，包括 ScrumMaster 和产品负责人，可以在会议中发言。其他感兴趣的人可以来旁听。在必要时，开发团队会基于会议中的发现重新组织他们的工作来完成 Sprint 的目标。

每日 Scrum 是 Scrum 的一个关键组成部分，它可以带来透明性,信任和更好的绩效。它能帮助快速发现问题，并促进团队的自组织和自立。所有 Scrum 会议都是限定时长的。每日 Scrum 通常不超过 15 分钟。

（4）Sprint 评审会议

Sprint 结束时，Scrum 团队和相关人员一起评审 Sprint 的产出。Sprint 评审会议的推荐时长是 Sprint 中的每一周对应一个小时（例如，一个 Sprint 包含 2 个星期，则 Sprint 评审会议时长为 2 个小时）。

讨论围绕着 Sprint 中完成的产品增量。由于 Sprint 的产出会涉及一些人的“利益”，因此一个明智的做法是邀请他们参加这个会议，这会很有帮助。这个会议是个非正式的会议，帮助大家了解我们目前进展到哪里，并一起讨论我们下一步如何推进。每个人都可以在 Sprint 评审会议上发表意见。当然，产品负责人会对未来做出最终的决定，并适当地调整产品待办事项列表 Product Backlog。

团队会找到他们自己的方式来开 Sprint 评审会议。通常会演示产品增量,整个小组也会经常讨论他们在 Sprint 中观察到了什么、有哪些新的产品想法出现。他们还会讨论产品待办事项列表的状态、可能的完成日期以及在这些日期前能完成什么。

Sprint 评审会议向每个人展示了当前产品增量的概况。因此，通常都会在 Sprint 评审会议中调整产品待办事项列表。

（5）Sprint 回顾会议

在每个 Sprint 结束后,Scrum 团队会聚在一起开 Sprint 回顾会议,目的是回顾一下团队在流程人际关系以及工具方面做得如何。团队识别出哪些做得好,哪些做得不好,并找出潜在的改进事项,为将来的改进制定计划。Sprint 回顾会议的推荐时长是 Sprint 中的每一周对应一个小时（例如，一个 Sprint 包含 2 个星期，则 Sprint 回顾会议时长为 2 个小时）。

Scrum 团队总是在 Scrum 的框架内，改进他们自己的流程。

2．Scrum 的 5 大价值观

Scrum 的 5 大价值观为：

承诺—愿意对目标做出承诺。

专注—把你的心思和能力都用到你承诺的工作上去。

开放—Scrum 把项目中的一切开放给每个人看。

尊重—每个人都有他独特的背景和经验。

勇气—有勇气做出承诺，履行承诺，接受别人的尊重。

6.4.4 水晶方法

水晶方法（Crystal），是由 Alistair Cockburn 和 Jim Highsmith 建立的敏捷方法系列，其目的是发展一种提倡“机动性的”方法，包含具有共性的核心元素，每个都含有独特的角色、过程模式、工作产品和实践。Crystal 家族实际上是一组经过证明、对不同类型项目非常有效的敏捷过程，它的发明使得敏捷团队可以根据其项目和环境选择最合适的 Crystal 家族成员（分为 Crystal Clear，Crystal Yellow，Crystal Orange 和 Crystal Red 分别适用于不同的项目）。水晶方法中，使用频度较高的是 Crystal Clear——透明水晶方法。透明水晶方法，适合于一个小团队来进行敏捷开发，人数在 6 人以下为宜。

透明水晶方法有七大体系特征：

（1）经常交付

任何项目，无论大小、敏捷程度，其最重要的一项体系特征是每过几个月就向用户交付已测试的运行代码。如果你使用了此体系特征，你就会发现，“经常交付”的作用还是很让人吃惊的。

项目主办者根据团队的工作进展获得重要反馈。用户有机会发现他们原来的需求是否是他们真正想要的，也有机会将观察结果反馈到开发当中。开发人员打破未决问题的死结，从

而实现对重点的持续关注。团队得以调整开发和配置的过程，并通过完成这些工作鼓舞团队的士气。

（2）反思改进

在我们的开发中，时常会出现这样那样的问题，技术难题、各种烦心事等，这会在很大的程度上影响项目的进展。而且，如果其他任务对这项任务有依赖的话，那么其他的任务也会被推迟，这就很可能会导致项目的失败。

换句话说，如果，我们能够经常在迭代会中及时地反思和改进，那么，这种事情应该是不会发生的，或者说发生了，也能够很快地找到解决方案去应对它。事实上，从慌乱的日常开发中，抽出一点时间来思考更为行之有效的工作方法就已经足够了。

（3）渗透式交流

渗透交流就是信息流向团队成员的背景听觉，使得成员就像通过渗透一样获取相关信息。这种交流通常都是通过团队成员在同一间工作室内工作而实现的。若其中一名成员提出问题，工作室内的其他成员可以选择关注或不关注的态度，可以加入到这个问题的讨论当中来，也可以继续忙自己的工作。

（4）个人安全

个人安全指的是当你指出困扰你的问题时，你不用担心受到报复。个人安全非常重要，有了它，团队可以发现和改正自身的缺点。没有它，团队成员们知而不言，缺点则愈发严重以至于损害整个团队。个人安全是迈向信任的第一步。有了信任，团队协作才能真正地实施，开发效率也就会直线上升的。

（5）焦点

所谓“焦点”，就是确定首先要做什么，然后安排时间，以平和的心态开展工作。确保团队成员清楚地了解他们自己最重要的任务是什么，确保他们能够有充分的时间去完成这些任务。

（6）与专家用户建立方便的联系

与专家用户持续建立方便的联系能够给团队提供：对经常交付进行配置以及测试的地方，关于成品质量的快速反馈，关于设计理念的快速反馈，最新的（用户）需求。

（7）配有自动测试、配置管理和经常集成功能的技术环境

自动测试可以为开发人员在代码修改后就可以进行自动测试，并且能够发现存在的一些bug，以至开发人员能够及时地进行修改，对于他们来说，节省了时间，提高了效率，而且还不用为烦人的测试而苦恼。

配置管理系统允许人们不同步地对工作进行检查，可撤销更改，并且可以将某一系统设置保存后进行新系统的发布，当新系统出现问题，即可还原原系统的设置。

经常集成可以使得团队在一天之内对系统进行多次集成。其实，团队越频繁地对系统进行集成，他们就能够越快地发现错误，堆积到一起的错误也会越少，并使他们产生更新的灵感。

最好的团队是将这三大技术结合成“持续测试集成技术”。这样做他们可以在几分钟内发现因集成所产生的错误。

6.4.5 其他敏捷方法

除了上面介绍的几种敏捷方法，以下敏捷方法我们也需要掌握其基本特征。开放式源码：开放式源码指的是开放源码界所用的一种运作方式。开放式源码项目有

一个特别之处，就是程序开发人员在地域上分布很广，这使得它和其他敏捷方法不同，因为一般的敏捷方法都强调项目组成员在同一地点工作。开放源码的一个突出特点就是查错排障（debug）的高度并行性，任何人发现了错误都可将改正源码的“补丁”文件发给维护者。然后由维护者将这些“补丁”或是新增的代码并入源码库。ASD 方法：ASD (Adaptive Software Development)方法由 Jim Highsmith 提出，其核心是三个非线性的、重叠的开发阶段：猜测、合作与学习。

6.5 软件重用

软件重用技术是一种重要的软件开发方法，虽然至今软件重用技术还不够成熟，离理想中的软件工厂还有很长的路要走，但现有的一些重用技术（例如，中间件、应用服务器等）已经改变了开发过程。

6.5.1 软件重用

软件产品与其他的产品不同，是抽象的，一旦产生就可以无限制地复制，因此重复利用软件产品的意义重大，可以节约大量的人力物力。软件重用指的是利用已经存在的软件元素建立新的软件系统，这其中的软件元素既可以是软件产品、源程序，也可以是文档、设计思想甚至是领域知识。软件重用可以直接提高软件的开发效率、降低软件的开发成本、缩短软件的开发周期、提高软件质量。

常见的软件重用形式包括：

（1）源代码重用。这是最简单也是最常见的重用形式，但由于软件系统的复杂性，很难大规模地重用已有源代码。

（2）架构重用。架构重用也很常见，随着软件架构风格和设计模式的推广和应用，架构重用已经对软件开发产生了重大的影响。

（3）应用框架的重用。随着软件技术的发展，应用框架的重用变得越来越普遍，很多成熟的软件公司都建立了自己的开发框架。在开源社区中，世界各地的技术爱好者也在不断地推出应用了各种新技术的开发框架，例如，应用了 AOP（Aspect Oriented Programming，面向方面编程）技术的 Spring 等。

（4）业务建模的重用。虽然不同的软件的业务领域各自不同，但人们还是总结出了一些常见领域的建模方法，重用这些领域模型可以降低因领域知识不足而造成的需求风险。

（5）文档及过程的重用。软件文档和软件过程也是软件开发中不可或缺的元素，有效地重用这些文档和过程也有助于提高开发效率和软件质量、降低开发成本。

（6）软构件的重用。关于软构件的重用，请参考 5.5.2 节。

（7）软件服务的重用。随着 Web 服务的提出，人们越来越关注服务的重用。SOA（Service-Oriented Architecture，面向服务的架构）提出了面向服务的软件架构，并定义了相应的标准。但 SOA 还不够成熟，相信这一领域在未来的几年中还将取得更大的进展。

6.5.2 构件技术

构件又称为组件，是一个自包容、可复用的程序集。首先，构件是一个程序集，或者说是一组程序的集合。这个集合可能会以各种方式体现出来，如源程序或二进制的代码。这个集合整体向外提供统一的访问接口，构件外部只能通过接口来访问构件，而不能直接操作构件的内部。

构件的两个最重要的特性是自包容与可重用。自包容指的是构件的本身是一个功能完整的独立体，构件内部与外部的功能界限清晰明确，可以独立配置与使用。而可重用既是构件的特点，也是构件出现的目的。早在 1968 年 NATO 软件工程会议，McIlroy 的论文《大量生产的软件构件》中，就提出了“软件组装生产线”的思想。从那以后，使用构件技术实现软件复用，采用“搭积木”的方式生产软件，就成为软件人员的梦想。

构件的开发者和使用者往往不是相同的人或组织，所以必须定义构件的标准才能够消除

其中的障碍。随着构件技术的发展，目前应用比较广泛的构件标准有 CORBA、Java Bean/EJB、COM/DCOM。

应用构件技术开发软件可以使用构件组装模型，见 5.5.1 中的介绍。

6.6 基于架构的软件设计

基于架构的软件设计（Architecture-Based Software Design，ABSD）是一种架构驱动方法。这种方法有 3 个基础：

（1）功能的分解。在功能分解中，ABSD 方法使用已有的基于模块的内聚和耦合技术。

（2）通过选择架构风格来实现质量和业务需求。

（3）软件模板的使用。软件模板利用了一些软件系统的结构。

然而，对于设计方法来说，软件模板的使用是一个新概念，下面，我们进行简单的介绍。

软件模板是一个特殊类型的软件元素，包括描述所有这种类型的元素在共享服务和底层构造的基础上如何进行交互。软件模板还包括属于这种类型的所有元素的功能，这些功能的例子有：每个元素必须记录某些重大事件，每个元素必须为运行期间的外部诊断提供测试点等。在软件产品线系统中，软件模板显得格外重要，因为新元素的引入是一个通用的技术，这种技术用来使产品线架构适应一个特定的产品。

ABSD 方法是递归的，且迭代的每一个步骤都是清晰定义的。因此，不管设计是否完成，架构总是清晰的，这有助于降低架构设计的随意性。

6.6.1 ABSD 方法与生命周期

图 6-8 描述了 ABSD 方法在生命周期中的位置。尽管我们没有描述一个需求获取、组织或跟踪的特定方法，但还是假设一个需求阶段至少部分地完成，从需求阶段（包括功能需求、质量和业务需求、约束等）获得了输出。ABSD 方法的输出是三个视图的概念构件的集合，包括能够产生每个概念构件的假定、软件模板的集合和那些已经做出的具体实现的决策，我们把具体实现决策当作附加约束来维护。

在 ABSD 方法中，必须记录所有做出的决策及这些决策的原理，这有利于决策的可跟踪性和决策评审。

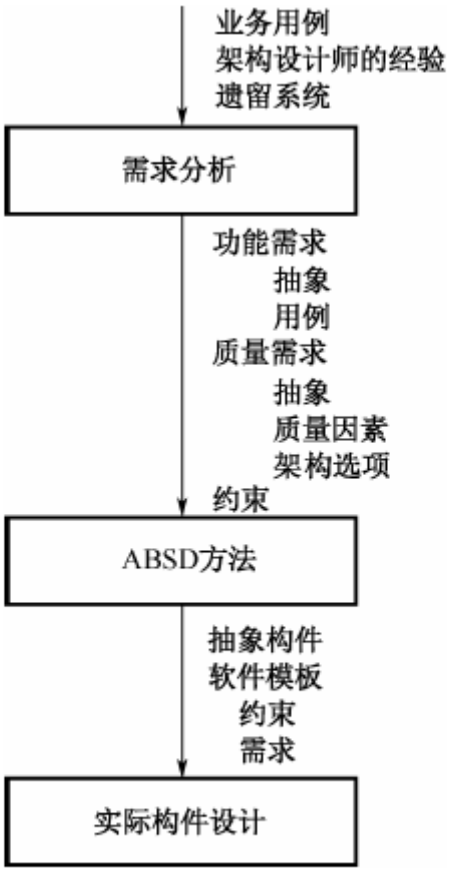


图 6-8　ABSD 方法与生命周期

ABSD 方法的输入由下列部分组成：

（1）抽象功能需求，包括变化的需求和通用的需求；

（2）用例（实际功能需求）；

（3）抽象的质量和业务需求；

（4）质量因素（实际质量和业务需求）；

（5）架构选项；

（6）约束。

下面，我们描述需求阶段的假定输出，即 ABSD 方法的输入。

1．抽象功能需求

ABSD 方法假定需求阶段的输出之一是功能需求的抽象描述，包括这些需求的粗略变化的描述。当获取需求时，考虑所有最终用户是重要的。

对一个特定系统来说，通常有不同类型的最终用户。不同的系统管理员（数据库管理员、系统管理员、网络管理员等）都可以是最终用户。维护工程师也可以是系统的最终用户。总之，一个最终用户就是当系统运行时使用系统的任何人员。

与抽象功能需求相联系的是对公共需求和与这些需求相关的粗略变化的描述，在设计阶段，理解这些需求之间的依赖关系是至关重要的。

我们必须在某种抽象级别上获取功能需求，产品的详细需求往往要等具体产品开发完成后才能知道。当详细需求明确时，抽象功能的获取为详细需求提供了分类。

2．用例

如前所述，用例是一个或多个最终用户与系统之间的交互的具体表述，在这里，最终用户既可以是操作人员，也可以是与系统进行交互操作的其他软件系统。虽然用例很容易找到和创建，甚至可能有成百上千个，但是，因为我们需要分析用例，所以必须限制用例的数量。在架构设计阶段，只有重要的用例才有用。我们必须对所创建的用例进行分组、设置优先级，以便筛选出最重要的用例，剩下的用例可以在设计阶段的任何时候创建。

3．抽象的质量和业务需求

我们必须对待构建系统的质量和业务需求进行编号，每个质量属性都包含一个特定的刺激，以及希望得到的响应。质量需求要尽量具体化。

4．架构选项

对每个质量和业务需求，我们都要列举能够满足该需求的所有可能的架构。例如，如果需求是支持一系列不同的用户界面，则可能的架构选择就是把不同的用户界面分解成不同的构件。又如，如果需求是保持操作系统的独立性，则可能的架构选择就是构建虚拟的操作系统层，接受所有的操作系统调用，并解释之为当前操作系统所能支持。

在这个时候，只需列举所有可能的选项，而不需要对这些架构选项进行决策，这种列举取决于设计师的经验，既可来自某些书籍介绍，也可直接来自设计师本身的实践。

5．质量场景

正如用例使功能需求具体化一样，质量场景使质量需求具体化。质量场景是质量需求的特定扩充。

与用例一样，质量场景也很容易找到和创建，可以创建很多个。我们必须对质量场景进行分组、设置优先级，只需验证最重要的质量场景。

6．约束

约束是一个前置的设计决策，设计过程本身包含决策。某些决策可以直接由业务目标导出而无须考虑对设计的影响。例如，如果一个公司在某个中间件产品上投入了大量资金，那么在产品的选择上就可以不必考虑其他决策。在需求获取阶段，约束主要来自系统的业务目标。

在某些特殊情况下，约束由遗留系统决定。今天，几乎没有软件系统不参考已有系统的，常见的情况是，新老系统同时并存，或者新系统替代老系统，但是必须尽可能重用老系统的功能。在设计阶段，虽然这些遗留系统处于被设计系统的外部，但设计师必须考虑遗留系统的特征。也就是说，在某种程度上，遗留系统影响着当前的设计，因此，理解遗留系统的结构和解决问题的技术都很重要。出于商业目的，可能要求重用遗留系统的构件，这种需求就变成了约束。

6.6.2 基于架构的软件开发模型

基于架构的软件开发模型（Architecture-Based Software Design Model，ABSDM）把整个基于架构的软件过程划分为架构需求、设计、文档化、复审、实现、演化等 6 个子过程，如图 6-9 所示。

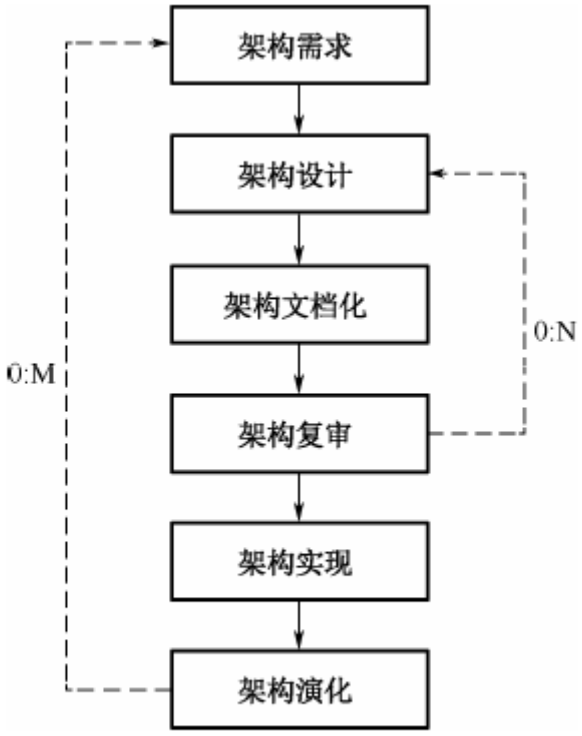


图 6-9　基于架构的软件开发模型

1．架构需求

需求是指用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。架构需求受技术环境和架构设计师的经验影响。需求过程主要是获取用户需求，标识系统中所要用到的构件。架构需求过程如图 6-10 所示。如果以前有类似的系统架构的需求，我们可以从需

求库中取出，加以利用和修改，以节省需求获取的时间，减少重复劳动，提高开发效率。

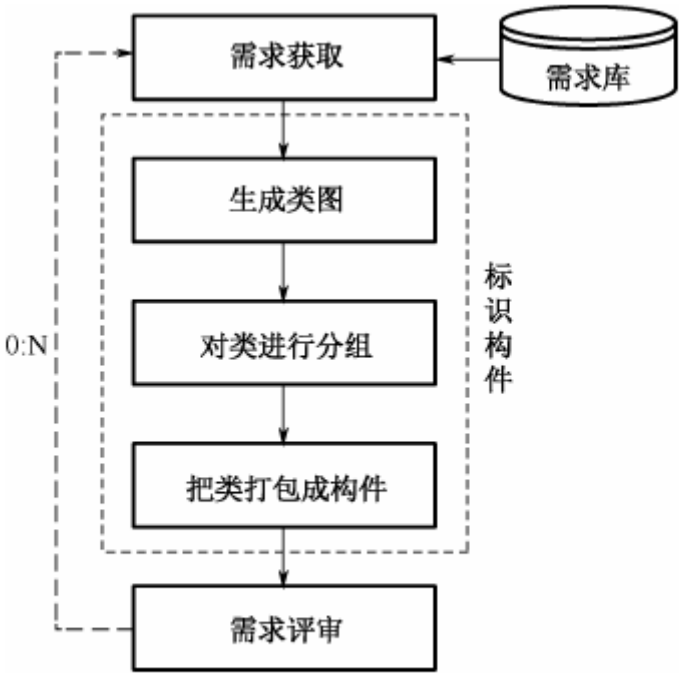


图 6-10　架构需求过程

（1）需求获取

架构需求一般来自三个方面，分别是系统的质量目标、系统的业务目标和系统开发人员的业务目标。软件架构需求获取过程主要是定义开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足业务上的功能需求。与此同时，还要获得软件质量属性，满足一些非功能需求。

（2）标识构件

在图 6-10 中虚框部分属于标识构件过程，该过程为系统生成初始逻辑结构，包含大致的构件。这一过程又可分为三步来实现。

第一步：生成类图。生成类图的 CASE 工具有很多，例如 Rational Rose 就能自动生成类图。

第二步：对类进行分组。在生成的类图基础上，使用一些标准对类进行分组可以大大简化类图结构，使之更清晰。一般地，与其他类隔离的类形成一个组，由泛化关联的类组成一个附加组，由聚合或组合关联的类也形成一个附加组。

第三步：把类打包成构件。把在第二步得到的类簇打包成构件，这些构件可以分组合并成更大的构件。

（3）需求评审

组织一个由不同代表（如分析人员、客户、设计人员、测试人员）组成的小组，对架构需求及相关构件进行仔细的审查。审查的主要内容包括所获取的需求是否真实反映了用户的要求，类的分组是否合理，构件合并是否合理等。

必要时，可以在“需求获取—标识构件—需求评审”之间进行迭代。

2．架构设计

架构需求用来激发和调整设计决策，不同的视图被用来表达与质量目标有关的信息。架构设计是一个迭代过程，如果要开发的系统能够从已有的系统中导出大部分，则可以使用已有系统的设计过程。软件架构设计过程如图 6-11 所示。

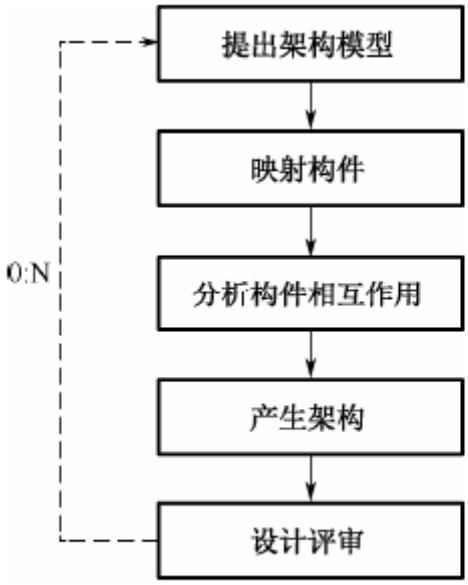


图 6-11 架构设计过程

（1）提出软件架构模型

在建立架构的初期，选择一个合适的架构风格是首要的。在这个风格基础上，开发人员通过架构模型，可以获得关于架构属性的理解。此时，虽然这个模型是理想化的（其中的某些部分可能错误地表示了应用的特征），但是，该模型为将来的实现和演化过程建立了目标。

（2）把已标识的构件映射到软件架构中

把在架构需求阶段已标识的构件映射到架构中，将产生一个中间结构，这个中间结构只包含那些能明确适合架构模型的构件。

（3）分析构件之间的相互作用

为了把所有已标识的构件集成到架构中，必须认真分析这些构件的相互作用和关系。

（4）产生软件架构

一旦决定了关键的构件之间的关系和相互作用，就可以在第 2 阶段得到的中间架构的基础上进行细化。

（5）设计评审

一旦设计了软件架构，我们必须邀请独立于系统开发的外部人员对架构进行评审。

3．架构文档化

绝大多数的架构都是抽象的，由一些概念上的构件组成。例如，层的概念在任何程序设计语言中都不存在。因此，要让系统分析师和程序员去实现架构，还必须得把架构进行文档化。文档是在系统演化的每一个阶段，系统设计与开发人员的通信媒介，是为验证架构设计和提炼或修改这些设计（必要时）所执行预先分析的基础。

架构文档化过程的主要输出结果是架构需求规格说明和测试架构需求的质量设计说明书这两个文档。生成需求模型构件的精确的形式化的描述，作为用户和开发者之间的一个协约。

软件架构的文档要求与软件开发项目中的其他文档是类似的。文档的完整性和质量是软件架构成功的关键因素。文档要从使用者的角度进行编写，必须分发给所有与系统有关的开发人员，且必须保证开发者手上的文档是最新的。

4．架构复审

从图 5-8 中我们可以看出，架构设计、文档化和复审是一个迭代过程。从这个方面来说，在一个主版本的软件架构分析之后，要安排一次由外部人员（用户代表和领域专家）参加的复审。

复审的目的是标识潜在的风险，以及早发现架构设计中的缺陷和错误，包括架构能否满足需求、质量需求是否在设计中得到体现、层次是否清晰、构件的划分是否合理、文档表达是否明确、构件的设计是否满足功能与性能的要求，等等。

由外部人员进行复审的目的是保证架构的设计能够公正地进行检验，使组织的管理者能够决定正式实现架构。

5．架构实现

所谓“实现”就是要用实体来显示出一个软件架构，即要符合架构所描述的结构性设计决策，分割成规定的构件，按规定方式互相交互。架构的实现过程如图 6-12 所示。

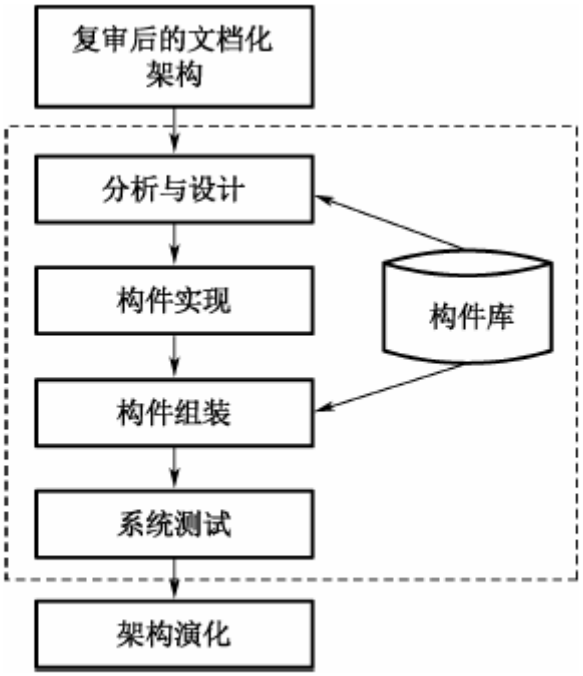


图 6-12　架构实现过程

图 6-12 中的虚框部分是架构的实现过程。整个实现过程是以复审后的文档化的架构说明书为基础的，每个构件必须满足软件架构中说明的对其他构件的责任。这些决定即实现的约束是在系统级或项目范围内做出的，每个构件上工作的实现者是看不见的。

在架构说明书中，已经定义了系统中构件与构件之间的关系。因为在架构层次上，构件接口约束对外唯一地代表了构件，所以可以从构件库中查找符合接口约束的构件，必要时开发新的满足要求的构件。

然后，按照设计提供的结构，通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成。

最后一步是测试，包括单个构件的功能性测试和被组装应用的整体功能和性能测试。

6．架构演化

在构件开发过程中，最终用户的需求可能还有变动。在软件开发完毕，正常运行后，由一个单位移植到另一个单位，需求也会发生变化。在这两种情况下，就必须相应地修改软件架构，以适应新的软件需求。架构演化过程如图 6-13 所示。架构演化是使用系统演化步骤去修改应用，以满足新的需求。主要包括以下七个步骤：

（1）需求变动归类首先必须对用户需求的变化进行归类，使变化的需求与已有构件对应。对找不到对应构件的变动，也要做好标记，在后续工作中，将创建新的构件，以对应这部分变化的需求。

（2）制订架构演化计划在改变原有结构之前，开发组织必须制订一个周密的架构演化计划，作为后续演化开发工作的指南。

（3）修改、增加或删除构件在演化计划的基础上，开发人员可根据在第一步得到的需求变动的归类情况，决定是否修改或删除存在的构件、增加新构件。最后，对修改和增加的构件进行功能性测试。

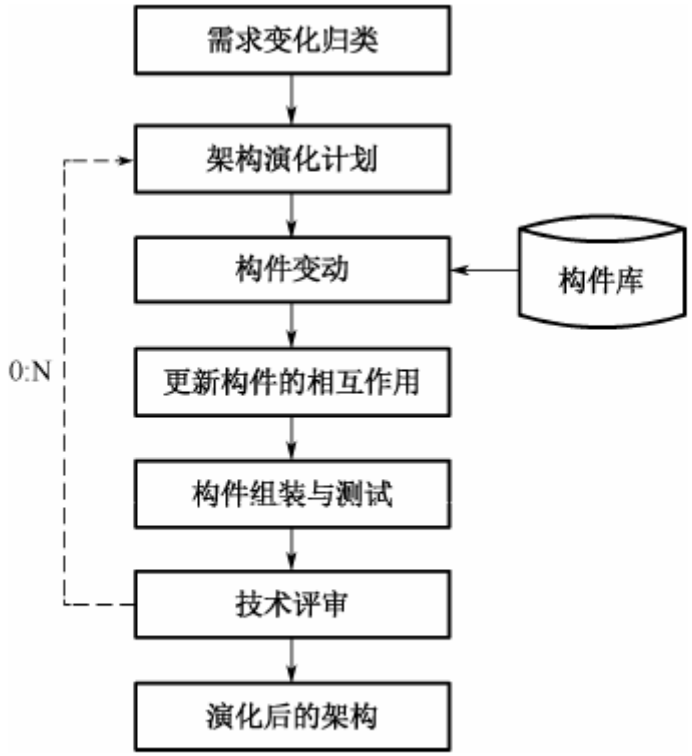


图 6-13　架构演化过程

（4）更新构件的相互作用随着构件的增加、删除和修改，构件之间的控制流必须得到更新。

（5）构件组装与测试通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成，形成新的架构。然后对组装后的系统的整体功能和性能进行测试。

（6）技术评审对以上步骤进行确认，进行技术评审。评审组装后的架构是否反映需求变动，符合用户需求。如果不符合，则需要在第 2 到第 6 步之间进行迭代。

（7）产生演化后的架构在原来系统上所作的所有修改必须集成到原来的架构中，完成一次演化过程。

6.7 形式化方法

形式化方法是指采用严格的数学方法，使用形式化规约语言来精确定义软件系统。非形式化的开发方法是通过自然语言、图形或表格描述软件系统的行为和特性，然后基于这些描述进行设计和开发，而形式化开发则是基于数学的方式描述、开发和验证系统。

形式化方法包括形式化描述和基于形式化描述的形式化验证两部分内容。形式化描述就是用形式化语言进行描绘，建立软件需求和特性，即解决软件“做什么”的问题。形式化验证指的是验证已有的程序是否满足形式化描述的定义。形式化描述主要可以分为两类，一类是通过建立计算模型来描述系统的行为特性，另一类则通过定义系统必须满足的一些属性来描述系统。形式化描述又称之为形式化规约，相对于自然语言描述，形式化描述是精确的、可验证的，避免了模糊性与二义性，消除需求中相互矛盾的地方，避免需求定义人员和开发人员对需求的理解偏差。

形式化描述可以通过计算机技术进行自动处理，进行一致性的检查和证明，提高需求分析的效率和质量。通过形式化描述，需求分析的质量大大提高，很多自然语言描述无法避免的缺陷在需求分析阶段就会被发现，并得到解决，从而降低后期开发和维护的成本，并提升软件的质量和可靠性。

在一些要求高可靠性的关键应用上，采用形式化开发方法可以保证软件系统的可靠性。如巴黎地铁 14 号线和 Roissy 机场穿梭车的自动控制系统。这两个系统中的部分程序使用了形式化方法进行开发，并取得了很好的效果，如表 6-1 所示。

表 6-1　形式化开发案例数据对比

项目	巴黎地铁 14 号线	Roissy 机场穿梭车
ADA 代码行数	86000	158000
交互证明所用人/月	7.1	4.6

表 6-1 中的 ADA 代码行数表示运用形式化方法开发的软件系统规模，这些代码是形式化方法自动生成的，开发人员并不需要直接修改这些代码。

希赛教育专家提示：这两个案例都没有进行单元测试，而在非形式化开发中，这类关键应用系统的软件的单元测试和集成测试都是非常重要的工作，通常要付出高昂的代价。形式化开发的优点可见一般。

第 7 章：系统规划

系统计划主要用于描述从项目提出、选择到确立的过程，包括系统项目的提出与可行性分析，系统方案的制订、评价和改进，新旧系统的分析和比较，以及现有软件、硬件和数据资源的有效利用等问题。

7.1 项目的提出与选择

组织在信息化的过程中，可能基于各种动机提出系统项目的建设，有关人员要根据这些动机，提出和确定信息系统的工作范围，确定项目立项，提出系统选择方案，给出选择结果。

7.1.1 项目的立项目标和动机

企事业单位在其自身的经营管理过程中，对于项目的立项建设可能具有多种动机，通常可归结为下列几种。

1．进行基础研究并获取技术

此类项目通常由大学院校或企业集团的战略研究性部门提出和实施。小规模的研究组织可能仅仅是企业中的一个研发部门或从事研发工作的团队；中大规模的研究组织包括研究所或研究院这种独立建制的单位；大规模的研究性项目可类似于国家 863 计划等跨行业、跨地域协作的国家级重大项目立项。

2．进行应用研发并获得产品

此类项目通常由企业进行立项和开发，企业立项的基本动机通常是为了得到应用软件产品并向目标客户群进行销售从而获取利润等。产品一般会基于某类特定客户群体的需求进行设计，有明确和具体的研发目标需求，有严格的时间限制、资源预算等，因此可归入 “应用研发”型软件。

应用研发型软件通常具有一定的通用性，客户广泛，既可能是面向个人消费者的工具软件（例如 Office、杀毒软件、游戏软件等），也可能是面向特定领域或行业的工具软件（例如 SQL Server 数据库、AutoCAD 工程绘图软件、Rational Rose 这样的建模工具软件等）。

3．提供技术服务

对此类项目进行立项的企业通常能向目标客户群提供比较全面的技术服务而不是单一的软件产品。因此企业的服务范围可能包含提供技术和解决方案的咨询、利用现有产品进行

系统集成和服务、面向特定客户的软件项目定制开发、对现有的软件系统进行升级和改造、提供软件应用相关的技术支持、服务和培训等服务中的一个或多个内容。

总的来说，此类组织通常会面向一个特定行业、具有相对稳定性的客户群体，通过提供一种综合性服务来获取市场价值，因此可以把此类公司看做“服务”导向的组织。

4．信息技术产品的使用者

信息技术的使用者是最终客户。对他们来说，软件项目的立项动机既不是为了得到软件产品而进行销售，也不是为了提供技术服务，而是通过购买产品或服务来得到使用价值。例如：一个消费者购买了绘图软件是为了存储和处理个人数码相机中的照片；而一个企业通过实施 ERP（Enterprise Resource Planning，企业资源计划）可能是为了达到生产能力的控制、生产计划科学性、提高管理水平、获取新的决策能力、降低库存成本、提高资金周转率、建立面向市场订单生产方式等目标，并期望通过这些目标的实现来增强企业竞争力、获取更大的市场份额。对信息技术的使用者来说，信息技术是一种手段，同时也是一种成本。如何用最小的成本和风险获得满意的效果是客户最关心的问题。

7.1.2 项目的选择和确定

系统项目的选择至少包含两种实用性目的，一个是软件开发公司在诸多的产品方向中选择适当的方向进行研究和开发，另一种是客户从诸多的产品中购买适合自己需要的产品或选择适合自己需要的技术方案进行实施。与系统项目提出的问题一样，并不存在一个统一模式进行系统项目的选择和取舍，但可以提出进行项目取舍和评估的若干原则。通过使用项目取舍和评估的原则，可以逐步排除那些不符合需求的项目定义，从而找到比较适合的项目或产品开发方向。

1．选择有核心价值的产品/项目或开发方向

这个策略关键在于确定什么样的系统项目是有价值的。由于立项单位所处的行业、在行业中的位置、立项目标等因素不同，对软件项目的价值判断也不同。但“有核心价值的软件项目”通常总是和企业或客户的核心业务相关的。

美国哈佛商学院的著名教授 Michael Porter 曾经在他的《竞争优势》（CompetitiveAdvantage）一书中提出了“价值链”的概念，价值链把企业运作的各种活动划分为产品设计、产品生产、产品营销和产品应用等独立领域，企业的价值链也可以进一步和上游供货商与下游买主的价值链相连，从而构成一个产业的价值链。如果以“价值链”的

观点来看待软件产品或项目，软件是作为一种技术服务手段被运用到企业业务的价值链上的，通过实现价值链中的关键业务的信息化从而最终改善客户单位的企业质量，同时也使软件开发公司获得现实的经济利益。

因此，在企业或客户经营活动中对价值链增值最大的部分，就是企业或客户的“核心业务”。针对核心业务的信息化产品或项目，通常都是具有高价值的，也可以说，所谓的 “行业信息化”的关键就是该行业中这些核心业务的信息化改造。例如：

（1）对生产制造业的企业来说，生产计划、库存控制、实现面向订单的生产就是核心业务，无论实施 ERP 还是小规模的 MIS 系统，针对这些部分的软件功能总是被客户认为是最有价值的。

（2）对于金融保险行业来说，由于保险公司的基本职责是分摊风险和补偿损失，所以一般要求保险公司有足够的分散风险的能力。因此，管理保单数据的业务系统、评估风险的定损系统等就是非常有价值的软件系统。

（3）对于教育行业来说，因为学校的核心职能是教书育人，因此与教研、教学、考试、评价等业务相关的软件系统，以及支持上述业务开展的教育资源库软件、电子图书馆软件等就是高价值的软件系统。

总之，选择软件项目，必须首先考察软件应用的行业、业务和目标，以便判明要建设的软件项目价值。

2．评估项目风险、收益和代价

在判断出一个潜在的软件项目后，还应评估项目实施的风险、收益和维护付出的代价。对于开发产品进行销售的情况，主要评估的是产品的预期收益和为完成开发投入的各种资源（包括时间、人力、资金等），项目的风险主要是技术难度、技术能力、经济能力和各种资源是否能承担、是否是企业需要优先实施的项目、是否符合行业标准和国家政策规定（例如：在电子签章没有经过国家法律许可之前，使用电子签章替代手工操作可能是有风险的）等。

对于购买产品或技术服务的客户来说，还应该评估项目实施后对自身业务变更，组织机构和人员职责的影响，现有的业务流程和人员的 IT 技能是否能满足要求，是否需制定相关的系统维护、运行规约和规章制度等。而项目实施的实际开销，除购买产品或服务的开支外，通常还包括各种系统维护、改进、培训，招聘新职员，变更业务流程等各种应用方面的开销。以总持有成本（Total Owner Cost，TOC）来评估信息化的代价才能比较准确地得到项目的实际代价。

评估项目风险、预期收益和代价后，可筛选掉多数不符合企业要求的建议项目。

3．评估项目的多种实施方式

对于已经确认有价值、并且有能力开发的软件项目，则可以进一步参照企业现状考察项目的实施方式。这种实施方式通常既包括了前面对项目风险、预期收益和资源开销的评估，也包含了企业对现阶段经营目标和现有资源如何合理运用的考虑。这个过程通常由项目的负责人和企业中高层经理进行决策，决策结果决定了项目的实施优先级及具体的实施方式。

需要说明的是，企业完成软件项目的方式并不单纯限制于自己组建开发团队进行软件项目或软件产品开发的策略。根据具体情况不同，还可能使用诸如转包开发业务给外部公司、直接 OEM（Original Equipment Manufacture，原始设备制造商）软件产品并进行系统集成、购买关键技术并进行“软件集成”方式的开发、完成技术方案和设计，然后寻求外部公司进行编码等各种方式。对这些项目实施方式的取舍，主要依据依然是对项目风险、收益和资源开销综合平衡的考虑。

4．平衡地选择适合的方案

人们在选择可行的方案时，总是希望得到高质量、低成本的产品和方案。软件开发人员通常也很愿意在产品开发中，向产品加入创造性的内容。另一方面，客户单位在面对诸多的投标方案时，会听到各种各样关于技术先进性、快速开发、产品质量稳定可靠、价格如何低廉、推荐的方案有多少成功应用等宣传。然而：

（1）新技术可能意味着未来更多的变化从而导致风险，也意味着未来产品的使用者需要更多的学习和导入期，而采用成熟的技术则可能享受不到新技术带来的好处。

（2）不基于某种快速开发技术或平台构造的产品可能会延长项目开发时间从而导致更多的开销，但基于某种平台的产品又可能使得用户未来“绑定”在某种平台之上，减少未来的自由选择性。

（3）不考虑系统的扩展性则很可能在业务变更时，会受阻于已经实施的 IT 设施，但过多考虑系统的扩展性，软件接口通常就需要花费较大的力气进行设计，那么用户是否在当前的购买中为一些自己并不需要的特性多支付成本？尤其在软件技术高速发展的今天，当用户期望进行系统升级的时候，常常会发现原来的计算体系已经早就被开发单位淘汰和抛弃。

（4）价格低廉的产品可能具有好的质量，也可能有些功能并不那么让人满意，而最重要的是，当关注这些具有先进性、低成本及拥有众多成功应用的产品或方案的时候，项目的选择者容易失去对自己目标的关注，即这些先进技术或宣传的产品特性是否确实是自己需要的？

事实上，对性能的要求常常是充满矛盾的，任何时候都不存在一个完美无缺的方案，只

存在一个对当前的项目目标相对比较适合的方案。项目的决策者必须从最终的项目目标出发，判明各种功能或性能的重要性和优先级。在抛弃明显存在问题的“差”项目后，选择项目的基本立场应该是“适合”，而不是尽可能的“好”。（实际上任何超出预期设定目标的“好”性能，通常都意味着更多的成本。）

更进一步地看，“适合”的方案就是平衡考虑开发单位利益和客户满意度的方案。

图 7-1 是 Noriaki Kano 提出的顾客质量模型图，要求质量是客户认为产品应该具备的功能或性能，实现越多客户会越满意；假想质量是客户想当然认为产品应具备的功能或性能，客户并不能正确描述自己想当然要得到的这些功能或性能需求；兴奋质量则是客户要求范围外的功能或性能（但通常是软件开发者很乐意赋予产品的技术特性），实现这些性 能客户会更高兴，但不实现也不影响其购买的决策。

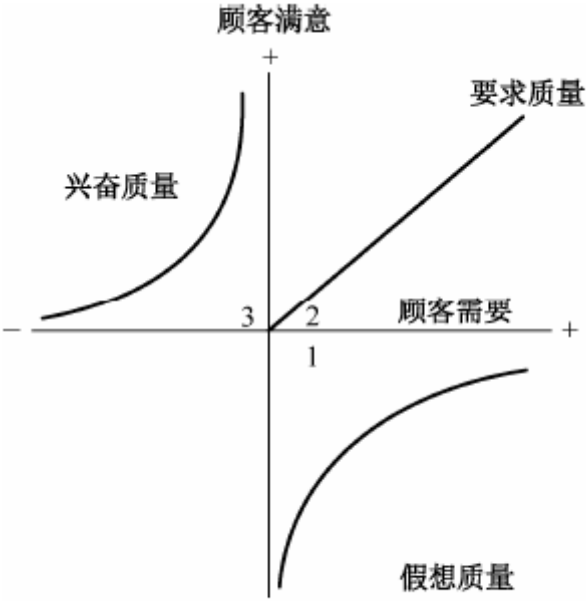


图 7-1　顾客质量模型图

显然，项目开发方更多考虑的是项目风险和回报。而客户更多关心的是成本和购买后的满意度。好的方案必须平衡考虑这些因素。系统分析师应尽可能用技术手段来平衡这些彼此对立的要求，保证在项目预期投入资源可接受的范围内，尽量实现客户要求质量对应的功能和性能、发掘客户假想质量对应的功能要求并进行沟通确认，但按自身所服务企业的经营目标平衡考虑客户兴奋质量的实现策略（是努力提供兴奋质量的功能、争取忠诚的客户获得远期潜在的收益，还是削减这些功能、以便使项目的成本最小化）。

希赛教育专家提示：系统设计师常犯的一个错误，就是用自己对技术的兴趣产生的兴奋质量，来替换客户最基本的要求质量和假想质量。而企业经营者常犯的错误，则可能是对客

户提出的合理要求质量视而不见；或者走向另一个极端，不加区分地把一切未经评估的假想要求质量不断指派给软件开发团队。这些都是错误的做法。

7.1.3 项目提出和选择的结果

系统项目提出和选择的结果，最终会以“产品/项目建议书”的方式来体现。典型的应用场景是：

（1）在投标项目中，产品/项目建议书通常是乙方提交给甲方竞标方案的一部分；

（2）企业单位在确立了要开发某类型产品后，对该产品进行多角度的评估，最终项目立项人向上级提交供决策的建议报告的主要内容就是“产品/项目建议书”。

产品/项目建议书是一个包含多种综合内容的报告，涉及的范围通常要比 GB 8567-1988 标准中规定的标准——“项目可行性分析报告”的内容更全面。在项目建议书中，可能包含如下几个部分：

用户单位、项目或产品的立项背景、需求来源和目标性的介绍；

用户的内外部环境、组织机构、现有的 IT 设施情况等；

用户的业务模型和业务规划；

预期要建设的技术系统在用户业务中的位置和作用；

信息化后的用户业务模型、软件应用方式、相关的部署环境、运行规则、管理规范等；

为实现信息化业务模型，技术系统的产品需求定义（功能、性能、约束）和部署方式等；

产品或项目的技术框架；

项目的要点、技术难点、主要实施障碍等；

项目或产品的可行性研究结果；

项目可选择的实施方式、组织方式、沟通和协调机制等；

项目的资源范围和预算（人、财、物、时间等）；

项目的成本/收益分析；

……

其他项目建议书可能包含的内容，或以单独文档列举的内容可能包括：

项目风险及影响评估；

项目进度计划；

项目质量计划；

项目过渡期资金的获得方式、财务计划；

产品或项目的商务模式、盈利模式论述；

同类产品或公司的市场调查结果，以及竞争性比较；

企业成功案例、资质等；

商务条款或供应商/客户合同；

……

项目建议书标志着项目立项和选择阶段性工作的完成，一旦项目建议书被批准通过，项目即可进入正式的开发准备和实施阶段。

7.2 可行性研究与效益分析

在项目计划和选择的过程中，需要完成的首要目标是对项目进行估算。项目估算的范围涉及方方面面，例如项目或产品开发的范围、投入和回报、项目风险、作用和意义等。在传统软件工程方法中，是以可行性研究的方式来组织项目的主要估算内容。

可行性研究的范围可能覆盖技术、经济、执行、环境等各种需要评估的因素，但它并不是最后的详细计划（例如：项目的时间进度及人员安排）。通常在进行可行性研究的阶段，项目的目标或产品的最终方向也是极易变化的。

但可行性研究的意义在于，虽然可行性研究不能指出项目最终的详细计划和方向，但可行性研究可以在项目定义阶段用较小的代价识别出错误构思的系统，从而规避未来更多的资源投入的损失（时间、资金、人力、机会），或者因遭遇到无法逾越的技术障碍或环境障碍导致的不可避免的失败。

对于那些可行性研究表明可执行的软件项目来说，可行性研究的结果也不承诺系统的收益一定很大或技术风险和资源投入就一定很低，但可行性研究的结果设立了一个“底线”，即如果做什么，风险和收益是什么样的控制范围。这些评估结果给了未来的项目评估、项目风险控制，甚至在资源剧烈变化的情况下有计划有重点地削减功能、重定义项目开发范围，提供了非常有价值的方向性指引

7.2.1 可行性研究的内容

可行性研究的主要内容包括经济可行性、技术可行性、法律可行性、执行可行性和方案的选择 5 个部分。

1．经济可行性

经济可行性主要评估项目的开发成本及项目成功后可能获得的经济收益。多数项目只有开发成本能控制在企业可接受的范围内的时候，项目才有可能被批准执行。而经济收益的考虑则非常广泛，例如：项目技术开发的直接现金收入、新产品在生命周期中预期的总销售收入、技术积累、对公司业务和产品线的完善和支持、开辟新市场和利润增长点、进入预期能带来较高收益的新市场、提高客户满意度和忠诚度、打击竞争对手抢夺市场份额、获得新的信息化能力从而改善经营或管理格局等。

2．技术可行性

技术可行性评估对于假想的软件系统需要实现的功能和性能，以及技术能力约束。技术可行性分析可通过“提问—回答”的方式来进行论证，包括：

（1）技术：现有的技术能力和 IT 技术的发展现状足以支持想象中的系统目标实现吗？

（2）资源：现有的资源（掌握技术的职员、公司的技术积累、构件库、软硬件条件等）足以支持项目实施吗？技术风险在评估的哪个范围内？

（3）目标：在目前设定的系统目标中，哪些目标会遭遇到较强的技术障碍？尤其是那些被设定为必须实现的系统目标。

由于在可行性研究阶段，项目的目标是比较模糊的，因此技术可行性最好与项目功能、性能和约束的定义同时进行。在可行性研究阶段，调整开发目标和选择可行的技术体系均是可用的手段，而一旦项目进入开发阶段，任何调整都意味着更多的开销。

需要再次指出的是，技术可行性绝不仅仅是论证在技术上是否可实现，实际上还包含了在当前资源条件下的技术可行性。

投资不足、时间不足、预设的开发目标技术难度过大、没有足够的技术积累、没有熟练的职员可用、没有足够的合作公司和外包资源积累等均是技术可行性的约束。软件系统的技术评估者通常都只考虑技术手段是否能实现而忽视了当前的资源条件和环境，从而对技术可行性研究得出了过于乐观的结果，这种错误判断对后期的项目实施会导致灾难性的后果！

加强前期的项目调研、寻求专家的咨询以及采用具有大量成功应用案例、被广泛支持的技术标准和事实标准等均有助于改善项目的技术可行性。

3．法律可行性

法律可行性评估可能由系统开发引发的侵权或法律责任，可能包括合同的订立和条款，职责、侵权情况的设定，违约、争议的解决等方方面面的内容。法律可行性还包括国家政策和法律的限制，例如：在政府信息化的领域中使用未被认可的加密算法或未经许可在产品中

使用了其他公司被保护的软件技术、构件等。

4．执行可行性

执行可行性也称操作可行性，它主要评估预期的软件系统在真实环境中能够被应用的程度和实施过程中障碍。例如：ERP 系统建成后的数据采集和数据质量问题，或客户工作人员没有足够的 IT 技能等。这些问题虽然与软件系统本身无关，但如果不经评估，很可能会导致投入巨资建成的软件系统毫无用处。

执行可行性还需要评估对用户的各种影响，包括对现有 IT 设施的影响、对用户组织机构的影响、对现有业务流程的影响、对地点的影响、对经费开支的影响等。如果某项影响会过多改变客户的现状，需要将这些因素作进一步的讨论并和软件系统的使用者进行沟通，提出建议的解决方法。

5．方案的选择

评估系统或产品开发的可选方法。一般来说，同样的项目，可以采用不同的方法来实现。甚至一个大项目的若干个子系统的实现方法也不一样。如何进行系统分解、如何定义各子系统的功能、性能和界面，实现方案不唯一。可以采用折中的方法，反复比较各个方案的成本和效益，选择可行的方案。

7.2.2 成本效益分析

效益分析实际上包含了“成本—收益”的分析。从内容上来看，效益分析是可以包含在可行性研究的经济可行性分析中的。但效益分析的目的在于，对项目开发目标的成本及可度量的项目现金收入和无形收益进行一次专门化的评估。这种以经济回报为收益的评估结果，是得到企业管理、决策层批准项目实施的重要因素。

效益分析中的成本分析，将尽可能地列举所有项目涉及的直接财务支出数字，以便管理层协调和制订各种资源的支出计划。效益分析中的收益分析，将尽可能清晰地列举实施项目带来的各种直接经济收益和无形收益，以便管理层理解项目的价值和给予项目资源上的支持。否则，一旦项目所需要的各种资源不能按计划投入，项目失败的风险将大大增加，并且除了变更项目预设的开发目标外，几乎没有可供选择的应急方案。

1．项目可能涉及的成本项目的成本部分，通常包括：

基础建设支出：如房屋和设施，办公设备，平台软件，必需的工具软件等购置费用；

一次性支出：如研究咨询费用、调研费、管理费用、培训费、差旅费、其他一次性杂费等；

运行维护费用：如设备租金和定期维护费用、定期消耗品支出、通信费、人员工资奖金、房屋租金、公共设施维护及其他经常性的支出项目。

2．项目可能涉及的收益

项目的收益，通常可以分为一次性收益、非一次性收益和不可定量的收益三个部分。

（1）一次性收益

开支的缩减。包括改进了的系统的运行所引起的开支缩减，如资源要求的减少，运行效率的改进，数据进入、存储和恢复技术的改进，系统性能的可监控，软件的转换和优化，数据压缩技术的采用，处理的集中化和分布化等；

价值的增升。包括由于一个应用系统的使用价值的增升所引起的收益，如资源利用的改进，管理和运行效率的改进及出错率的减少等；

其他。如从多余设备出售回收的收入等。

（2）非一次性收益

在整个系统生命期内由于运行所建议系统而导致的按月的、按年的能用人民币表示的收益，包括开支的减少。

（3）不可定量的收益

无法直接用人民币表示的收益，如服务的改进，由操作失误引起的风险的减少，信息掌握情况的改进，组织机构给外界形象的改善等。有些不可捉摸的收益只能大概估计或进行极值估计（按最好和最差情况估计）。

3．效益分析的若干指标和进一步的分析

收益/投资比：软件项目实施后整个系统生命期的收益/投资比值；

投资回收周期：收益的累计数开始超过支出的累计数的时间；

敏感性分析：分析项目中的一些关键性因素如系统生命期长度、系统的工作负荷量、工作负荷的类型、处理速度、设备和软件的配置等因素发生变化或进行合理搭配时，对开支和收益的影响最灵敏的范围估计。通常当项目需要在不同因素之间取舍和调整的时候，需要参考敏感性分析的内容。

7.2.3 可行性分析报告

在国家标准 GB 8567-1988 中，规定了可行性分析报告的详细格式和内容。这个规范文本基本上涵盖了可行性分析需要考察的问题，可作为书写可行性研究报告的参考文档模板。

不管可行性报告的形式如何，最重要的内容应当有以下几项。

项目背景：包括问题描述、实现环境和限制条件；

管理概要和建议：包括重要的研究结果、说明、建议和影响；

候选方案：包括候选系统的配置和最终方案的选择标准；

系统描述：包括系统工作范围的简要说明和被分配系统元素的可行性；

经济可行性（成本/效益分析）：包括经费概算和预期的经济效益；

技术可行性（技术风险评价）：包括技术实力、已有工作基础和设备条件；

法律可行性：包括系统开发可能导致的侵权，违法和责任等；

用户使用可行性：包括用户单位的行政管理，工作制度和使用人员的素质；

其他与项目有关的问题：例如，其他方案介绍和未来可能的变化。

可行性研究报告首先由项目负责人审查（审查内容是否可靠），再上报给上级主管审阅（评估项目的地位）。从可行性研究报告中应当得出“行或不行”的决断。

7.3 方案的制订和改进

通过在问题定义和归结模型阶段的工作，已经分析并定义了与系统开发目标相关的各种模型、分析出了系统的功能清单、性能要求等，解释了“系统目标是什么”的问题。在系统方案阶段，主要完成的工作则是解释“系统如何实现”的问题。

系统方案制订的最主要内容，包括以下几个方面。

1．确定软件架构

在问题定义阶段得到的软件概念模型使用各种工具定义了项目的开发目标。在系统方案制订阶段才开始真正考虑如何去实现软件。其中最重要的工作，就是制定系统的实现架构。

系统的实现架构与一些很具体的方面相关：

（1）分析模型的结构。例如，采用结构化分析方法得到的功能分解体系，或面向对象的类和“对象－关系图”、“对象－行为图”。

（2）一些对应于系统目标的最基本、最重要的实现要素。例如，关键的用例、最主要的控制类、对象组织的模式、常用和最关键的实现算法模型等。这些实现要素对应于系统目标实现最重要的场景，表示了整个系统最主要的控制流程和实现机制。

（3）特性和要点的解释。这些附加的内容解释系统的一些特性、服务等是如何实现的。

2．确定实现的各种关键性要素和实现手段关键性的实现要素通常包括：

关键的用例、最主要的控制类、功能和服务的首要组织方式（例如网站首页）；

对象的组织模式；

常用和最关键的实现算法模型。关键性的实现手段通常包括：

选定基础计算平台，如操作系统、数据库、Web 服务器、中间件平台等；

选定开发工具和开发环境，如计算机语言、构件库、工具软件等。

3.归结目标到最适合的计算体系

通常，提供开发工具和开发环境的组织总是有一些标准的计算体系可以选择（例如，.NET 和 J2EE 等），因此对于大多数系统开发项目来说，比较各种标准计算体系与预期目标之间的匹配程度即可选定计算体系。选择标准的计算体系去实现系统可以忽略大多数基础平台和底层支撑技术的实现问题，从而大大提高系统的质量、降低开发风险和成本。开发人员常根据基础平台的系统实现能力支持，公司或项目组在特定实现平台上的技术积累，甚至技术的“先进性”或流行程度这样的因素去选择系统的实现技术体系。

在另一些情况下，出于各种诸如用户投资力度，与用户现有的 IT 设施保持一致性、兼容性、扩展性及未来维护的能力等因素，系统的基础平台很可能在项目的论证阶段就已经被确定，如操作系统、数据库系统、Web 服务器、开发工具或开发环境等。在这种情况下，系统的实现体系实际上已经确定。

通过同时参考系统概念模型，将前面得到的系统功能清单和系统实现的各种关键要素整理并分类，然后与现有的技术、标准的实现体系进行比较和匹配，就可以将系统概念模型定义的系统目标，进一步映射到真正可计算、可实现的系统架构上。这个过程可以理解为一种不断归结、比较并匹配的过程。

进行匹配的过程常常是一种双向的选择和探究过程，一方面拿出一个系统目标中的功能或实现要素，询问：这部分功能属于表示层、业务逻辑、还是数据服务？另一方面，也研究标准计算体系提供的功能，例如：放在业务逻辑层合适吗？技术人员具有这方面的开发经验积累吗？甚至是标准构件或服务可用吗？

各种标准的计算体系可能很复杂，但通常总是包括一些逻辑上的划分，例如，.NET 体系将应用系统理解为三个层次，表示层、事务逻辑层和数据服务层构成。

（1）表示层：用户的界面部分。例如，单一应用程序的用户界面、C/S 计算模式的客户端、B/S 模式在浏览器中运行的 HTML、DHTML、Scripting、JavaApplet、ActiveX 等。

（2）事务逻辑层：负责处理表示层的应用请求，完成商务逻辑的计算任务，并将处理结果返回给用户。事务逻辑处理层是将原先置于客户端的事务逻辑分离出来，集中置于服务

器部分，为所有用户共享。事务逻辑层是整个应用的核心部分，而组件对象模型 COM 则相当于其心脏。事务逻辑层通过 COM 进行事务处理，并由 IIS（Internet Information Server，Internet 信息服务器）和 MTS（Microsoft Transaction Server，微软事务处理服务器）为各种应用组件提供完善的管理。

（3）数据服务层：为应用提供数据来源。和以往的两层架构不同，数据库不再和每个活动客户保持一个连接，而是若干个客户通过应用逻辑组件共享数据库的连接，从而减少连接次数，提高数据服务器的性能和安全性。

相同的三层计算模式，也会表现为不同的实现方式。例如，表示层可能是单一应用系统的用户界面、C/S 计算的客户端、或 B/S 计算的 Web 页面和元素；事务逻辑层可能是单一应用系统的程序模块、C/S 的服务器端服务、B/S 应用服务器中的业务脚本或业务对象；当利用类似存储过程来实现数据操作逻辑的时候，存储过程也被看作事务逻辑层的一部分，但如果利用 ADO（ActiveX Data Object，ActiveX 数据对象）这样的数据访问组件访问数据时，ADO 和后台的数据库系统及数据库的逻辑则被看作数据服务层的一部分。

在必要的情况下，某个层次还可能进一步细分，例如，使用面向对象设计方法的系统常常会将事务逻辑层划分为基本的计算对象、业务对象及黏合业务对象实现功能的脚本 “胶水”或一些控制类。

不同标准的计算体系的逻辑划分，甚至同一个计算体系的不同版本，通常也不会套用这样的三层分类方式，但却有类似之处。图 7-2 表示了利用 JSP 开发 Web 程序的计算模式。JSP 页面构成了前端的表示层，EJB 构成了业务逻辑层，JDBC（Java DataBase Connectivity，Java 数据库连接）和后台的数据库构成了数据服务层。

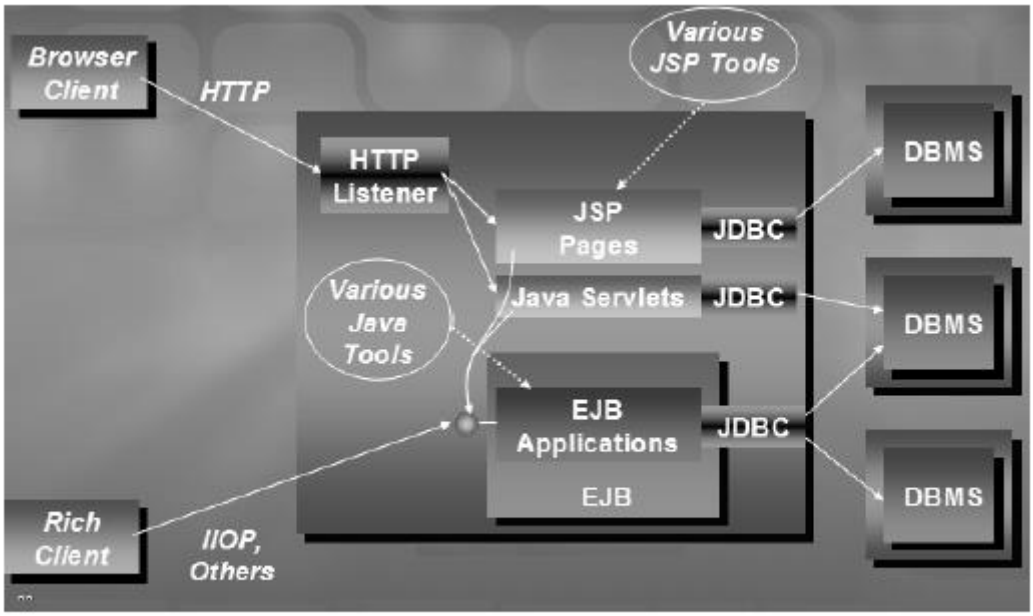


图 7-2　利用 JSP 开发 Web 程序的计算模式

对于小规模的网站系统，开发者可能直接在 JSP 页面中书写所有的应用逻辑脚本，这样业务逻辑层就和表示层合并了，而对于使用 J2EE 体系的开发人员来说，利用 EJB 的容器、对象操作语言等机制直接实现了对象级的接口，开发人员直接在业务逻辑层去构思应用，JDBC 和后台数据库系统的数据服务层被隐含在 J2EE 的平台机制内，在更高的抽象级别上被屏蔽。

因此，归结系统实现要素到计算体系的时候，要点在于理解各种计算体系的大致分层和构成，比较实现要素的目标和实现手段之间的“适合程度”，而不是生搬硬套某种实现机制，或盲目追求某种“流行的”或“先进的”算体系。

系统方案制订后，需要根据有关标准进行评价，找出不符合实际的地方，然后进行改进。

7.4 新旧系统的分析和比较

计算机技术飞速发展，日新月异，许多企业因为业务发展的需要和市场竞争的压力，需要建设新的企业信息系统。在这种升级改造的过程中，怎么处理和利用那些历史遗留下来的老系统，成为影响新系统建设成败和开发效率的关键因素之一。通常称这些老系统为遗留系统。

目前，学术和工业界对遗留系统的定义没有统一的意见。Bennett 在 1995 年对遗留系统做了如下的定义：遗留系统是不知道如何处理但对组织又至关重要的系统。Brodie 和

Stonebraker 对遗留系统的定义如下：遗留系统是指任何基本上不能进行修改和演化以满足新的变化了的业务需求的信息系统。

笔者认为，遗留系统应该具有以下特点：

（1）系统虽然能完成企业中许多重要的业务管理工作，但已经不能完全满足要求。一般实现业务处理电子化及部分企业管理功能，很少涉及经营决策。

（2）系统在性能上已经落后，采用的技术已经过时。如多采用主机/终端形式或小型机系统，软件使用汇编语言或第三代程序设计语言的早期版本开发，使用文件系统而不是数据库。

（3）通常是大型的系统，已经融入企业的业务运行和决策管理机制之中，维护工作十分困难。

（4）系统没有使用现代系统工程方法进行管理和开发，现在基本上已经没有文档，很难理解。

在企业信息系统升级改造过程中，如何处理和利用遗留系统，成为新系统建设的重要组成部分。处理恰当与否，直接关系到新系统的成败和开发效率。遗留系统的演化方式可以有很多种，根据系统的技术条件、商业价值及维护和运行系统的组织特征不同，可以采取继续维护、某种形式的重构或替代策略，或者联合使用几种策略。究竟采用哪些策略来处理遗留系统，需要根据对遗留系统的所有系统特性的评价来确定。

7.4.1 遗留系统的评价方法

对遗留系统评价的目的是为了获得对遗留系统更好的理解，这是遗留系统演化的基础，是任何遗留系统演化项目的起点。本文的评价方法包括度量系统技术水准、商业价值和与之关联的组织特征，其结果作为选择处理策略的基础。

评价方法由一系列活动组成，如图 7-3 所示。

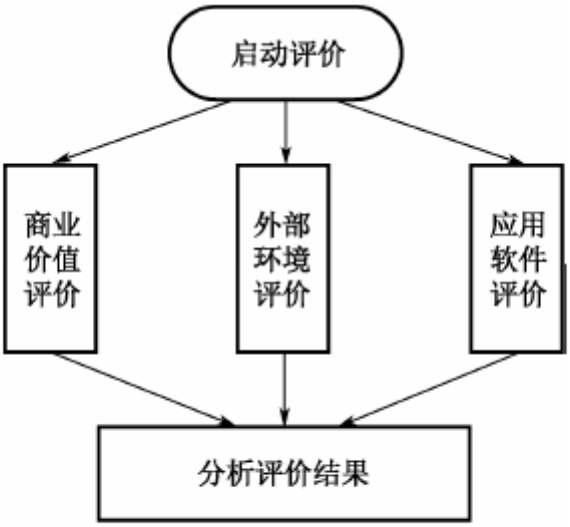


图 7-3　评价活动

1．启动评价

评价是为了获得对遗留系统的足够深度的理解，从技术、商业和企业角度对系统的理解为系统处理策略提供基础，开始评价前，需要了解以下问题。

（1）对企业来说，遗留系统是否是至关重要的。在评价过程中，可能会发现系统对企业的继续运作产生的影响不大。在这种情况下，就没有必要考虑系统的演化问题。

（2）企业的商业目标是什么。从商业观点来看，评估师必须理解企业的商业目标，因为商业目标产生演化需求。

（3）演化需求是什么。演化需求来自企业的商业目标和评价活动。需求必须是可见的，以便决定已存在的系统是否能满足需求。

（4）所期望的系统寿命多长。一个系统的寿命由软件和硬件的服务能力决定，一旦系统硬件或支撑软件过时，系统的有效性就受到限制。

（5）系统使用期限多久。如果系统的使用期限只是短期的，就没有必要花费成本来演化系统。相反，如果系统将在相当长的时期内支持主要业务流程，则必须进行演化。

（6）系统的技术状态如何。例如，如果应用软件的技术状况很差，则很难理解，维护费用会很高。

（7）企业是否愿意改变。企业对改变的态度是遗留系统演化成功的关键因素之一。

（8）企业是否有能力承受演化。企业的技术成熟度，员工的素质，支撑工具的级别等都是影响演化的因素。

2．商业价值评价

商业价值评价的目标是判断遗留系统对企业的重要性。在多数情况下，重要业务过程的改变意味着旧的系统现在仅仅具有外围价值，修改这种系统只需花费少许财力和物力。

在其他情况下，系统的业务价值很大，需要继续维护运行。可以在概要和详细两个级别上进行遗留系统的商业价值评价。

概要级评价将为更加详细的分析提供信息。概要级评价包括：

（1）咨询。向有关专家进行咨询，包括最终用户和负责业务处理的管理人员。

（2）评价问卷。问卷应该标识系统在业务处理过程中的哪些地方使用，本系统与其他系统的关系，如果系统不再运行所需的代价，系统已有的缺点和存在的问题等。问题的准确性依赖于所评价的系统。

（3）进行评价。有了问卷的基础后，必须认真分析系统是如何使用的，这往往会发现系统的价值，而这在问卷中是得不到的。

希赛教育专家提示：详细级评价包括应用系统不符合业务规范的风险分析，这种分析十分费时，最好由业务分析师来完成详细级的评价。

3．外部环境评价系统的外部技术环境是指硬件、支撑软件和企业基础设施的统一体。

（1）硬件。系统硬件包括许多需要进行常规性维护的部件，这些硬件或者在一个站点，或者分布在许多站点并由网络连接。一般来说，遗留系统的硬件包括主机和小型机、磁盘驱动器、磁带、终端、打印机和网络硬件。

与商业价值评价类似，硬件评价也可以分为概要级评价和详细级评价。概要级评价把遗留系统作为一个整体，提供硬件质量估计。详细级评价包括识别系统中的每个部件。在这两种情况下，必须识别一系列特征，用作评价的基础。特征的选择取决于要评价的系统，系统的一些常见特征有供应商、维护费用、失效率、年龄、功能、性能等。

具体评价方法是：每一个部件（或整个系统）在每个特征上分配一个价值分数（取值为1～4），然后把所有分数相加，获得该部件的总分。

（2）支撑软件。系统的支撑软件环境也由许多部分组成，可包括操作系统、数据库、事务处理程序、编译器、网络软件、应用软件等。一般来说，支撑软件是依赖于某个硬件的，应用软件依赖于系统软件。在评价过程中，必须考虑这种依赖性。支撑软件的评价方法类似于硬件评价，在此省略。

（3）企业基础设施。企业基础设施包括开发和维护系统的企业职责和运行该系统的企业职责（两者可能为同一个企业），这些基础设施是很难评价的，但对遗留系统的演化起关键作用。因此必须考虑以下问题。

企业和使用者的类型。企业或者有自己的系统开发队伍，或者所有开发和应用管理都是请其他企业完成。系统用户或许只重复一些记录性工作，或许包括一些更有技术性的工作。

开发组织的技术成熟度。开发组织的技术成熟度包括是否使用了现代系统工程方法，是否遵循了统一的标准，是否进行了过程改进等。

企业的培训过程。如果企业（包括开发方和客户方）的培训做得好，遗留系统的演化可能会更成功。

系统支持人员的技术水平。如果系统支持人员的水平和经验不够，就不要急于对系统做大的改动。

企业是否愿意改变。企业对改变的态度是遗留系统演化成功的关键因素之一。企业基础设施的评价方法类似于硬件评价，在此省略。

4．应用软件评价应用软件评价也有两个级别。

（1）系统级。把整个系统看作是不可分的原子，评价时不考虑系统的任何部分。

（2）部件级。关注系统的每个子系统，考虑每个子系统的特征，包括复杂性、数据、文档、外部依赖性、合法性、维护记录、大小、安全性等。

具体评价方法也与硬件评价类似，在此省略。

5．分析评价结果评价活动将产生硬件、支撑软件、企业基础设施和应用软件的特征值矩阵，这些特征值体现了遗留系统当前的技术因素，其加权平均值代表了系统的技术水平。计算公式如下：

OR=(P1ORH+P2ORS+P3OAF+ P4ORA)/4

其中 ORH 是硬件的评价值，ORS 是支撑软件的评价值，ORF 是企业基础设施的评价值，ORA 是应用软件的评价值，Pi (1 i 4) 分别是它们的权系数，即第 i 个评价值对遗留系统的影响因子。

把对技术水平的全面评价结果与商业评价进行比较，可以为系统演化提供第一手的资料。具体方法是按照商业评价分值和技术水平分值的情况，把评价结果分为四种类型，如图 7-4 所示。

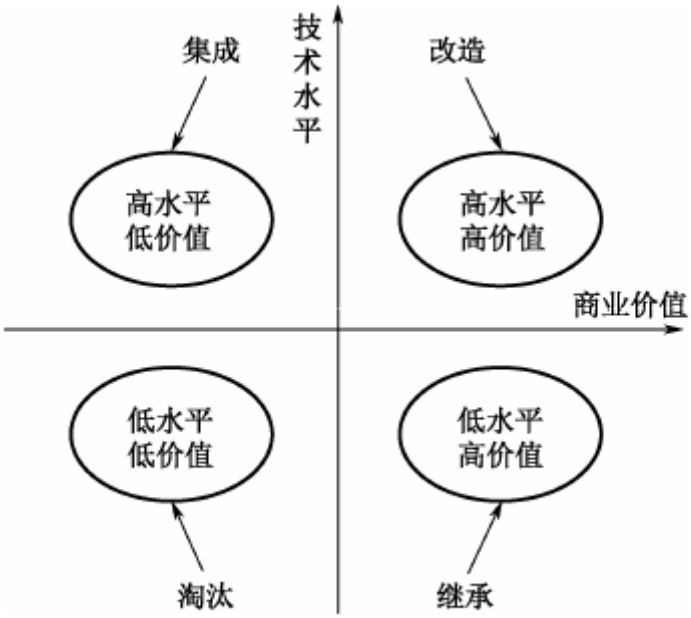


图 7-4　评价结果分析

7.4.2 遗留系统的演化策略

在图 7-4 中，把对遗留系统的评价结果分列在坐标的四个象限内。对处在不同象限的遗留系统采取不同的演化策略。

1．淘汰策略

第 3 象限为低水平、低价值区，即遗留系统的技术含量较低，且具有较低的商业价值。对这种遗留系统的演化策略为淘汰，即全面重新开发新的系统以代替遗留系统。

完全淘汰是一种极端性策略，一般是企业的业务产生了根本的变化，遗留系统基本上不再适应企业运作的需要；或者是遗留系统的维护人员、维护文档资料都丢失了。经过评价，发现将遗留系统完全淘汰，开发全新的系统比改造旧系统从成本上更合算。

对遗留系统的完全淘汰是企业资源的根本浪费，应该善于“变废为宝”，通过对遗留系统功能的理解和借鉴，可以帮助新系统的设计，降低新系统开发的风险。

2．继承策略

第 4 象限为低水平、高价值区，即遗留系统的技术含量较低，可满足企业运作的功能或性能要求，但具有较高的商业价值，目前企业业务对该系统仍有很大的依赖性。对这种遗留系统的演化策略为继承。在开发新系统时，需要完全兼容遗留系统的功能模型和数据模型。为了保证业务的连续性，新老系统必须并行运行一段时间，再逐渐切换到新系统上运行。

要做到对遗留系统的继承，必须对系统进行分析，得到旧系统的功能模型和数据模型，这种分析可以部分代替或验证系统的需求分析。

如果遗留系统的维护文档不完整，而又必须解析系统的功能模型和数据模型，那将是一项十分艰巨的任务。这时可使用有关系统重构的 CASE 工具，通过分析系统的代码生成系统结构图或其他报告。

3．改造策略

第 1 象限为高水平、高价值区，即遗留系统的技术含量较高，本身还有较大的生命力，且具有较高的商业价值，基本上能够满足企业业务运作和决策支持的要求。这种系统可能建成的时间还很短，对这种遗留系统的演化策略为改造。

这些改造包括系统功能的增强和数据模型的改造两个方面。系统功能的增强是指在原有系统的基础上增加新的应用要求，对遗留系统本身不做改变。数据模型的改造是指将遗留系统的旧的数据模型向新的数据模型转化的过程。

4．集成策略

第 2 象限为高水平、低价值区，即遗留系统的技术含量较高，但其商业价值较低，可能只完成某个部门（或子公司）的业务管理。这种系统在各自的局部领域里工作良好，但从企业全局来看，多个这样的系统，他们各自基于不同的平台，不同的数据模型，无法互联互通，数据还不一致，这就是很严重的问题了。

对这种遗留系统的演化策略为集成。

在集成过程中，可采用由互连系统构成的系统的架构，遗留系统可作为从属系统来描述。

在企业信息系统建设过程中，如何处理那些遗留系统，将会是越来越突出的问题，因为即使是今天看来很先进的系统在明天也会成为遗留系统。对遗留系统的处理恰当与否，直接关系到新系统的成败和开发效率。如何建立一套系统的、行之有效的方法，以期望对实际工作有所指导，已成为一个迫切的问题。在实际工程项目中，遇到处理遗留系统的问题时，要具体情况具体分析，选择最佳的演化策略。

第 8 章：系统分析与设计方法

对于架构设计师而言，如何进行系统设计是其“看家本领”，而设计是在对系统进行分析的基础上进行的，否则，设计就是“无米之炊”。从软件开发项目中的角色分配来看，系统架

构设计师应该在信息系统项目管理师的协调下，与系统分析师协同工作。

8.1 定义问题与归结模型

软件系统的目的是为了解决问题，因此在建模之初最重要的步骤是对问题的分析与定义，并在此基础上归结模型，这样才能够获得切实有效的模型。定义问题的过程包括：理解真实世界中的问题和用户的需要，并提出满足这些需要的解决方案的过程。

8.1.1 问题分析

问题分析的目标就是在开发之前对要解决的问题有一个更透彻的理解。为了达到这一目标，通常需要经过在问题定义上达成共识，理解问题的本质，确定项目干系人和用户，定义系统的边界和确定系统实现的约束这五个步骤。

1．在问题定义上达成共识

要检验大家是否在问题的定义上达成了共识，最简单的方法就是把问题写出来，看看是否能够获得大家的认可。而要使得这个过程更加有效，应该将问题用标准化的格式写出来，根据 UP 的建议，应该包括以下几个方面的要素。

问题概述：用简短的几句话，将所理解的问题本质描述出来；

影响：说明该问题将会对哪些项目干系人（Stakeholder，风险承担者）产生影响；

结果：确定问题对项目干系人和商业活动会产生什么样的影响；

优点：概要性地提出解决方案，并列举出该解决方案的主要优点。

在问题定义上达成共识，就能够有效地将开发团队的理解与用户的需求达成一致，这样就能够使得整个系统的开发沿着合理的方向发展。

2．理解问题的本质

每一句描述都会夹杂着叙述者的个人理解和判断，因此透过表面深入本质，理解问题背后的问题，是问题分析阶段一个十分关键的任务。其中一种技术是“根本原因”分析，这是一种提示问题或其表象的根本原因的系统化方法。在实际的应用中，常使用因果鱼骨图和帕累托图两种方法。

（1）因果鱼骨图。因果鱼骨图是一种有效的探寻问题根源的技术，它通过直观的图形找出问题或现象的所有潜在原因，从而追踪出问题的根源。它能够帮助人们将问题的原因而放在首位，提供了一种运用集体智慧解决问题的方法。在使用时，通常按照以下步骤进行。

将问题简明扼要地写在右边的方框里；

确定问题潜在原因的主要类别，将它们连到鱼的脊骨上；

用头脑风暴法寻找原因并归类。图 8-1 是鱼骨图的一个示例。

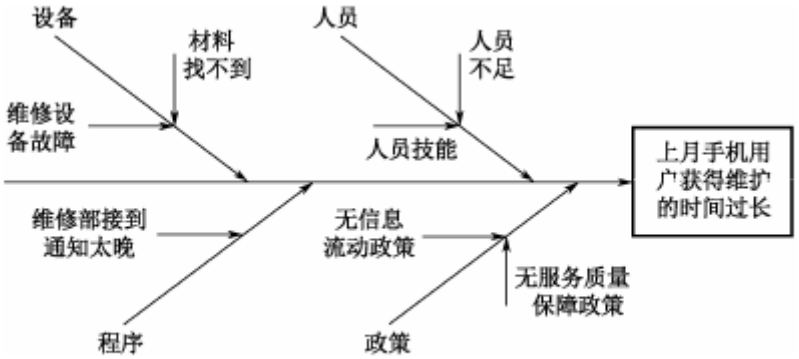


图 8-1　鱼骨图示例

（2）帕累托图。

帕累托图是采用直方图的形式，根据问题的相对频率或大小从高往低降序排列，帮助设计师将精力集中在重要的问题上。它为 80%的问题找到关键的 20%的原因，它可以一目了然地显示出各个问题的相对重要程度，有助于预防在解决了一些问题后，却使另外一些问题变得更糟的现象发生。在使用时，通常按照以下步骤进行。

明确问题：也就是前面达成共识的问题定义；

找出问题的各种可能原因：通常可以利用头脑风暴来收集意见，并通过参考以往积累的资料和运营的数据来综合分析；

选择评价标准和考察期限：最常用的评价标准包括频率（占总原因的百分比）和费用（产生的影响），而考察的期限则应具有相应问题的代表性，并不是越长越好；

收集各种原因发生的频率及费用数据；

将原因按照发生的频率或费用从大到小排列起来；

将原因排在横轴上，频率或费用排列在纵轴上，形成如图 8-2 所示的结果。

这样就能够将造成问题的关键原因捕获出来，以便指导设计出更符合需要、更能够解决问题的解决方案。

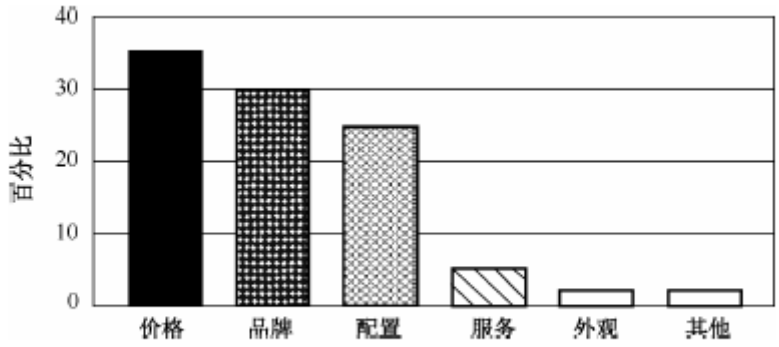


图 8-2　帕累托图示例

3．确定项目干系人和用户

要想有效地解决问题，必须了解用户和其他相关的项目干系人（任何将从新系统或应用的实现中受到实质性影响的人）的需要。不同的项目干系人通常对问题有不同的看法和不同的需要，这些在解决问题时必须加以考虑。事实上，许多项目干系人就是系统的用户，这一部分通常是易于识别的；但还有一部分项目干系人是系统的间接用户，甚至只是受系统影响的商业结果，这一部分不易识别，但十分重要。

在寻找项目干系人时，可以思考：系统的用户是谁？系统的客户（购买者）是谁？还有哪些人会受到系统输出的影响？系统完成并投入使用后，有谁会对它进行评估？还有没有其他系统内部或外部的客户，他们的需要有没有必要去考虑？系统将来由谁来维护？

4．定义系统的边界

系统的边界是指解决方案系统和现实世界之间的边界。在系统边界中，信息以输入和输出的形式流入系统并由系统流向系统外的用户，所有和系统的交互都是通过系统和外界的接口进行的。在定义系统的边界时，将世界分为两个部分：系统及与系统进行交互的事物。要描述系统的边界有两种方法：一种是结构化分析中的“上下文范围图”，另一种则是面向对象分析中的“用例模型”。

（1）上下文范围图。也就是数据流图中的顶层图，它是一个反映领域信息的模型，能够清晰地显示出系统的工作职责和相邻系统的职责起止之处，从而让读者能够从宏观的层面去了解系统。图 8-3 就是一个描述“证券经纪人系统”的上下文范围图。

（2）用例模型。用例模型则通过引入参与者来描述“和系统进行交互的事物”，只要识别了参与者，自然而然系统的界限就确定下来了。在寻找参与者时，可以思考以下问题：谁会对系统提供信息？谁会在系统中使用信息？谁会从系统中删除信息？谁将操作系统？系统将会在哪里被使用？系统从哪里得到信息？哪些外部系统要和系统进行交互？

然后，再根据每个参与者的功能需求，识别出代表系统功能的用例，从而界定系统的边

界。而关于用例模型的更多细节，请参考 8.4.3 节。

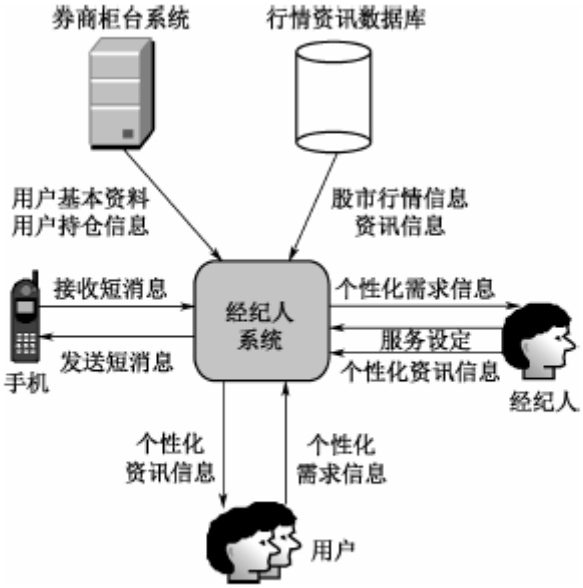


图 8-3　上下文范围图示例

5．确定系统实现的约束

由于各种因素的存在，会对解决方案的选择造成一定的限制，称这种限制为约束。每条约束都将影响到最后的解决方案的形成，甚至会影响是否能够提出解决方案。

在考虑约束时，首先应该考察到不同的约束源，其中包括进度、投资收益、人员、设备预算、环境、操作系统、数据库、主机和客户机系统、技术问题、行政问题、已有软件、公司总体战略和程序、工具和语言的选择、人员及其他资源限制等。

8.1.2 问题定义

通过对问题进行细致周密的分析，就可以对其进行综合的定义。对于一个问题的完整定义，通常应包括目标、功能需求和非功能需求三个方面。

1．目标

目标是指构建系统的原因，它是最高层次的用户需求，是业务上的需要，而功能、性能需求则必须是以某种形式对该目标做出贡献。在描述目标时，应该注意以下几个方面。

优势：目标应该不仅仅是解决问题，还要提供业务上的优势；

度量：不仅要说明业务的优势，而且还必须提供度量这种优势的标准；

合理性：要确保完成解决方案所需的工作量少于所获得的业务优势，这才是合理的解决方案；

可行性：要探寻能够满足度量标准的解决方案；

可达成性：对于组织而言，是否具备获取该系统的技能，构建完成后是否能够操作它。

例如，表 8-1 就是一个很好的目标描述的例子。

表 8-1　目标描述的例子

目标：在冬季道路养护支出上节省费用
优势：减少除冰和道路养护的费用
度量标准：除冰费用将在目前道路处理费用的基础上降低 25%，冰对道路造成的损伤将降低 50%

2．功能需求

功能需求是用来指明系统必须做的事情，只有这些行为的存在，才有系统存在的价值。功能需求应该源于业务需求，它只与问题域相关，与解决方案域无关。也就是说，功能需求是在与用户或某个业务人员交谈时，他们所描述的内容是为了完成他们某部分的工作而必须做的事情。而在设计解决方案时，会遇到一些限制条件，这些东西也是“系统需求” 的一部分，不过应该是设计约束或非功能需求形式指定。

在规定功能需求时要注意其详细程度。由于这些需求是业务需求，因此应该由业务人员来验证。也就是说，用户应该能够指明系统要达到有用的程度，功能是否已经足够；考虑到工作的成果，它的功能是否正确。另外，在描述功能需求时，应该注意需求的二义性。而二义性主要体现在以下几个方面。

（1）同名异义的词：在自然语言中存在许多同名但异义的词语，应该谨慎地排除它们带来的影响。

（2）代词：在需求描述中，代词经常会产生指代不明的现象，应该尽量避免使用，而是换成主语及宾语。

希赛教育专家提示：在检查功能需求的二义性时，一种有效的方法是大声地朗读出来，大家一起边听边进行讨论，这样可以不断地优化。

3．非功能需求

非功能需求是系统必须具备的属性，这些属性可以看作是一些使产品具有吸引力、易用、快速或可靠的特征或属性。非功能需求并不改变产品的功能，它是为工作赋予特征的。在识别功能需求和非功能需求时，有一种十分有用的思维模式：功能需求是以动词为特征的，而非功能性需求则是以副词为特征的。非功能需求主要包括以下几种。

（1）观感需求：即产品外观的精神实质，也就是与用户界面的观感相关的一组属性；

（2）易用性需求：也就是产品的易用性程度，以及特殊的可用性考虑，通常包括用户的接受率、因为引入该产品而提高的生产效率、错误率、特殊人群的可用性等指标；

（3）性能需求：也就是关于功能实现要求有多快、多可靠、多少处理量及多精确的约束。例如：速度、精度、安全性、容量、值范围、吞吐量、资源使用效率、可靠性（平均无故障时间）、可用性（不停机时间）、可扩展性等；

（4）可操作性需求：衡量产品的操作环境，以及对该操作环境必须考虑的问题；

（5）可维护性和可移植性需求：期望的改变，以及完成改变允许的时间；

（6）安全性需求：产品的安全保密性，通常体现为保密性、完整性和可获得性；

（7）文化和政策需求：由产品的开发者和使用者所带来的特别需求；

（8）法律需求：哪些法律和标准适用于本产品。

8.2 需求分析与软件设计

需求分析是软件生命周期中相当重要的一个阶段。根据 Standish Group 对 23000 个项目进行的研究结果表明，28%的项目彻底失败，46%的项目超出经费预算或者超出工期，只有约 26%的项目获得成功。需求分析工作在整个软件开发生命周期中有着十分重要的意义。而在这些高达 74%的不成功项目中，有约 60%的失败是源于需求问题，也就是差不多有一半的项目都遇到了这个问题，这一可怕的现象引起人们对需求分析的高度重视。需求分析阶段的主要任务是通过开发人员与用户之间的广泛交流，不断澄清一些模糊的概念，最终形成一个完整的、清晰的、一致的需求说明。

而当明确了用户的需求之后，下一步的任务就是对未来的软件系统进行设计，它是确定系统实现的关键工作。需求分析和设计的方法对软件开发过程而言是十分重要的，因此必须扎实地掌握它。

需求分析与软件设计是软件生存期中最重要的两个步骤，需求分析解决的是“做什么”的问题，系统设计则是解决“怎么做”的问题。

8.2.1 需求分析的任务与过程

需求分析所要做的工作是深入描述软件的功能和性能，确定软件设计的限制和软件同其他系统元素的接口细节，定义软件的其他有效性需求，细化软件要处理的数据域。用一句话概括就是：需求分析主要是确定待开发软件的功能、性能、数据、界面等要求。需求分析的实现步骤通常包括：获取当前系统的物理模型，抽象出当前系统的逻辑模型，建立目标系统的逻辑模型三个部分。具体来说，需求分析阶段的工作可以分成 4 个方面：

（1）问题识别：用于发现需求、描述需求，主要包括功能需求、性能需求、环境需求、可靠性需求、安全保密需求、用户界面需求、资源使用需求、软件成本消耗与开发进度需求，以此来预先估计以后系统可能达到的目标。

（2）分析与综合：也就是对问题进行分析，然后在此基础上整合出解决方案。这个步骤经常是反复进行的，常用的方法有面向数据流的结构化分析方法（Structured Analysis，SA），面向数据结构的 Jackson 方法，面向对象的分析方法（Object Oriented Analysis，OOA），以及用于建立动态模型的状态迁移图和 Petri 网。

（3）编制需求分析的文档：也就是对已经确定的需求进行文档化描述，该文档通常称为“需求规格说明书”。

（4）需求分析与评审：它是需求分析工作的最后一步，主要是对功能的正确性、完整性和清晰性，以及其他需求给予评价。

1．需求的分类

什么是软件的需求呢？软件需求就是系统必须完成的事及必须具备的品质。具体来说，软件需求包括功能需求、非功能需求和设计约束三方面内容。各种需求的概念示意图如图 8-4 所示。

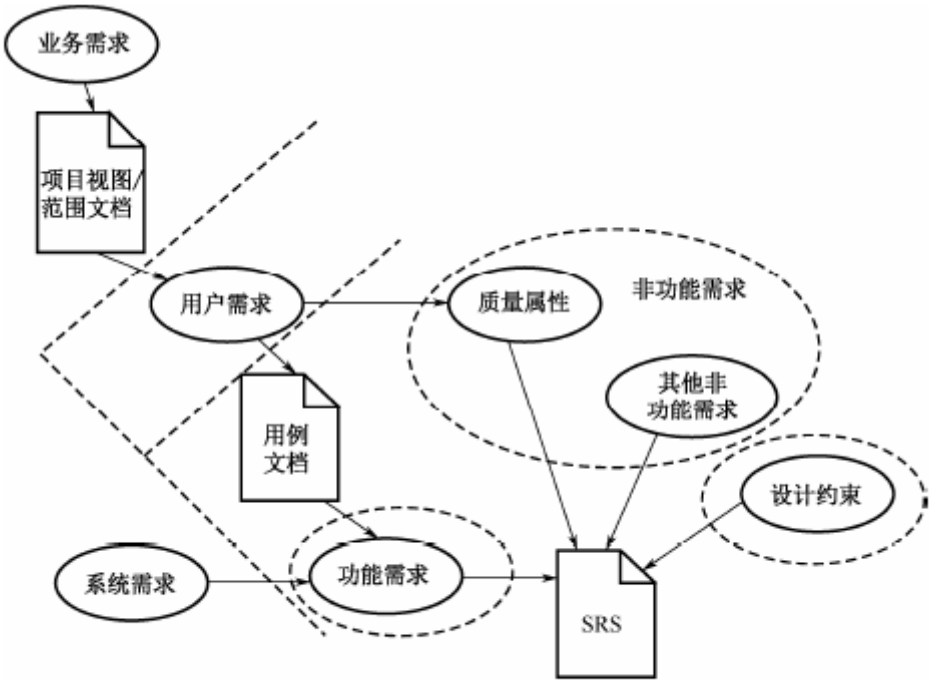


图 8-4　需求概念示意图

功能需求：是指系统必须完成的那些事，即为了向它的用户提供有用的功能，产品必须执行的动作。

非功能需求：是指产品必须具备的属性或品质，如性能、响应时间、可靠性、容错性、扩展

性等。

设计约束：也称为限制条件、补充规约，这通常是对解决方案的一些约束说明，例如必须采用国有自主知识版权的数据库系统，必须在 UNIX 操作系统之下运行等。

除了这三种需求之外，还有业务需求、用户需求、系统需求这三个处于不同层面的概念，充分地理解这样的模型才能够更加清晰地理清需求的脉络。

业务需求（Business Requirement）：是指反映组织机构或客户对系统、产品高层次的目标要求，通常问题定义本身就是业务需求。

用户需求（User Requirement）：是指描述用户使用产品必须要完成什么任务，怎么完成的需求，通常是在问题定义的基础上进行用户访谈、调查，对用户使用的场景进行整理，从而建立从用户角度出发的需求。

系统需求（System Requirement）：是从系统的角度来说明软件的需求，它包括用特性说明的功能需求、质量属性、非功能需求及设计约束。

分析师经常围绕着“功能需求”来展开工作，而功能需求大部分都是从“系统需求” 的角度来分析与理解的，也就是用“开发人员”的视角来理解需求。但要想真正地得到完整的需求，仅戴上“开发人员”的眼镜是不够的，还需要“领域专家”的眼镜，要从更高的角度来理解需求，这就是“业务需求”；同时还应该更好地深入用户，了解他们的使用场景，了解他们的想法，这就是“用户需求”。这是一个理解层次的问题，并不仅仅是简单的概念。

2．需求工程

需求工程就是包括创建和维护系统需求文档所必需的一切活动的过程，主要包括需求开发和需求管理两大工作。

（1）需求开发：包括需求捕获、需求分析、编写规格说明书和需求验证 4 个阶段。在这个阶段需要完成确定产品所期望的用户类型、获取每种用户类型的需求、了解实际用户任务和目标及这些任务所支持的业务需求、分析源于用户的信息、对需求进行优先级分类、将所收集的需求编写成为软件规格说明书和需求分析模型、对需求进行评审等工作。

（2）需求管理：通常包括定义需求基线、处理需求变更、需求跟踪等方面的工作。

这两个方面是相辅相成的，需求开发是主线，是目标；需求管理是支持，是保障。换句话说，需求开发是努力更清晰、更明确地掌握客户对系统的需求；而需求管理则是对需求的变化进行管理的过程。

3．需求分析方法

需求分析的方法可谓种类繁多，不过如果按照分解方式的不同，可以很容易地划分出几

种类型。本节先从分析方法发展的历史，对其建立一个概要性的认识，在本章的后面几节中将详细说明最具有代表性的结构化分析与设计、面向对象分析与设计两种方法。

（1）结构化分析方法：最初的分析方法都不成体系，而且通常都只包括一些笼统的告诫，在 20 世纪 70 年代分析技术发展的分水岭终于出现了。这时人们开始尝试使用标准化的方法，开发和推出各种名为“结构化分析”的方法论，而 Tom DeMacro 则是这个领域最有代表性和权威性的专家。

（2）软系统方法：这是一个过渡性的方法论，并未真正流行过，它的出现只是证明了结构化分析方法的一些不足。因为结构化分析方法采用的相对形式化的模型不仅与社会观格格不入，而且在解决“不确定性”时显得十分无力。最有代表性的软系统方法是 Checkland 方法。

（3）面向对象分析方法：在 20 世纪 90 年代，结构化方法的不足在面对多变的商业世界时，显得更加苍白无力，这就催使了 OOA 的迅速发展。

（4）面向问题域的分析（Problem Domain Oriented Analysis，PDOA）：现在又发现面向对象分析方法也存在着很多的不足，应运而生了一些新的方法论，PDOA 就是其中一种。不过现在还在研究阶段，并未广泛应用。

8.2.2 如何进行系统设计

当设计者拿到一个需求，他如何开展系统设计呢？许多想进入系统分析与设计的年轻人以为自己知道了面向对象、统一建模语言、设计模式等新鲜深奥的名词就可以进行设计了，可是掌握工具和技能绝不是成为优秀设计者的充分条件，甚至也不是必要条件。遗憾的是这里没有捷径，只有设计者在实践中不断学习和总结。而在实践中，系统设计与其说是在设计，不如说是在选择和妥协。

妥协，就是在各个系统目标之间找到一个平衡点。系统目标包括但不限制于功能、性能、健壮性、开发周期、交付日期等。不幸的是，这些目标往往是矛盾的，提高软件性能直接意味着开发周期的增加、交付日期的推迟，盲目地增加功能可能导致性能降低，维护成本提高。软件设计者的难题在于在如此众多的目标之间找到这个平衡点，并且明确知道如何设计能实现这个平衡，既可以让投资者觉得在预算之内，又能让用户相对满意。可行性分析阶段应该已经论述了这样一个平衡点，可是如果设计者发现没有这样一个平衡点，如同没有一个设计者能让人骑着自行车到月球上去，那么设计者只能提出放弃某个方面的过度要求，否

则系统要遭受必然失败的命运。更多的情况是没有经验的设计者不知道是否存在这些平衡点，更不知道如何利用合理的设计及有效的工具来达到平衡。因此设计者需要了解可以解决问题的各种方案，并清楚知道各个方案的效果、成本、缺点，以及这些方案的区别，并在各种方案中进行选择。而这些，不是一个人能在一两天了解的。

没有一个设计者会完全重新开始设计一个系统，他们总参考多个与目标系统相类似的系统，再从中进行甄别、取舍和补充来作为新系统的设计。人们发现一个优秀的设计者似乎能在听完需求后就立即构想出目标系统的框架，这并不是因为他聪明或者比不知所措的设计者新手多一个脑袋，而是因为他在平时已经了解大量的系统，对各种设计的优缺点、局限性也了然于胸，能够把以前的设计根据需要再次使用。所以要成为优秀的设计者，了解、掌握、消化、总结前人和自己以前的设计成果是最好的方法，这似乎也是唯一的方法。

设计者的苦恼有时候和编程人员一样。计算机系统的发展如此迅速，解决方案也越来越多，如同编程语言的发展，同时，随着人类社会的进步，投资者和客户也提出了越来越高的要求，这又需要设计者不断学习、创造新的方案。

但系统设计也并非没有规律可以遵循，如同幸福的家庭都很相似，不幸的家庭各有各的不幸，人们在实践中发现优秀的系统设计一般在以下几个方面都很出色。

（1）组件的独立性。审视自己设计的系统，是否做到了高内聚、低耦合？

（2）例外的识别和处理。谁能保证系统使用者都精确按照使用说明书使用？

（3）防错和容错。当网络中断、数据库崩溃这样的灾难性事件发生时，系统也跟着崩溃吗？

而且，更幸运的是，也有一些技术能够改进系统设计，这些方法包括：降低复杂性、通过合约进行设计、原型化设计、错误树分析等。

8.2.3 软件设计的任务与活动

软件设计是一个把软件需求变换成软件表示的过程。最初这种表示只是描绘出软件的总体框架，然后再进一步细化，并在此框架中填入细节。

1.软件设计的两个阶段从工程管理角度，软件设计可以分为两个步骤：

（1）概要设计：也称为高层设计，将软件需求转化为数据结构和软件的系统结构。例如，如果采用结构化设计，则将从宏观的角度将软件划分成各个组成模块，并确定模块的功能及模块之间的调用关系。

（2）详细设计：也称为低层设计，将对结构表示进行细化，得到详细的数据结构与算法。同样的，如果采用结构化设计，则详细设计的任务就是为每个模块进行设计。

2．主要的设计方法比较

在结构化设计风行的时代，主流的设计方法还包括 Jackson 方法和 Parnas 方法。结构化方法侧重于“模块相对独立且功能单一，使模块间联系弱、模块内联系强”；而 Jackson 方法则是从数据结构导出模块结构；Parnas 方法的主要思想则是将可能引起变化的因素隐藏在有关模块内部，使这些因素变化时的影响范围受到限制，它只提供了重要的设计准则，但没有规定出具体的工作步骤。

而近年来，对象技术凭借其对数据的高效封装及良好的消息机制，实现了高内聚、低耦合的系统设计，成了现代软件设计的主流方法学。

8.3 结构化分析与设计

结构化分析与设计方法是一种面向数据流的需求分析和设计方法，它适用于分析和设计大型数据处理系统，是一种简单、实用的方法，曾获得广泛的应用。

8.3.1 结构化分析

结构化分析方法的基本思想是自顶向下逐层分解。分解和抽象是人们控制问题复杂性的两种基本手段。对于一个复杂的问题，人们很难一下子考虑问题的所有方面和全部细节，通常可以把一个大问题分解成若干个小问题，每个小问题再分解成若干个更小的问题，经过多次逐层分解，每个最底层的问题都是足够简单、容易解决的，于是复杂的问题也就迎刃而解了。这个过程就是分解过程。

结构化分析与面向对象分析方法之间的最大差别是：结构化分析方法把系统看作一个过程的集合体，包括人完成的和电脑完成的；而面向对象方法则把系统看成一个相互影响的对象集。结构化分析方法的特点是利用数据流图来帮助人们理解问题，对问题进行分析。

结构化分析一般包括以下工具：数据流图（Data Flow Diagram，DFD）、数据字典（Data Dictionary，DD）、结构化语言、判定表、判定树。在接下来的部分将对它们一一做简单介绍。

结构化系统分析方法从总体上来看是一种强烈依赖数据流图的自顶向下的建模方法。它不仅是需求分析技术，也是完成需求规格化的有效技术手段。

1．结构化分析的工作步骤

在介绍具体的结构化分析方法之前，先对如何进行结构化分析做一个总结性描述，以帮助大家更好地应用该方法。

（1）研究“物质环境”。首先，应画出当前系统（可能是非计算机系统，或是半计算机系统）的数据流图，说明系统的输入、输出数据流，说明系统的数据流情况，以及经历了哪些处理过程。在这个数据流图中，可以包括一些非计算机系统中数据流及处理的命名，例如部门名、岗位名、报表名等。这个过程可以帮助分析员有效地理解业务环境，在与用户的充分沟通与交流中完成。

（2）建立系统逻辑模型。当物理模型建立完成之后，接下来的工作就是画出相对于真实系统的等价逻辑数据流图。在前一步骤建立的数据流图的基础上，将所有自然数据流都转成等价的逻辑流，例如，将现实世界的报表存储在计算机系统中的文件里；又如将现实世界中“送往总经理办公室”改为“报送报表”。

（3）划清人机界限。最后，确定在系统逻辑模型中，哪些将采用自动化完成，哪些仍然保留手工操作。这样，就可以清晰地划清系统的范围。

2．数据流图

DFD 是一种图形化的系统模型，它在一张图中展示信息系统的主要需求，即输入、输出、处理（过程）、数据存储。由于从 DFD 中可以很容易地看出系统紧密结合的各个部分，而且整个图形模式只有 5 个符号需要记忆，所以深受分析人员的喜爱，因而广为流行。

如图 8-5 所示，DFD 中包括以下几个基本元素。

过程：也称为加工，一步步地执行指令，完成输入到输出的转换。

外部实体：也称为源/宿，系统之外的数据源或目的。

数据存储：也称为文件，存放数据的地方，一般是以文件、数据库等形式出现。

数据流：从一处到另一处的数据流向，如从输入或输出到一个过程的数据流。

实时连接：当过程执行时，外部实体与过程之间的来回通信。

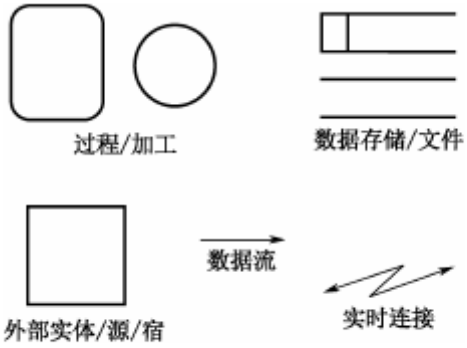


图 8-5　数据流图的符号集

希赛教育专家提示：不同的参考书籍中关于 DFD 的符号可能有些不一样，其中加工、外部实体、数据存储和数据流是公共的部分。

（1）数据流图的层次。正如前面提到的，结构化分析的思路是依赖于数据流图进行自顶而下的分析。这也是因为系统通常比较复杂，很难在一张图上将所有的数据流和加工描述清楚。因此，数据流图提供一种表现系统高层和低层概念的机制。也就是先绘制一张较高层次的数据流图，然后在此基础上，对其中的过程（处理）进行分解，分解成为若干独立的、低层次的、详细的数据流图，而且可以这样逐一地分解下去，直至系统被清晰地描述出来。数据流图的层次如图 8-6 所示。

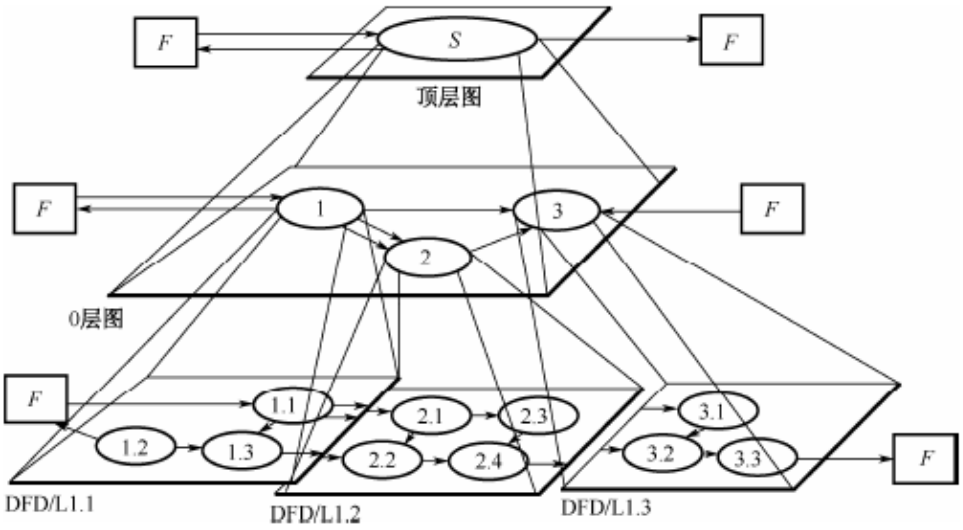


图 8-6　数据流图的层次

（2）Context 图。Context 图，也就是系统上下文范围关系图。这是描述系统最高层结构的 DFD 图。它的特点是，将整个待开发的系统表示为一个过程，将所有的外部实体和进出系统的数据流都画在一张图中。图 8-7 就是一个 Context 图的例子。

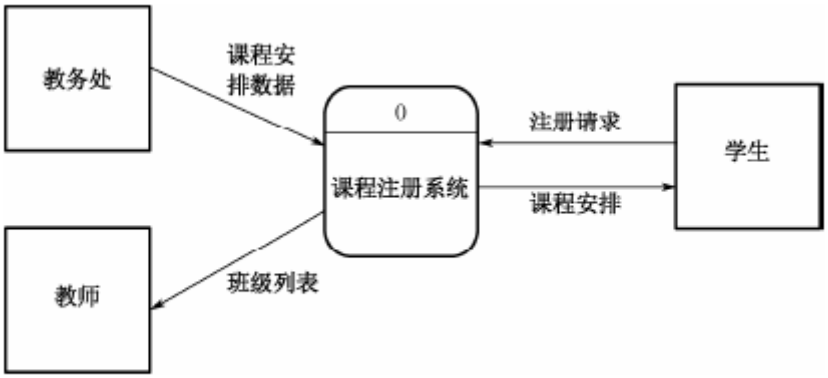


图 8-7　Context 图实例

Context 图用来描述系统有什么输入、输出数据流，与哪些外部实体直接相关，可以把整个系统的范围勾画出来。

（3）逐级分解。当完成了 Context 图的建模之后，就可以在此基础上进行进一步的分解。以图 8-7 为例，进行再分解，在对原有流程了解的基础上，可以得到如图 8-8 所示的结果。

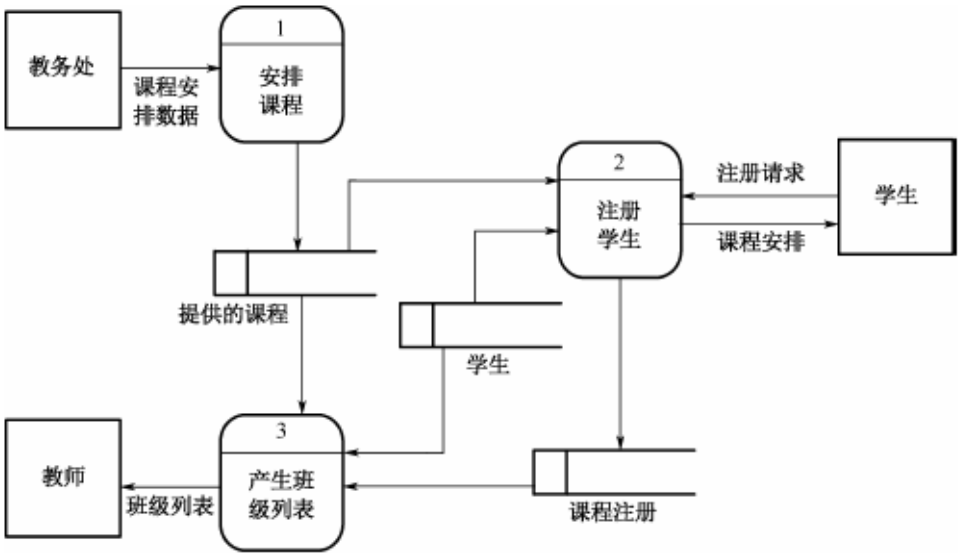


图 8-8　DFD 0 层图实例

图 8-8 是在 Context 图的基础上做的第一次分解，而在 Context 图中只有一个过程，那就是系统，将其编号为 0。而接下来对 Context 图进行的分解，其实就是对这个编号为 0 的过程进行更细化的描述，在这里引入了新的过程、数据存储，为了能够区分其位置的级别，在这层次上的过程将以 1、2、3 为序列进行编号。

由于这是对过程 0 的分解，因此也称之为 DFD 0 层图。而可以根据需要对 DFD 0 层图上的过程（编号为 1、2、3）进行类似的分解，那么就称之为 DFD 1 层图，在 DFD 1 层

图中引入的新过程，其编号规则就是 1.1，1.2…，以及 2.1，2.2…，以此类推，直到完成分析工作。

另外，这里存在一个很关键的要点，那就是 DFD 0 层图是 Context 图的细化，因此所有的输入和输出应该与 Context 图完全一致，否则就说明存在着错误。

（4）如何画 DFD。DFD 的绘制是一个自顶向下、由外到里的过程，通常按照以下几个步骤进行。

画系统的输入和输出：就是在图的边缘标出系统的输入、输出数据流。这一步其实是决定研究的内容和系统的范围。在画的时候，可以先将尽可能多的输入、输出画出来，然后再删除多余的，增加遗漏的。

画数据流图的内部：将系统的输入、输出用一系列的处理连接起来，可以从输入数据流画向输出数据流，也可以从中间画出去。

为每一个数据流命名：命名的好坏与数据流图的可理解性密切相关，应避免使用空洞的名字。

为加工命名：注意应该使用动宾短语。

不考虑初始化和终点，暂不考虑出错路径等细节，不画控制流和控制信息。

3.细化记录 DFD 部件

为了更好地描述 DFD 的部件，结构化分析方法还引入了数据字典、结构化语言及决 策树、决策表等方法。通过使用这些工具，能对数据流图中描述不够清晰的地方进行有效的补充。其其中数据字典应用最为广泛，下面将详细说明数据字典的相关使用方法。

数据字典技术是一种很实用、有效的表达数据格式的手段。它是对所有与系统相关的 数据元素的一个有组织的列表和精确严格的定义，使得用户和系统分析员对于输入、输出、 存储成分和中间计算机有共同的理解。通常数据字典的每一个条目中包括以下信息。

① 名称：数据或控制项、数据存储或外部实体的主要名称，如果有别名的还应该将别名列出来。

② 何处使用/如何使用：使用数据或控制项的加工列表，以及如何使用。

③ 内容描述：说明该条目的内容组成，通常采用以下符号进行说明。

=：由…构成。

+：和，代表顺序连接的关系。

[|]：或，代表从中选择一个。

{}*：n 次重复。

（）：代表可选的数据项。

…：表示特定限制的注释。

④ 补充信息：关于数据类型、默认值、限制等信息。

下面就是一个数据字典的实例：

客户基本信息=客户编号+客户名称+身份证号码+手机+家庭电话

客户编号 = {0…9}8

客户名称 = {字}4

身份证号码 = [{0…9}15|{0…9}18]

手机 = [{0…9}11|{0…9}12]

家庭电话 =（区号）+本地号区号 = {0…9}4

本地号 = [{0…9}7|{0…9}8]

8.3.2 结构化设计

结构化设计包括架构设计、接口设计、数据设计和过程设计等任务。它是一种面向数据流的设计方法，是以结构化分析阶段所产生的成果为基础，进一步自顶而下、逐步求精和模块化的过程。

1．概要设计与详细设计的主要任务

概要设计阶段的主要任务是设计软件的结构、确定系统是由哪些模块组成，以及每个模块之间的关系。它采用结构图（包括模块、调用、数据）来描述程序的结构，此外还可以使用层次图和 HIPO（层次图加输入/处理/输出图）。

整个过程主要包括：复查基本系统模型、复查并精化数据流图、确定数据流图的信息流类型（包括交换流和事务流）、根据流类型分别实施变换分析或事务分析、根据软件设计原则对得到的软件结构图进一步优化。

而详细设计阶段的主要任务则是确定应该如何具体地实现所要求的系统，得出对目标系统的精确描述。它采用自顶向下、逐步求精的设计方式和单入口单出口的控制结构。常使用的工具包括程序流程图、盒图、PAD（Problem Analysis Diagram，问题分析图）、PDL（Program Design Language，程序设计语言）。

2．结构图

如图 8-9 所示，结构图的基本成分包括模块、调用（模块之间的调用关系）和数据（模块间传递及处理数据信息）。

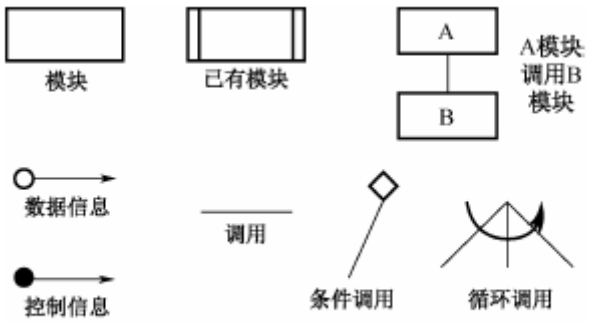


图 8-9　结构图的基本成分

结构图是在需求分析阶段产生的数据流图的基础上进行进一步的设计。它将 DFD 图中的信息流分为两种类型。

变换流：信息首先沿着输入通路进入系统，并将其转换为内部表示，然后通过变换中心（加工）的处理，再沿着输出转换为外部形式离开系统。具有这种特性的加工流就是变换流。

事务流：信息首先沿着输入通路进入系统，事务中心根据输入信息的类型在若干个动作序列（活动流）中选择一个执行，这种信息流称为事务流。

3．程序流程图和盒图

程序流程图和盒图都是用来描述程序的细节逻辑的，其符号如图 8-10 所示。

程序流程图的特点是简单、直观、易学，但它的缺点也正是由于其随意性而使得画出来的流程图容易成为非结构化的流程图。而盒图正是为了解决这一问题设计的，它是一种符合结构化程序设计原则的图形描述工具。

盒图的主要特点是功能域明确、无法任意转移控制、容易确定全局数据和局部数据的作用域、容易表示嵌套关系、可以表示模块的层次结构。但它也带来了一个副作用，那就是修改相对比较困难。

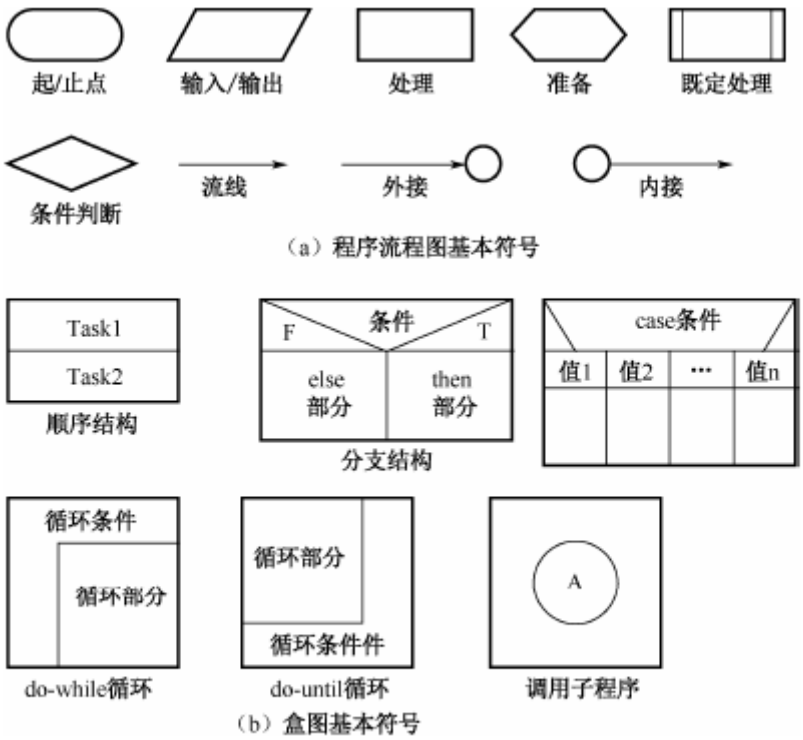


图 8-10　程序流程图和盒图基本符号示意图

4．PAD 和 PDL

PAD 是问题分析图的缩写，它符合自顶向下、逐步求精的原则，也符合结构化程序设计的思想，它最大的特点在于能够很方便地转换为程序语言的源程序代码。

PDL 则是语言描述工具的缩写，它和高级程序语言很相似，也包括数据说明部分和过程部分，还可以带注解等成分，但它是不可执行的。PDL 是一种形式化语言，其控制结构的描述是确定的，但内部的描述语法是不确定的。PDL 通常也被称为伪码。

8.3.3 模块设计

在结构化方法中，模块化是一个很重要的概念，它是将一个待开发的软件分解成为若干个小的简单部分——模块，每个模块可以独立地开发、测试。这是一种复杂问题的“分而治之”原则，其目的是使程序的结构清晰、易于测试与修改。

具体来说，模块是指执行某一特定任务的数据结构和程序代码。通常将模块的接口和功能定义为其外部特性，将模块的局部数据和实现该模块的程序代码称为内部特性。而在模块设计时，最重要的原则就是实现信息隐蔽和模块独立。模块经常具有连续性，也就意味着作用于系统的小变动将导致行为上的小变化，同时规模说明的小变动也将影响到一小部分模块。

1．信息隐蔽原则

信息隐蔽是开发整体程序结构时使用的法则，即将每个程序的成分隐蔽或封装在一个单一的设计模块中，并且尽可能少地暴露其内部的处理。通常将难的决策、可能修改的决策、数据结构的内部连接以及对它所做的操作细节、内部特征码、与计算机硬件有关的细节等隐蔽起来。通过信息隐蔽可以提高软件的可修改性、可测试性和可移植性，它也是现代软件设计的一个关键性原则。

2．模块独立性原则

模块独立是指每个模块完成一个相对独立的特定子功能，并且与其他模块之间的联系最简单。保持模块的高度独立性，也是设计过程中的一个很重要的原则。通常用耦合（模块之间联系的紧密程度）和内聚（模块内部各元素之间联系的紧密程度）两个标准来衡量，设计的目标是高内聚、低耦合。

模块的内聚类型通常可以分为 7 种，根据内聚度从高到低排序，如表 8-2 所示。

表 8-2　模块的内聚类型

内聚类型	描　　述
功能内聚	完成一个单一功能，各个部分协同工作，缺一不可
顺序内聚	处理元素相关，而且必须顺序执行
通信内聚	所有处理元素集中在一个数据结构的区域上
过程内聚	处理元素相关，而且必须按特定的次序执行
瞬时内聚	所包含的任务必须在同一时间间隔内执行（如初始化模块）
逻辑内聚	完成逻辑上相关的一组任务
偶然内聚	完成一组没有关系或松散关系的任务

与此相对应的，模块的耦合性类型通常也分为 7 种，根据耦合度从低到高排序，如表 8-3 所示。

表 8-3　模块的耦合性类型

耦合类型	描　　述
非直接耦合	没有直接联系，互相不依赖对方
数据耦合	借助参数表传递简单数据
标记耦合	一个数据结构的一部分借助于模块接口来传递
控制耦合	模块间传递的信息中包含用于控制模块内部逻辑的信息
外部耦合	与软件以外的环境有关
公共耦合	多个模块引用同一个全局数据区
内容耦合	一个模块访问另一个模块的内部数据 一个模块不通过正常入口转到另一个模块的内部 两个模块有一部分程序代码重叠 一个模块有多个入口

除了满足以上两大基本原则之外，通常在模块分解时还需要注意：保持模块的大小适中，尽可能减少调用的深度，直接调用该模块的个数应该尽量大，但调用其他模块的个数则不宜

过大；保证模块是单入口、单出口的；模块的作用域应该在控制域之内；功能应该是可预测的。

8.4 面向对象的分析与设计

面向对象方法是一种非常实用的软件开发方法，它一出现就受到软件技术人员的青睐，现已成为计算机科学研究的一个重要领域，并逐渐成为软件开发的一种主要方法。面向对象方法以客观世界中的对象为中心，其分析和设计思想符合人们的思维方式，分析和设计的结构与客观世界的实际比较接近，容易被人们接受。在面向对象方法中，分析和设计的界面并不明显，它们采用相同的符号表示，能够方便地从分析阶段平滑地过渡到设计阶段。此外，在现实生活中，用户的需求经常会发生变化，但客观世界的对象及对象间的关系比较稳定，因此用面向对象方法分析和设计的结构也相对比较稳定。

8.4.1 面向对象的基本概念

1．对象和类

对象是系统中用来描述客观事物的一个实体，它由对象标识（名称）、属性（状态、数据、成员变量）和服务（操作、行为、方法）三个要素组成，它们被封装为一个整体，以接口的形式对外提供服务。

在现实世界中，每个实体都是对象，如学生、书籍、收音机等；每个对象都有它的操作，例如书籍的页数，收音机的频道、按钮等属性，以及收音机的切换频道等操作。

而类则是对具有相同属性和服务的一个或一组对象的抽象。类与对象是抽象描述和具体实例的关系，一个具体的对象被称为类的一个实例。在系统设计过程中，类可以分为三种类型，分别是实体类、边界类和控制类。

（1）实体类：实体类映射需求中的每个实体，实体类保存需要存储在永久存储体中的信息，例如，在线教育平台系统可以提取出学员类和课程类，它们都属于实体类。实体类通常都是永久性的，它们所具有的属性和关系是长期需要的，有时甚至在系统的整个生存期都需要。

实体类是对用户来说最有意义的类，通常采用业务领域术语命名，一般来说是一个名词，在用例模型向领域模型的转化中，一个参与者一般对应于实体类。通常可以从 SRS 中的那些与数据库表（需要持久存储）对应的名词着手来找寻实体类。通常情况下，实体类一定有

属性，但不一定有操作。

（2）控制类：控制类是用于控制用例工作的类，一般是由动宾结构的短语（“动词+名词”或“名词+动词”）转化来的名词，例如，用例“身份验证”可以对应于一个控制类“身份验证器”，它提供了与身份验证相关的所有操作。控制类用于对一个或几个用例所特有的控制行为进行建模，控制对象（控制类的实例）通常控制其他对象，因此，它们的行为具有协调性。

控制类将用例的特有行为进行封装，控制对象的行为与特定用例的实现密切相关，当系统执行用例的时候，就产生了一个控制对象，控制对象经常在其对应的用例执行完毕后消亡。通常情况下，控制类没有属性，但一定有方法。

（3）边界类：边界类用于封装在用例内、外流动的信息或数据流。边界类位于系统与外界的交接处，包括所有窗体、报表、打印机和扫描仪等硬件的接口，以及与其他系统的接口。要寻找和定义边界类，可以检查用例模型，每个参与者和用例交互至少要有一个边界类，边界类使参与者能与系统交互。边界类是一种用于对系统外部环境与其内部运作之间的交互进行建模的类。常见的边界类有窗口、通信协议、打印机接口、传感器和终端等。实际上，在系统设计时，产生的报表都可以作为边界类来处理。

边界类用于系统接口与系统外部进行交互，边界对象将系统与其外部环境的变更（例如，与其他系统的接口的变更、用户需求的变更等）分隔开，使这些变更不会对系统的其他部分造成影响。通常情况下，边界类可以既有属性也有方法。

2．继承与泛化

继承是面向对象方法中重要的概念，用来说明特殊类（子类）与一般类（父类）的关系，而通常用泛化来说明一般类与特殊类的关系，也就是说它们是一对多关系。

如图 8-11 所示，“交通工具”是“自行车”和“小轿车”的泛化；“自行车”和“小轿车”从“交通工具”中继承。

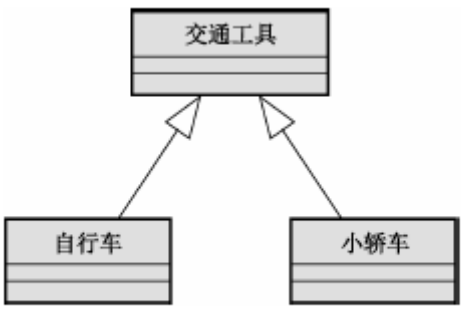


图 8-11　继承与泛化示意图

3．多态与重载

多态（即多种形式）性是指一般类中定义的属性或服务被特殊类继承后，可以具有不同的数据类型或表现出不同的行为，通常是使用重载和改写两项技术来实现的。一般有 4 种不同形式的多态，如表 8-4 所示。

表 8-4　多态类型一览表

多态类型	描　述	示　例
重载 （专用多态）	描述一个函数名称有多种不同实现方式，通常可以在编译时基于类型签名来区分各个重载函数的名称	class OverLaoder{ public void test (int x); public void test (int x, double y); public void test (string x); }

改写
（包含多态）
	是重载的一种特殊情况，只发生在有关父类和子类之间关系中。注：通常签名相同，内容不一样	class Parent{
		public void test (int x); }
		class Child extends Parent{
		public void test (int); }

多态变量 （赋值多态强制多态）	声明为一种类型，但实际上却可以包含另一种类型数值的变量	Parent p=new Child ( );
泛型（模板，参数多态）	它提供了一种创建通用工具的方法，可以在特定的场合将其特化	template <class T> T max (T a, T b) {　if (a< b) return b; Return a; }

注 1：重载也称为过载、重置；

注 2：参数多态和包含多态称为通用多态，重载多态和强制多态称为特定多态。

希赛教育专家提示：虽然重载和改写都是在多种潜在的函数体中，选择和调用某一个函数或方法并对其进行执行，但它们的本质区别在于：重载是编译时执行的（静态绑定），而改写则是运行时选择的（动态绑定）。

4．模板类

也称为类属类，它用来实现参数多态机制。一个类属类是关于一组类的一个特性抽象，它强调的是这些类的成员特征中与具体类型无关的那些部分，而用变元来表示与具体类型有关的那些部分。

5．消息和消息通信

消息就是向对象发出的服务请求，它通常包括提供服务的对象标识、消息名、输入信息和回答信息。消息通信则是面向对象方法学中的一个重要原则，它与对象的封装原则密不可分，为对象间提供了唯一合法的动态联系的途径。

8.4.2 面向对象分析

面向对象分析的目标是开发一系列模型，这些模型描述计算机软件，当它工作时以满足一组客户定义的需求。对象技术的流行，演化出了数十种不同的 OOA 方法，每个方法都引入了一个产品或系统分析的过程、一组过程演化的模型及使软件工程师能够以一致的方式创建每个模型的符号体系。其中比较流行的方法包括 OMT、OOA、OOSE、Booch 方法等，而 OMT、OOSE、Booch 最后则统一成为 UML。

1．OOA/OOD 方法

这是由 Peter Coad 和 Edward Yourdon 提出的，OOA 模型中包括主题、对象类、结构、属性和服务 5 个层次，需经过标识对象类、标识结构与关联（包括继承、聚合、组合、实例化等）、划分主题、定义属性、定义服务 5 个步骤来完成整个分析工作。

OOD 中将继续贯穿 OOA 中的 5 个层次和 5 个活动，它由人机交互部件、问题域部件、任务管理部件、数据管理部件 4 个部分组成，其主要的活动就是这 4 个部件的设计工作。

设计问题域部分：OOA 的结果恰好是 OOD 的问题域部件，分析的结果在 OOD 中可以被改动或增补，但基于问题域的总体组织框架是长时间稳定的；

设计人机交互部件：人机交互部件在上述结果中加入人机交互的设计和交互的细节，包括窗口和输出报告的设计。可以用原型来帮助实际交互机制进行开发和选择；

设计任务管理部分：这部分主要是识别事件驱动任务，识别时钟驱动任务，识别优先任务和关键任务，识别协调者，审查每个任务并定义每个任务。

设计数据管理部分：数据管理部分提供了在数据管理系统中存储和检索对象的基本结构，其目的是隔离数据管理方法对其他部分的影响。

2．Booch 方法

Booch 认为软件开发是一个螺旋上升的过程，每个周期中包括标识类和对象、确定类和对象的含义、标识关系、说明每个类的接口和实现 4 个步骤。它的模型中主要包括如表 8-5 所示的几种图形。

表 8-5　Booch 方法的模型概述

	静态模型	动态模型
逻辑模型	类图 对象图	状态转换图 时序图
物理模型	模块图 进程图	

Booch 方法的开发过程是一个迭代的、渐进式的系统开发过程，它可以分为宏过程和微过程两类。宏过程用于控制微过程，是覆盖几个月或几周所进行的活动，它包括负责建立核心需求的概念化，为所期望的行为建立模型的分析，建立架构的设计，形成实现的进化，以及管理软件交付使用的维护等 5 个主要活动。

而微过程则基本上代表了开发人员的日常活动，它由 4 个重要、没有顺序关系的步骤组成：在给定的抽象层次上识别出类和对象，识别出这些类和对象的语义，识别出类间和对象间的关系，实现类和对象。

3．OMT 方法

OMT 是对象建模技术的缩写，它是由 Jam Rambaugh 及其同事合作开发的，它主要用于分析、系统设计和对象设计。包括对象模型（静态的、结构化的系统的“数据”性质，通常采用类图）、动态模型（瞬时的、行为化的系统“控制”性质，通常使用状态图）和功能模型（表示变化的系统的“功能”性质，通常使用数据流图）。OMT 方法的三大模型如表 8-6 所示。

表 8-6　OMT 方法的三大模型

模　型	说　明	主要技术
对象模型	描述系统中对象的静态结构、对象之间的关系、属性、操作。它表示静态的、结构上的、系统的“数据”特征	对象图
动态模型	描述与时间和操作顺序有关的系统特征，如激发事件、事件序列、确定事件先后关系的状态。它表示瞬时、行为上的、系统的“控制”特征	状态图
功能模型	描述与值的变换有关的系统特征：功能、映射、约束和函数依赖	数据流图

4.OOSE 方法

OOSE 是面向对象软件工程的缩写，它是由 Ivar Jacobson 提出的。它在 OMT 的基础上，对功能模型进行了补充，提出了“用例”的概念，最终取代数据流图进行需求分析和建立功能模型。

8.4.3 统一建模语言

统一建模语言（Unified Modeling Language，UML）是用于系统的可视化建模语言，它将 OMT、OOSE 和 Booch 方法中的建模语言和方法有机地融合在一起，是国际统一的软件建模标准。虽然它源于 OO 软件系统建模领域，但由于其内建了大量扩展机制，也可以应用于更多的领域中，例如工作流程、业务领域等。

1．UML 是什么

UML 是一种语言：UML 在软件领域中的地位与价值就像“1、2、3、+、 、…”等符号在数学领域中的地位一样。它为软件开发人员之间提供了一种用于交流的词汇表和一种用于软件蓝图的标准语言。

UML 是一种可视化语言：UML 只是一组图形符号，它的每个符号都有明确语义，是一种直观、可视化的语言。

UML 是一种可用于详细描述的语言：UML 所建的模型是精确的、无歧义和完整的，因此适合于对所有重要的分析、设计和实现决策进行详细描述。

UML 是一种构造语言：UML 虽然不是一种可视化的编程语言，但其与各种编程语言直接相连，而且有较好的映射关系，这种映射允许进行正向工程、逆向工程。

UML 是一种文档化语言：它适合于建立系统架构及其所有的细节文档。

2．UML 的结构

UML 由构造块、公共机制和架构三个部分组成。

（1）构造块。构造块也就是基本的 UML 建模元素（事物）、关系和图。

建模元素：包括结构事物（类、接口、协作、用例、活动类、组件、节点等）、行为事物（交互、状态机）、分组事物（包）、注释事物。

关系：包括关联关系、依赖关系、泛化关系、实现关系。

图：UML 2.0 包括 14 种不同的图，分为表示系统静态结构的静态模型（包括类图、对象图、包图、构件图、部署图、制品图），以及表示系统动态结构的动态模型（包括对象图、用例图、顺序图、通信图、定时图、状态图、活动图、交互概览图）。

（2）公共机制。公共机制是指达到特定目标的公共 UML 方法，主要包括规格说明、修饰、公共分类和扩展机制 4 种。

规格说明：规格说明是元素语义的文本描述，它是模型的重要组成部分。

修饰：UML 为每一个模型元素设置了一个简单的记号，还可以通过修饰来表达更多的信息。

公共分类：包括类元与实体（类元表示概念，而实体表示具体的实体）、接口和实现（接口用来定义契约，而实现就是具体的内容）两组公共分类。

扩展机制：包括约束（添加新规则来扩展元素的语义）、构造型（用于定义新的 UML 建模元素）、标记值（添加新的特殊信息来扩展模型元素的规格说明）。

（3）架构。UML 对系统架构的定义是：系统的组织结构，包括系统分解的组成部分、它们的关联性、交互、机制和指导原则，这些提供系统设计的信息。而具体来说，就是指 5 个系统视图。

逻辑视图：以问题域的语汇组成的类和对象集合。

进程视图：可执行线程和进程作为活动类的建模，它是逻辑视图的一次执行实例。

实现视图：对组成基于系统的物理代码的文件和组件进行建模。

部署视图：把组件物理地部署到一组物理的、可计算的节点上。

用例视图：最基本的需求分析模型。

3．用例图基础

用例是什么呢？Ivar Jacobson 是这样描述的:“用例实例是在系统中执行的一系列动作，这些动作将生成特定参与者可见的价值结果。一个用例定义一组用例实例。”首先，从定义中得知用例是由一组用例实例组成的，用例实例也就是常说的“使用场景”，就是用户使用系统的一个实际的、特定的场景。其次，可以知道，用例应该给参与者带来可见的价值，这点很关键。最后，用例是在系统中的。

而用例模型描述的是外部参与者所理解的系统功能。用例模型用于需求分析阶段，它的建立是系统开发者和用户反复讨论的结果，表明了开发者和用户对需求规格达成的共识。图 8-12 就是一个用例图的例子。

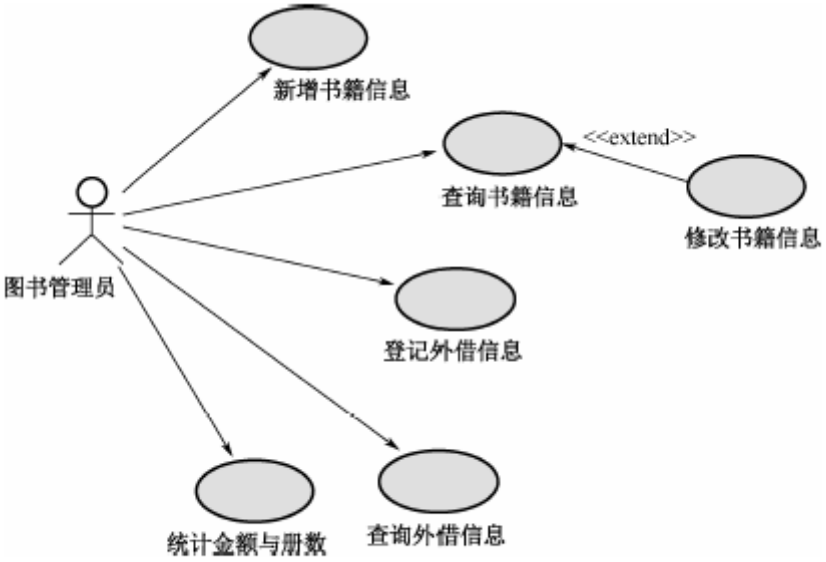


图 8-12　用例图示例

（1）参与者。参与者代表与系统接口的任何事物或人，它是指代表某一种特定功能的角色，因此，参与者都是虚拟的概念。在 UML 中，用一个小人表示参与者。

图 8-12 中的“图书管理员”就是参与者。对于该系统来说，可能可以充当图书管理员角色的有多个人，由于他们对系统均起着相同的作用，扮演相同的角色，因此只用一个参与者来表示。切忌不要为每一个可能与系统交互的真人画出一个参与者。

（2）用例。用例是对系统行为的动态描述，它可以促进设计人员、开发人员与用户的沟通，理解正确的需求，还可以划分系统与外部实体的界限，是系统设计的起点。在识别出参与者之后，可以使用下列问题帮助识别用例。

每个参与者的任务是什么？

有参与者将要创建、存储、修改、删除或读取系统中的信息吗？

什么用例会创建、存储、修改、删除或读取这个信息？

参与者需要通知系统外部的突然变化吗？

需要通知参与者系统中正在发生的事情吗？

什么用例将支持和维护系统？

所有的功能需求都对应到用例中了吗？

系统需要何种输入/输出？输入从何处来？输出到何处？

当前运行系统的主要问题是什么？

（3）包含和扩展。两个用例之间的关系可以主要概括为两种情况。一种是用于重用的包含关系，用构造型<<include>>或者<<use>>表示；另一种是用于分离出不同的行为，用构造型<<extend>>表示。

包含关系：当可以从两个或两个以上的原始用例中提取公共行为，或者发现能够使用一个组件来实现某一个用例的部分功能是很重要的事时，应该使用包含关系来表示。所提取出来的公共行为称为抽象用例。包含关系的例子如图 8-13 所示。

扩展关系：如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种事情。可以将这个用例分为一个主用例和一个或多个辅用例，描述可能更加清晰。扩展关系的例子如图 8-14 所示。

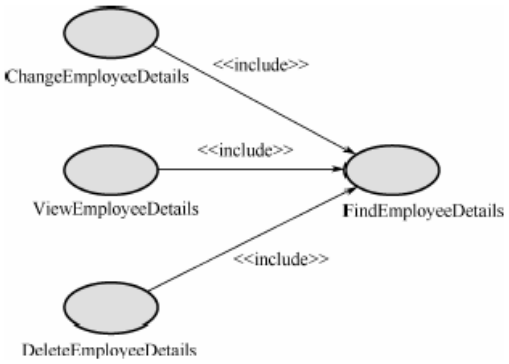


图 8-13　包含关系示例

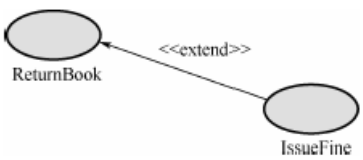


图 8-14　扩展关系示例

希赛教育专家提示：此处介绍的包含和扩展关系属于用例之间所特有的关系，因为用例

也是 UML 的一种结构事物，因此，事物之间的 4 种基本关系对用例也是适用的。

4．类图和对象图基础

在面向对象建模技术中，将客观世界的实体映射为对象，并归纳成一个个类。类、对象和它们之间的关联是面向对象技术中最基本的元素。对于一个想要描述的系统，其类模型和对象模型揭示了系统的结构。在 UML 中，类和对象模型分别由类图和对象图表示。类图技术是 OO 方法的核心。图 8-15 是一个类图的实例。

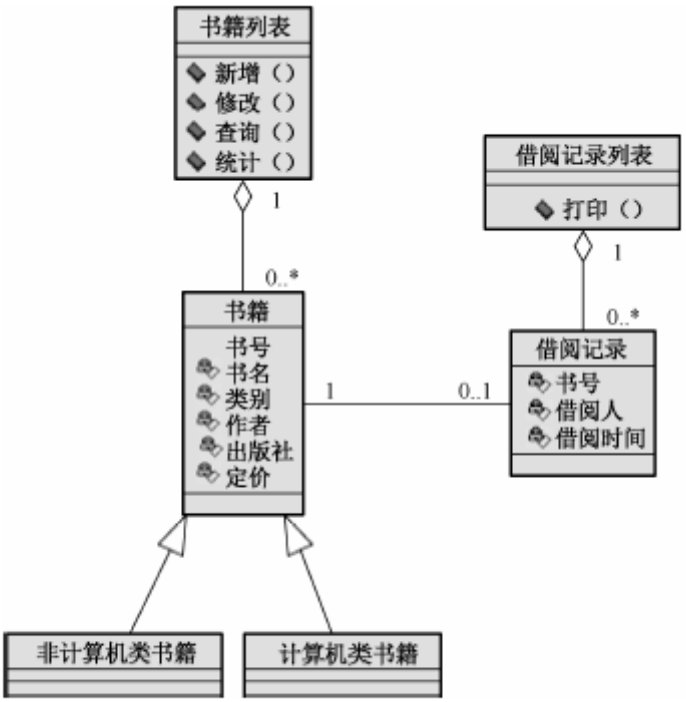


图 8-15　类图示例

（1）类和对象。对象与人们对客观世界的理解相关。人们通常用对象描述客观世界中某个具体的实体。所谓类是对一类具有相同特征的对象的描述。而对象是类的实例。在 UML 中，类的可视化表示为一个划分成三个格子的长方形（下面两个格子可省略）。图 8-15 中，“书籍”、“借阅记录”等都是一个类。

类的获取和命名：最顶部的格子包含类的名字。类的命名应尽量用应用领域中的术语，应明确、无歧义，以利于开发人员与用户之间的相互理解和交流。

类的属性：中间的格子包含类的属性，用以描述该类对象的共同特点。该项可省略。图 8-15 中“书籍”类有“书名”、“书号”等属性。UML 规定类的属性的语法为：　“可见性 属性名：类型 = 默认值 {约束特性}”。

可见性包括 Public、Private 和 Protected，分别用+、-、#号表示。

类型表示该属性的种类：它可以是基本数据类型，例如整数、实数、布尔型等，也可以

是用户自定义的类型。一般它由所涉及的程序设计语言确定。约束特性则是用户对该属性性质的一个约束说明。例如“{只读}”说明该属性是只读属性。

类的操作（Operation）：该项可省略。操作用于修改、检索类的属性或执行某些动作。操作通常也被称为功能，但是它们被约束在类的内部，只能作用到该类的对象上。操作名、返回类型和参数表组成操作界面。UML 规定操作的语法为：“可见性：操作名（参数表）：返回类型 {约束特性}”。

类图描述了类和类之间的静态关系。定义了类之后，就可以定义类之间的各种关系了。

（2）类之间的关系。在建立抽象模型时，会发现很少有类会单独存在，大多数都将会以某种方式互相协作，因此还需要描述这些类之间的关系。关系是事物间的连接，在面向对象建模中，有 4 个很重要的关系。

① 依赖关系。有两个元素 X、Y，如果修改元素 X 的定义可能会引起对另一个元素 Y 的定义的修改，则称元素 Y 依赖于元素 X。在 UML 中，使用带箭头的虚线表示依赖关系。

在类中，依赖由多种原因引起，如：一个类向另一个类发消息；一个类是另一个类的数据成员；一个类是另一个类的某个操作参数。如果一个类的界面改变，它发出的任何消息可能不再合法。

② 泛化关系。泛化关系描述了一般事物与该事物中的特殊种类之间的关系，也就是父类与子类之间的关系。继承关系是泛化关系的反关系，也就是说子类是从父类中继承的，而父类则是子类的泛化。在 UML 中，使用带空心箭头的实线表示，箭头指向父类。

在 UML 中，对泛化关系有 3 个要求：

子类应与父类完全一致，父类所具有的关联、属性和操作，子类都应具有。

子类中除了与父类一致的信息外，还包括额外的信息。

可以使用子父类实例的地方，也可以使用子类实例。

③ 关联关系。关联表示两个类之间存在某种语义上的联系。例如，一个人为一家公司工作，一家公司有许多办公室。就认为人和公司、公司和办公室之间存在某种语义上的联系。

关联关系提供了通信的路径，它是所有关系中最通用的、语义最弱的。在 UML 中，用一条实线来表示关联关系。

聚合关系：聚合是一种特殊形式的关联。聚合表示类之间的关系是整体与部分的关系。例如一辆轿车包含四个车轮、一个方向盘、一个发动机和一个底盘，就是聚合的一个例子。在 UML 中，用一个带空心菱形的实线表示，空心菱形指向的是代表“整体”的类。

组合关系：如果聚合关系中的表示“部分”的类的存在，与表示“整体”的类有着紧密的关

系，例如“公司”与“部门”之间的关系，那么就应该使用“组合”关系来表示。在 UML 中，用带有实心菱形的实线表示，菱形指向的是代表“整体”的类。

希赛教育专家提示：聚合关系是指部分与整体的生命周期可以不相同，而组合关系则是指部分与整体的生命周期是相同的。

④ 实现关系。实现关系是用来规定接口和实现接口的类或组件之间的关系的。接口是操作的集合，这些操作用于规定类或组件的服务。在 UML 中，用一个带空心箭头的虚线表示。

（3）多重性问题。重复度又称多重性，多重性表示为一个整数范围 n…m，整数 n 定义所连接的最少对象的数目，而 m 则为最多对象数（当不知道确切的最大数时，最大数用*号表示）。最常见的多重性有：0…1；0…*；1…1；1…*；*。

多重性是用来说明关联的两个类之间的数量关系的，例如：

书与借书记录之间的关系，就应该是 1 对 0…1 的关系，也就是一本书可以有 0 个或 1 个借书记录。

经理与员工之间的关系，则应为 1 对 0…*的关系，也就是一个经理可以领导 0 个或多个员工。

学生与选修课程之间的关系，就可以表示为 0…*对 1…*的关系，也就是一个学生可以选择 1 门或多门课程，而一门课程可以有 0 个或多个学生选修。

（4）类图。对于软件系统，其类模型和对象模型类图描述类和类之间的静态关系。与数据模型不同，它不仅显示了信息的结构，同时还描述了系统的行为。类图是定义其他图的基础。

（5）对象图。UML 中对象图与类图具有相同的表示形式。对象图可以看作是类图的一个实例。对象是类的实例；对象之间的链（Link）是类之间的关联的实例。对象与类的图形表示相似，均为划分成两个格子的长方形（下面的格子可省略）。上面的格子是对象名，对象名下有下画线；下面的格子记录属性值。链的图形表示与关联相似。对象图常用于表示复杂类图的一个实例。

5．交互图基础

交互图是表示各组对象如何依某种行为进行协作的模型。通常可以使用一个交互图来表示和说明一个用例的行为。在 UML 中，包括 3 种不同形式的交互图，强调对象交互行为顺序的顺序图，强调对象协作的通信图（UML1.X 版本中称为“协作图”），强调消息的具体时间的定时图，它们之间没有什么本质不同，只是排版不尽相同而已。

（1）顺序图。顺序图用来描述对象之间动态的交互关系，着重体现对象间消息传递的时间顺序。顺序图允许直观地表示出对象的生存期，在生存期内，对象可以对输入消息做出响应，并且可以发送信息。图 8-16 是一个顺序图的示例。

如图 8-16 所示，顺序图存在两个轴，水平轴表示不同的对象，即图中的 Client、Factory、Product 等；而垂直轴表示时间，表示对象及类的生命周期。

对象间的通信通过在对象的生命线间画消息来表示。消息的箭头指明消息的类型。顺序图中的消息可以是信号、操作调用或类似于 C++中的 RPC（Remote Procedure Calls）和 Java 中的 RMI（Remote Method Invocation）。当收到消息时，接收对象立即开始执行活动，即对象被激活了。通过在对象生命线上显示一个细长矩形框来表示激活。

消息可以用消息名及参数来标识，消息也可带有顺序号。消息还可带有条件表达式，表示分支或决定是否发送消息。如果用于表示分支，则每个分支是相互排斥的，即在某一时刻仅可发送分支中的一个消息。

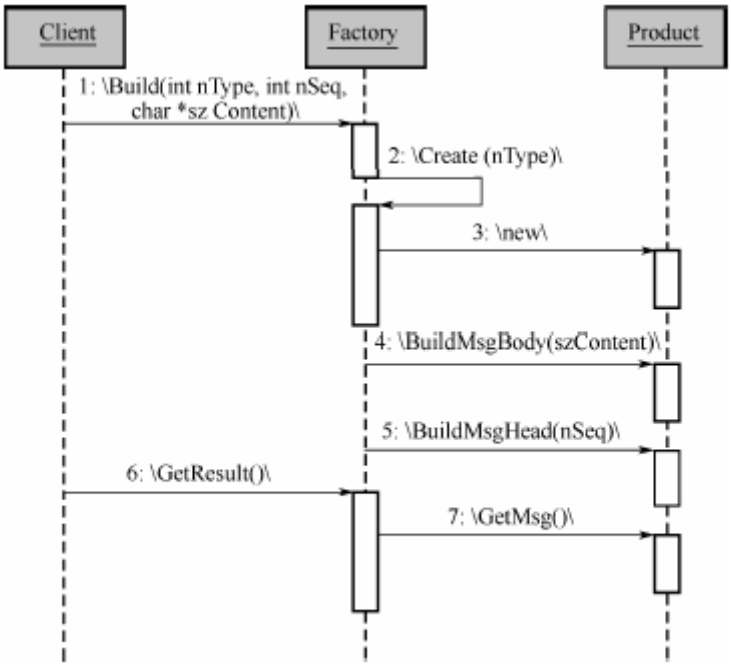


图 8-16　顺序图示例

（2）通信图。通信图用于描述相互合作的对象间的交互关系和链接关系。虽然顺序图和通信图都用来描述对象间的交互关系，但侧重点不一样。顺序图着重体现交互的时间顺序，通信图则着重体现交互对象间的静态链接关系。图 8-17 就是与图 8-16 相对应的通信图，可以从图 8-17 中很明显地发现它与顺序图之间的异同点。

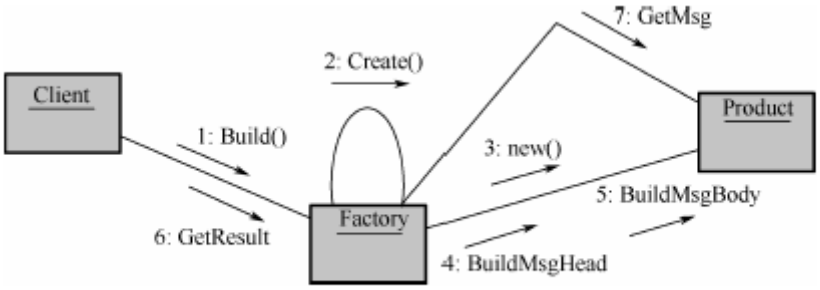


图 8-17　通信图示例

（3）定时图。如果要表示的交互具有很强的时间特性（例如，现实生活中的电子工程、实时控制等系统中），在 UML 1.X 中是无法有效地表示出来的。而在 UML 2.0 中引入了一种新的交互图来解决这类问题，这就是着重表示定时约束的定时图。

根据 UML 的定义，定时图实际上是一种特殊形式的顺序图（即一种变化），它与顺序图的区别主要体现在以下几个方面。

坐标轴交换了位置，改为从左到右来表示时间的推移。

用生命线的“凹下凸起”来表示状态的变化，每个水平位置代表一种不同的状态，状态的顺序可以有意义、也可以没有意义。

生命线可以跟在一根线后面，在这根线上显示一些不同的状态值。

可以显示一个度量时间值的标尺，用刻度来表示时间间隔。

图 8-18 是一个定时图的实际例子，其中小黑点加曲线表示的是注释。它用来表示一个电子门禁系统的控制逻辑，该门禁系统包括门（物理的门）、智能读卡器（读取用户的智能卡信息）、处理器（用来处理是否开门的判断）。

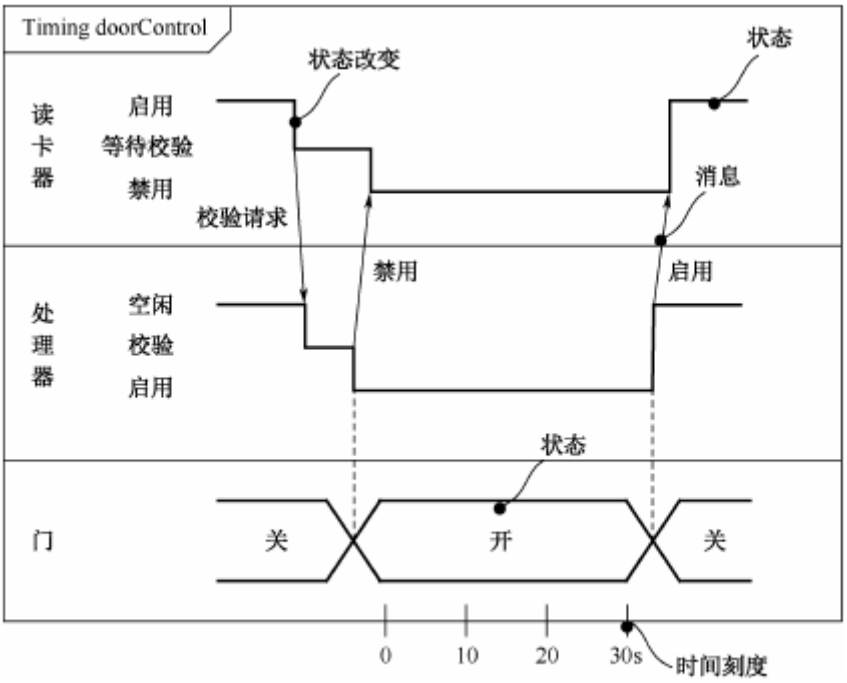


图 8-18　定时图示例

在这个例子中，它所表示的意思是一开始读卡器是启用的（等用户来刷卡）、处理器是空闲的（没有验证的请求）、门是关的；接着，当用户刷卡时，读卡器就进入了“等待校验”的状态，并发一个消息给处理器，处理器就进入了校验状态。如果校验通过，就发送一个“禁用”消息给读卡器（因为门开的时候，读卡器就可以不工作），使读卡器进入禁用状态；并且自己转入启用状态，这时门的状态变成了“开”。而门“开”了 30 秒钟（根据时间刻度得知）之后，处理器将会把它再次“关”上，并且发送一个“启用”消息给读卡器（门关了，读卡器开始重新工作），这时读卡器再次进入启用状态，而处理器已经又回到了空闲状态。

从上面的例子中，不难看出定时图所包含的图元并不多，主要包括生命线、状态、状态变迁、消息、时间刻度，可以根据自身的需要来使用它。

6．状态图基础状态图

用来描述一个特定对象的所有可能状态及其引起状态转移的事件。大多数面向对象技术都用状态图表示单个对象在其生命周期中的行为。一个状态图包括一系列的状态及状态之间的转移。图 8-19 是一个数码冲印店的订单状态图实例。

状态图包括以下部分。

状态：又称为中间状态，用圆角矩形框表示；

初始状态：又称为初态，用一个黑色的实心圆圈表示，在一张状态图中只能够有一个初始状态；

结束状态：又称为终态，在黑色的实心圆圈外面套上一个空心圆，在一张状态图中可能有多个结束状态；

状态转移：用箭头说明状态的转移情况，并用文字说明引发这个状态变化的相应事件是什么。

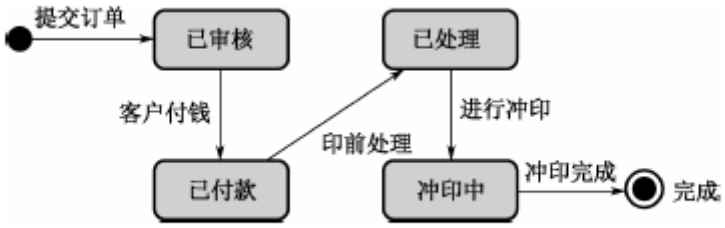


图 8-19　状态图示例

一个状态也可能被细分为多个子状态，那么如果将这些子状态都描绘出来的话，那这个状态就是复合状态。

状态图适合用于表述在不同用例之间的对象行为，但并不适合用于表述包括若干用例协作的对象行为。通常不会需要对系统中的每一个类绘制相应的状态图，而通常会在业务流程、控制对象、用户界面的设计方面使用状态图。

7．活动图基础

活动图的应用非常广泛，它既可用来描述操作（类的方法）的行为，也可以描述用例和对象内部的工作过程。活动图是由状态图变化而来的，它们各自用于不同的目的。活动图依据对象状态的变化来捕获动作（将要执行的工作或活动）与动作的结果。活动图中一个活动结束后将立即进入下一个活动（在状态图中状态的变迁可能需要事件的触发）。

（1）基本活动图。图 8-20 给出了一个基本活动图的例子。

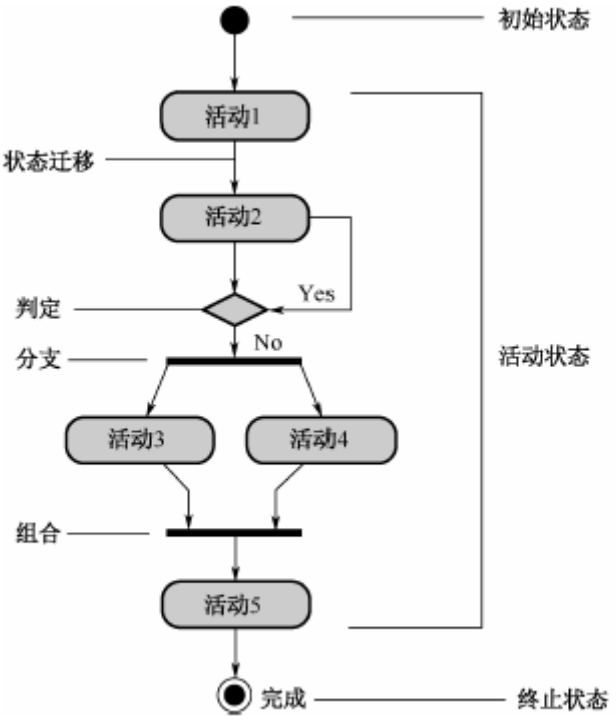


图 8-20　活动图示例

如图 8-20 所示，活动图与状态图类似，包括了初始状态、终止状态，以及中间的活动状态，每个活动之间，也就是一种状态的变迁。在活动图中，还引入了以下几个概念。

判定：说明基于某些表达式的选择性路径，在 UML 中使用菱形表示。

分支与组合：由于活动图建模时，经常会遇到并发流，因此在 UML 中引入了如上图 8-20 所示的粗线来表示分支和组合。

（2）带泳道的活动图。在前面说明的基本活动图中，虽然能够描述系统发生了什么，但没有说明该项活动由谁来完成。而针对 OOP 而言，这就意味着活动图没有描述出各个活动由哪个类来完成。要想解决这个问题，可以通过泳道来解决这一问题。它将活动图的逻辑描述与顺序图、协作图的责任描述结合起来。图 8-21 是一个使用了泳道的例子。

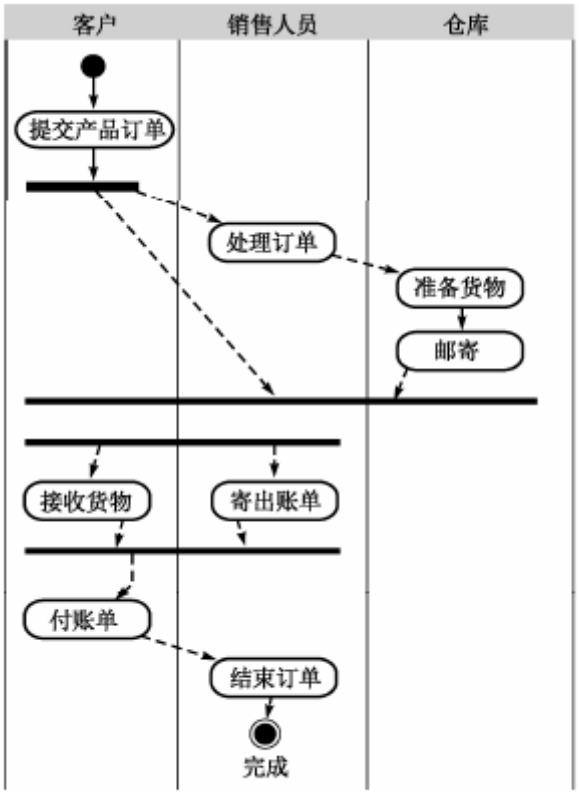


图 8-21　带泳道活动图示例

（3）对象流。在活动图中可以出现对象。对象可以作为活动的输入或输出，对象与活动间的输入/输出关系由虚线箭头来表示。如果仅表示对象受到某一活动的影响，则可用不带箭头的虚线来连接对象与活动。

（4）信号。在活动图中可以表示信号的发送与接收，分别用发送和接收标识来表示。发送和接收标识也可与对象相连，用于表示消息的发送者和接收者。

8．构件图基础

构件图是面向对象系统的物理方面进行建模要用的两种图之一。它可以有效地显示一组构件，以及它们之间的关系。构件图中通常包括构件、接口及各种关系。图 8-22 就是一个构件图的例子。

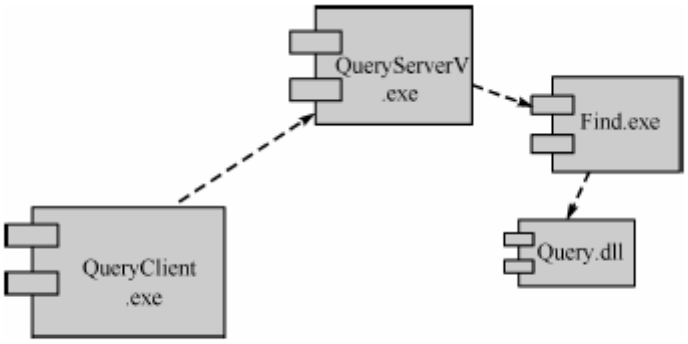


图 8-22　构件图示例

通常构件指的是源代码文件、二进制代码文件和可执行文件等。而构件图就是用来显示编译、链接或执行时构件之间的依赖关系的。例如，在图 8-22 中，就是说明 QueryClient.exe 将通过调用 QueryServer.exe 来完成相应的功能，而 QueryServer.exe 则需要 Find.exe 的支持， Find.exe 在实现时调用了 Query.dll。

通常来说，可以使用构件图完成以下工作。

对源代码进行建模：这样可以清晰地表示出各个不同源程序文件之间的关系。

对可执行体的发布建模：如图 8-22 所示，将清晰地表示出各个可执行文件、DLL 文件之间的关系。

对物理数据库建模：用来表示各种类型的数据库、表之间的关系。

对可调整的系统建模：例如对应用了负载均衡、故障恢复等系统的建模。

在绘制构件图时，应该注意侧重于描述系统的静态实现视图的一个方面，图形不要过于简化，应该为构件图取一个直观的名称，在绘制时避免产生线的交叉。

9．部署图基础

部署图，也称为实施图，它和构件图一样，是面向对象系统的物理方面建模的两种图之一。构件图是说明构件之间的逻辑关系，而部署图则是在此基础上更进一步地描述系统硬件的物理拓扑结构及在此结构上执行的软件。部署图可以显示计算结点的拓扑结构和通信路径、结点上运行的软件构件，常用于帮助理解分布式系统。

图 8-23 就是与图 8-22 对应的部署图，这样的图示可以使系统的安装、部署更为简单。在部署图中，通常包括以下一些关键的组成部分。

（1）节点和连接。节点代表一个物理设备及其上运行的软件系统，如一台 UNIX 主机、一个 PC 终端、一台打印机、一个传感器等。

如图 8-23 所示，“客户端：个人 PC”和“服务器”就是两个节点。在 UML 中，使用

一个立方体表示一个节点，节点名放在左上角。节点之间的连线表示系统之间进行交互的通信路径，在 UML 中被称为连接。通信类型则放在连接旁边的“《》”之间，表示所用的通信协议或网络类型。

（2）构件和接口。在部署图中，构件代表可执行的物理代码模块，如一个可执行程序。逻辑上它可以与类图中的包或类对应。例如，在图 8-23 中，“服务器”结点中包含“QueryServer.exe”、“Find.exe”和“Query.dll”3 个构件。

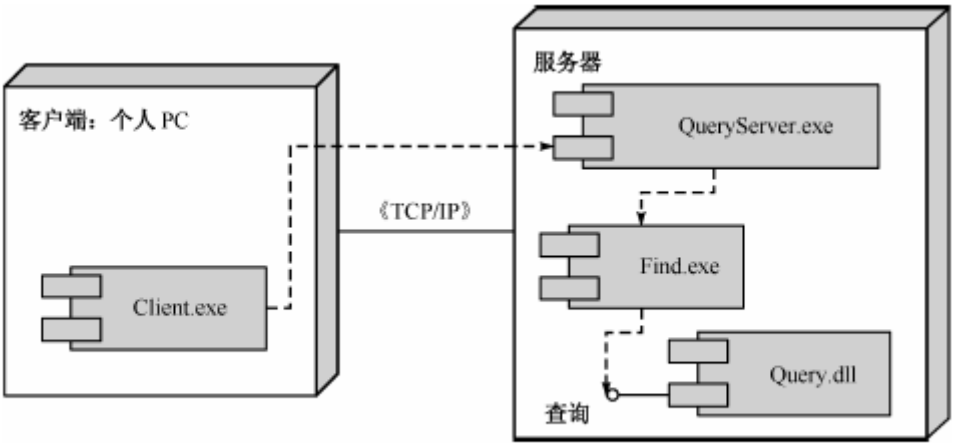


图 8-23 部署图示例

在面向对象方法中，类和构件等元素并不是所有的属性和操作都对外可见。它们对外提供了可见操作和属性，称之为类和构件的接口。界面可以表示为一头是小圆圈的直线。在图 8-23 中，“Query.dll”构件提供了一个“查询”接口。

8.5 用户界面设计

接口设计主要包括三个方面的内容：一是设计软件构件间的接口；二是设计模块和其他非人的信息生产者和消费者（如外部实体）的接口；三是人（如用户）和计算机间界面设计。

软件构件间接口的设计与架构的设计紧密相关，而设计模块和外部实体的接口则与详细设计相关，人机界面接口是相当容易被忽视的环节，在此就对其重点内容进行一个概要性描述。

8.5.1 用户界面设计的原则

用户界面设计必须考虑软件使用者的体力和脑力，根据 Theo Mandel 的总结，设计时必须遵从三个黄金法则。

置用户于控制之下：具体来说就是以不强迫用户进入不必要的或不希望的动作的方式来定义交互模式、提供灵活的交互、允许用户交互可以被中断和撤销、当技能级别增长时可以使交互流水化并允许定制交互、使用户隔离内部技术细节、设计应允许用户和出现在屏幕上的对象直接交互。

减少用户的记忆负担：具体来说就是减少对短期记忆的要求、建立有意义的默认、定义直觉性的捷径、界面的视觉布局应该基于对真实世界的隐喻、以不断进展的方式提示信息。

保持界面的一致：具体来说，就是允许用户将当前任务放入有意义的语境、在应用系列内保持一致性，如果过去的交互模型已经建立了用户期望，除非有不得已的理由，否则不要改变它。

除此之外，还应该考虑表 8-7 所示的设计原则。

表 8-7　用户界面设计原则

原　　则	描　　述
用户熟悉	界面所使用的术语和概念应该来自于用户的经验，这些用户是将要使用系统最多的人
意外最小化	永远不要让用户对系统的行为感到吃惊
可恢复性	界面应该有一种机制来允许用户从错误中恢复
用户指南	在错误发生时，界面应该提供有意义的反馈，并有上下文感知能力的用户帮助功能
用户差异性	界面应该为不同类型用户提供合适的交互功能

8.5.2 用户界面设计过程

用户界面的设计过程也应该是迭代的，它通常包括 4 个不同的框架活动，如图 8-24 所示。

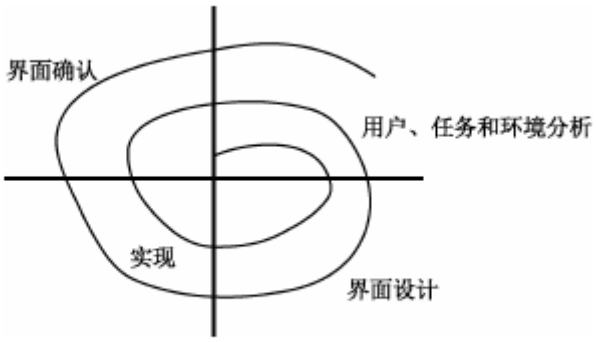


图 8-24　用户界面设计过程

（1）用户、任务和环境分析：着重于分析将和系统交互的用户的特点。记录下技术级别、业务理解及对新系统的一般感悟，并定义不同的用户类别。然后对用户将要完成什么样的任务进行详细的标识和描述。最后对用户的物理工作环境进行了解与分析。

（2）界面设计：主要包括建立任务的目标和意图，为每个目标或意图制定特定的动作序列，按在界面上执行的方式对动作序列进行规约，指明系统状态，定义控制机制，指明控制机制如何影响系统状态，指明用户如何通过界面上的信息来解释系统状态。

（3）实现：就是根据界面设计进行实现，前期可以通过原型工具来快速实现，减少返工的工作量。

（4）界面确认：界面实现后就可以进行一些定性和定量的数据收集，以进行界面的评估，以调整界面的设计。

8.6 工作流设计

工作流技术的发展，经过多年的努力，取得了一定的成果。但在实际应用中，应用的企业还是较少，应用的范围窄，效果不理想。

流程的设计是对设计者更高的挑战，现实中对计算机所管理的流程需要灵活的定义、方便的路径修改、容易使用，可是这几个目标是矛盾的。更严重的是，如何分析现实中的流程本身就是个困难的问题，更不用谈如何来设计实现了。流程设计的主要困难实际上也就是软件的主要困难：现实复杂性。

任何对现实的描述（图形也罢，文字也罢）都是不完美的，“不识庐山真面目”是设计面临的共同难题。设计者不得不意识到所有的流程模型都是对现实的简化，计算机只根据确定的信息作判断，而现实中的流程存在大量的不确定性，虽然计算机专家们自信地告诉企业管理者这是管理的问题，信誓旦旦地保证可以用计算机系统来“完善”企业的管理，但他们似乎没有意识到企业管理已经发展了几百年，而计算机还没有百年的历史。

人们常常抱怨计算机系统的流程设计太过刻板，因为许多时候，标准流程是先于应用构造的且由一些集中的权威强制执行的，所以这种刻板性是不可避免的。同时，对参与者而言缺乏自由度导致工作流管理系统显得很不友好。结果是它们经常被忽略或绕过，甚至最终被放弃。

另外的困难是：对于流程处理，不但名称众多，例如，动态模型、工作流等，而且对流程的定义也是千姿百态。面对这些困难，设计者无疑需要巨大的勇气来进行流程设计。

8.6.1 工作流设计概述

限于篇幅，这里只列出工作流管理联盟对于工作流的定义：“工作流是一类能够完全或

者部分自动执行的经营过程，根据一系列过程规则、文档、信息或任务在不同的执行者之间传递、执行”。

（1）工作流。简单地说，工作流是现实中的具体工作从开始到结束过程的抽象和概括。这个过程包括了众多因素：任务顺序、路线规则、时间时限约束等。

（2）流程定义。流程定义是指对业务过程的形式化表示，它定义了过程运行中的活动和所涉及的各种信息。这些信息包括过程的开始和完成条件、构成过程的活动及进行活动间导航的规则、用户所需要完成的任务、可能被调用的应用、工作流间的引用关系，以及工作流数据的定义。这个定义的过程可能是由设计者用纸和笔来完成的，但越来越多的设计者倾向于使用流程定义工具来完成这个工作。

（3）流程实例。也常常称为工作，是一个流程定义的运行实例。例如客户的一次订购过程，客服中心受理的一次客户投诉过程等。

（4）工作流管理系统。和数据库管理系统类似，是一个软件系统。这个程序存储流程的定义，按照所使用的流程定义来触发流程状态的改变，推动流程的运转。这个推动的依据常常称为工作流引擎。

（5）流程定义工具。同样是一套软件系统，这个软件和工作流管理系统的关系就如同数据库设计工具和数据库管理系统的关系一样。它可能是独立的软件，也可能是工作流管理系统的一部分。如前所说，设计者常常使用流程定义工具来完成流程定义的工作。它提供一些常用的工作流元素素材，以提高设计者的效率。

（6）参与者。回答业务流程中“谁”这个问题。它可以是具体的人或者角色（企业内部有特别共同作用的多个人），也可以是自动化系统，甚至是其他系统。

（7）活动。活动是流程定义中的一个元素，一次活动可能改变流程处理数据的内容、流程的状态，并可能将流程推动到其他活动中去。活动可以由人来完成，也可以是系统自动的处理过程，典型的自动处理是当活动超过了这个活动可以容忍的时限时，自动过程将向流程定义中指定的参与者发出一条消息。

（8）活动所有者。参与者之一，他有权决定该活动是否结束，当他决定活动结束时，将活动推动到其他活动中，可能是下一个活动，也可能是前一个活动。

（9）工作所有者。工作所有者是有权整体控制流程实例执行过程的参与者。

（10）工作项。代表流程实例中活动的参与者将要执行的工作。

要分析现实中的处理流程，必须首先描述目标系统的流程，这个过程也可以称为建模。流程是个复杂的事务，必须从多方面才可以描述一个流程，包括:“谁”，流程的参与者；“什

么”，参与者做什么工作；“何时”，工作完成的时间限制，还需要说明工作的数据流和完成工作的控制流。人们认为自然语言在描述如此复杂的事务时容易引起歧义，所以人们定义了一些形式化语言试图在自然语言中挑选一个子集，这个子集既可以真正描述流程，又能够摆脱自然语言的复杂和多变，实际上想在人和机器在理解处理流程上架起一座桥梁，如同其他计算机语言及后来发展的统一建模语言一样，这些形式化的语言也称为“工作流定义语言”。

同样，为了描述实际中的处理流程，人们也想到了图形的方式。有限状态自动机是一种分析状态和改变状态的良好工具，这种方法需要完全列出流程中所有状态及这些状态的组合，当处理流程变得庞大时，自动机所对应的状态图膨胀得太厉害。由于 Petri 网有严格的数学基础和图形化的规范语义，在描述离散事件动态系统方面的能力已经得到公认，具有较强的模型分析能力，在工作流的描述和分析中已经是人们广泛采用的一种方法。虽然有人认为图形并非工作流的最佳表示方法，但由于图形的直观性，大多数设计者都愿意采用图形的描述方式。有关有限状态自动机和 Petri 网的论述请参考其他书籍。

8.6.2 工作流管理系统

根据工作流管理联盟（Workflow Management Coalition，WFMC）的定义，工作流管理系统是一种“在工作流形式化表示的驱动下，通过软件的执行而完成工作流定义、管理及执行的系统”，其主要目标是对业务过程中各活动发生的先后次序及与活动相关的相应人力或信息资源的调用进行管理，而实现业务过程的自动化。

如同关系数据库一样，现在已经出现了专门的工作流管理系统，这些系统经过专门的设计，从不同的角度负责解决设计者在流程设计中遇到的共同问题：节点定义、路径选择、数据流动等。不幸的是，和关系数据库共同基于关系代数、支持标准的 SQL 不同，这些工作流管理系统基于不同的数学模型，提供完全不同的接口。所以这些工作流管理系统各具特色，但在通用性和其他系统相互协作上的不足使得这些系统的应用受到了限制。

WFMC 给出了包含六个基本模块的参考模型，这六个模块被认为是工作流管理系统的最基本组成，这个参考模型同时包括了这些模块之间的接口标准。

（1）流程定义工具。这部分软件提供图形化或者其他方式的界面给设计者。由设计者将实际工作流程进行抽象，并将设计者提交的流程定义转换为形式化语言描述，提供给计算机工作流执行服务进行流程实例处理的依据。

（2）工作流执行服务。这个服务程序是工作流管理系统的核心，它使用一种或者多种

数据流引擎，对流程定义进行解释，激活有效的流程实例，推动流程实例在不同的活动中运转。和客户应用程序、其他工作流服务执行程序及其他应用程序进行交互，从而完成流程实例的创建、执行和管理工作。同时这部分软件为每个用户维护一个活动列表，告诉用户当前必须处理的任务。如果有必要，还可以通过电子邮件甚至是短消息的形式提醒用户任务的到达。

（3）其他工作流执行服务。大型的企业工作流应用，往往包括多个工作流管理系统。这就需要这些工作流管理系统之间进行有效的交互，避免画地为牢、信息孤岛的现象出现。

（4）客户应用程序。这是给最终用户的界面，用户通过使用这部分软件对工作流的数据进行必要的处理，如果用户是当前活动的拥有者，还可通过客户应用程序改变流程实例的活动，将流程实例推动到另外一个活动中。

（5）被调用应用程序。这常常是对工作流所携带数据的处理程序，用得很多的是电子文档的处理程序。它们由工作流执行服务在流程实例的执行过程中调用，向最终用户展示待处理数据。在流程定义中应该定义这些应用程序的信息，包括名称、调用方式、参数等。

（6）管理和监控工具。如同数据库管理系统或多或少地提供一些方式告诉管理员当前数据库的使用状态一样，工作流管理系统也应该提供对流程实例的状态查询、挂起、恢复、销毁等操作，同时提供系统参数、系统运行情况统计等数据。

看到这里，没有处理流程设计经验的设计者一定已经云里雾里了。确实，流程设计是系统设计中最困难的一部分，它的复杂性直接来源于现实世界的复杂性。而且直到现在，人们对流程的设计，仍然是在探索之中。

8.7 简单分布式计算机应用系统的设计

网络极大地扩展了计算机的应用范围，同时，由于升级到更强的服务器的费用常常远远高于购买多台档次稍低的机器，更何况虽然计算机有了长足的发展，可是单台计算机的功能仍然十分有限，利用联网的计算机协同工作，共同完成复杂的工作成为相对成本较低的选择，而且可以完成单台计算机所无法完成的任务。分布式系统使得这一目标成为可能。另外，网络本质上并不可靠，特别是远程通信，分布式系统还带来了并发和同步的问题。

分布式系统可以由两种完全不同的方式来进行协同和合作，第一种是基于实例的协作。这种方式所有的实例都处理自己范围内的数据，这些对象实例的地位是相同的，当一个对象实例必须要处理不属于它自己范围的数据时，它必须和数据归宿的对象实例通信，请求另外

一个对象实例进行处理。请求对象实例可以启动对象、调用远程对象的方法，以及停止运行远程实例。基于实例的协作具有良好的灵活性，但由于实例之间的紧密联系复杂的交互模型，使得开发成本提高，而且，由于实例必须能够通过网络找到，所以通信协议必须包括实例的生存周期管理，这使得基于实例的协作大多只限于统一的网络，对于复杂的跨平台的系统就难以应付。所以基于实例的协作适用于比较小范围内网络情况良好的环境中，这种环境常常被称为近连接。这种情况下对对象的生存周期管理所带来不寻常的网络流量是可以容忍的。

使用基于实例的协作常常使用被称为“代理”的方法，某个对象实例需要调用远程对象时，它可以只和代理打交道，由代理完成和远程对象实例的通信工作：创建远程对象，提交请求、得到结果，然后把结果提交给调用的对象实例。这样，这个对象实例甚至可以不知道自己使用了远程对象。当远程对象被替换掉（升级）时，对本地代码也没有什么需要修改的地方。

另外一种方式是基于服务的协作，该方法试图解决基于实例的协作的困难。它只提供远程对象的接口，用户可以调用这些方法，却无法远程创建和销毁远程对象实例。这样减少了交互，简化了编程，而且使得跨平台协作成为可能。同样由于只提供接口，这种协作方式使得对象间的会话状态难以确定，而且通信的数据类型也将有所限制，基本上很难使用自定义的类型。基于服务的协作适用于跨平台的网络，网络响应较慢的情况，这种环境常称为远连接，这时，简化交互性更为重要，而频繁的网络交换数据会带来难以容忍的延时。

基于服务的协作往往采用分层次的结构，高层次的应用依赖于低层次的对象，而低层次对象实例的实现细节则没有必要暴露给高层次对象，这种安排使得高层次的实现不受低层次如何实现的影响，同时，当低层次服务修改时，高层次的服务也不应该受到影响。

设计者在进行设计时，通常会倾向于比较细致的设计，对象往往提供了大量的操作和方法，响应许多不同的消息，以增加达到系统的灵活性、可维护性等。这在单个系统中没有什么问题，当考虑分布式系统的设计时，这种细致的设计所带来对对象方法的大量调用会比较严重地影响性能，所以在分布式系统中，倾向于使用大粒度的设计方式，往往在一个方法中包含了许多参数，每个方法基本上代表了一个独立的功能。当然这样的设计使得参数的传递变得复杂，当需要修改参数时，需要对比较大范围的一段过程代码进行修改，而不是像小粒度设计一样，只需要修改少量的代码。

8.8 系统运行环境的集成与设计

在设计一个新的系统时，设计者必须考虑目标系统的运行环境问题，人们往往认为软件应该能够在任何环境中运行，常常看到这样的系统，硬件已经升级了多次，而软件还是原来的软件。软件的运行环境是指系统运行的设备、操作系统和网络配置。

本节给出软件运行的几个典型环境，设计者可以从这几种典型环境中选择适合自己的目标系统的环境，也可以将这些典型环境做一些组合，来满足自己设计的系统的特殊要求。

1．集中式系统

早期的计算机系统没有什么可以选择的，除了集中式系统。所有的操作都集中于一台主机中，而操作员必须在主机的附近操作，结果也在附近给出。这种系统仍然广泛地应用于批处理应用系统及更大的分布式系统的一部分。集中式系统常见于银行、保险、证券行业，它们含有大规模的处理应用。而现在流行的电子商务又给大型处理机注入了新的活力，人们发现电子商务要面对大量的事务，需要大型处理机来处理。但是，实践中很少单独使用集中式系统，因为大量的系统需要处理在地理上分布得很远的连接请求，这些请求有的需要实时响应，并可能要发送到其他某个地方的一个集中式系统。所以，在现代的系统中，集中式系统通常是某个分布式系统的一个环节。

集中式系统由以下几个部分组成。

（1）单计算机结构：这种结构简单、容易维护，但是处理能力受到限制。

（2）集群结构：由多个计算机组成，这些计算机具有类似的硬件平台、操作系统等。通常采用负载均衡、资源共享等方式实现更大的处理能力和容量。

（3）多计算机结构：由多个计算机组成，这些计算机之间操作环境可能不同。适用于当系统可以分解成多个不同的子系统时。

2．分布式系统

在 7.9 节中，已经简单介绍了分布式系统。分布式系统由于网络的普遍延伸，费用的不断降低而越来越成为大型系统的首选环境。分布式系统必须基于网络，这个网络可以是在一个地域内的局域网，也可以是跨越不同城市乃至国家的广域网。对比集中式的计算机环境，分布式系统有着多种多样的形式。这也给设计者在确定系统运行环境时带来一定的烦恼。

3．C/S 结构

系统由提供服务的服务器和发起请求、接受结果的客户机构成。这种结构是一种可以使用很多方式实现的通用结构模型。并非只限于数据库的 C/S 结构，典型的还有网络打印服

务系统，现在流行的网络游戏也显然是基于这种结构的。

4．多层结构

这种结构是 C/S 结构的扩展，典型的分为由存储数据的数据库服务器作为数据层、实现商业规则的程序作为逻辑层、管理用户输入输出的视图层所组成的三层结构。当系统更复杂时，可以再增加其他层次构成多层结构。

多层结构形式复杂，功能多样。实现多层结构常常需要来实现不同层次间通信的专门程序——管件，也称为中间件。中间件大多数实现远程程序调用、对象请求调度等功能。

现代企业级的计算机系统大量地基于分布式结构。支持分布式系统的软件也曾经如同雨后春笋。系统如何分层、如何处理分布带来的同步等问题也就同样在考验设计者。

5.Internet、Intranet 和 Extranet

Internet 是全球的网络集合，使用通用的 TCP/IP 协议来相互连接。Internet 提供电子邮件、文件传输、远程登录等服务。Intranet 是私有网络，只限于内部使用，也使用 TCP/IP 协议。Extranet 是一个扩展的 Intranet。它包括企业之外的和企业密切相关合作的其他企业。Extranet 允许分离的组织交换信息并进行合作，这样就形成了一个虚拟组织。现在的 VPN 技术允许在公用网络上架构只对组织内部开发服务。

Web 同样基于 C/S 结构，实际上 Web 接口是一个通用的接口，不是只能使用浏览器的协议，它同样能够在普通的程序中使用。Internet 和 Web 已经给设计者提供了一个非常富有吸引力的选择方案。它的优势在于：它们已经成为网络的事实上的标准，支持它们的软件已经广泛地存在于全世界的计算机中，而且通信费用已经下降到很有竞争力的水平。从某种程度上来说，企业可以把 Internet 当作自己廉价的广域网。没有它们，电子商务还是水中月。

当然，事物有相反的一面，当设计者试图采用 Internet 时，必须考虑其不利的一面。Internet 的安全性过去是，现在是，以后仍然是设计者头痛的问题。其他诸如可靠性、系统吞吐量、不断发展的技术和标准都是影响系统选择它们作为运行环境的不利因素。 设计者应该根据目标系统的实际需要来选择不同的运行环境。不过，已经有越来越多的公司提供支持 Internet 和电子商务的接口。

8.9 系统过渡计划

当新系统似乎开发完毕，要取代原来的系统时，系统过渡就是设计者不得不面对的问题。

这个问题，不幸的是，比许多人想象得要复杂，和软件开发一样，存在着许多冲突和限制。例如，费用、客户关系、后勤保证和风险等。设计者需要考虑的问题也很多，其中比较重要的几个问题是：

如果同时运行两个系统，会给客户造成多大的开销；

如果直接运行新系统，客户面对的风险有多大；

对新系统试运行时的查错和纠错，以及出现严重错误而导致停止运行时的应急措施；

客户运行新系统将面临的不利因素有哪些；

人员的培训。

使用不同的系统过渡方案意味着不同的风险，不同的费用及不同的复杂度。

1．直接过渡

这是一种快速的系统过渡方式，当新系统运行时，立即关闭原来的系统。这种过渡方式非常简单，没有后勤保障的问题，也不要消耗很多资源。同时，它也意味着大风险，目标系统的特性决定了风险的大小。设计者主要要权衡当新系统失败时，系统停止运行或者勉强运行给客户带来的损失有多大。由于这种过渡方式简单而费用低廉，对于可以容忍停机一段时间的系统的实践者，可以采用这种方式。

2．并行过渡

设计者采用并行过渡方式，让新系统和旧系统在一段时间里同时运行，通过这样的旧系统作为新系统的备份，可以大大降低系统过渡的风险。可是并行过渡显然比直接过渡要消耗更多的资源：现有的硬件资源必须保证能同时跑两套系统，这常常意味着增加服务器和额外的存储空间，需要增加人员来同时使用两套系统，或者增加现有员工的工作量，让他们同时操作两套系统。这种方式同时也增加了管理和后勤保障的复杂度。据统计，并行过渡时期的开销是旧系统单独运行时的 2.5～3 倍。

设计者还会发现有些系统无法使用并行过渡的方式，主要是客户没有足够的资源来维持两个系统同时运行，另外一种情况是新、旧系统差别太大，旧系统的数据无法为新系统采用。当客户无法使用并行过渡，又想尽可能地减少风险，设计者可以使用部分并行过渡的策略，使并行的开销减少到客户能够接受的范围内。

3．阶段过渡

通常在系统非常复杂、过于庞大以至于无法一次性进行过渡时采用，也适用于分阶段开发的系统。设计者需要设计一系列步骤和过程来完成整个系统的过渡，这种过渡方式和系统的复杂程度相关，随着系统的不同往往有很大的不同。和并行过渡一样，阶段过渡也能够减

少风险，显然局部的失败要比全体的失败更容易接受，带来的损失更小。阶段过渡也带来了复杂性，有时候比并行过渡更加复杂。

第 9 章：软件架构设计

像学写文章一样，在学会字、词、句之后，就应上升到段落，就应追求文章的“布局谋篇”，这就是架构。通俗地讲，软件架构设计就是软件系统的“布局谋篇”。

人们在软件工程实践中，逐步认识到了软件架构的重要性，从而开辟了一个崭新的研究领域。软件架构的研究内容主要涉及软件架构描述、软件架构设计、软件架构风格、软件架构评价和软件架构的形成方法等。

软件设计人员学习软件架构知识旨在站在较高的层面上整体地解决好软件的设计、复用、质量和维护等方面的实际问题。

9.1 软件架构概述

软件架构是软件抽象发展到一定阶段的产物，从编程的角度，可以清晰地看到软件抽象层次和表达工具的发展历史。

20 世纪 60 年代是子程序的年代：出现了原始的软件架构，即子程序，并以程序间的调用为连接关系。

20 世纪 70 年代是模块化的年代：出现了数据流分析、实体—关系图（E-R 图）、信息隐藏等工具和方法，软件的抽象层次发展到了模块级。

20 世纪 80 年代是面向对象的年代：基于模块化的编程语言进一步发展成面向对象的语言，继承性地增加了一种新元素之间的连接关系。

20 世纪 90 年代是框架的年代：标准的基于对象的架构以框架的形式出现了。如电子数据表、文档、图形图像、音频剪辑等可互换的黑箱对象，可以相互嵌入。

当前（最近 10 年来）：中间件和 IT 架构作为标准平台出现，用可购买可复用的元素来构建系统，同时，基于架构的开发方法和理论不断成熟。

9.1.1 软件架构的定义

软件架构仍在不断发展中，还没有形成一个统一的、公认的定义，这里仅举出几个较权

威的定义。

定义 1：软件或计算机系统的软件架构是该系统的一个（或多个）结构，而结构由软件元素、元素的外部可见属性及它们之间的关系组成。

定义 2：软件架构为软件系统提供了一个结构、行为和属性的高级抽象，由构成系统的元素的描述、这些元素的相互作用、指导元素集成的模式及这些模式的约束组成。

定义 3：软件架构是指一个系统的基础组织，它具体体现在：系统的构件，构件之间、构件与环境之间的关系，以及指导其设计和演化的原则上。（IEEE1471- 2000）

前两个定义都是按“元素—结构—架构”这一抽象层次来描述的，它们的基本意义相同，其中定义 1 较通俗，因此，本章采用这一定义。该定义中的“软件元素”是指比“构件”更一般的抽象，元素的“外部可见属性”是指其他元素对该元素所做的假设，如它所提供的服务、性能特征等。

为了更好地理解软件架构的定义，特作如下说明：

（1）架构是对系统的抽象，它通过描述元素、元素的外部可见属性及元素之间的关系来反映这种抽象。因此，仅与内部具体实现有关的细节是不属于架构的，即定义强调元素的“外部可见”属性。

（2）架构由多个结构组成，结构是从功能角度来描述元素之间的关系的，具体的结构传达了架构某方面的信息，但是个别结构一般不能代表大型软件架构。

（3）任何软件都存在架构，但不一定有对该架构的具体表述文档。即架构可以独立于架构的描述而存在。如文档已过时，则该文档不能反映架构。

（4）元素及其行为的集合构成架构的内容。体现系统由哪些元素组成，这些元素各有哪些功能（外部可见），以及这些元素间如何连接与互动。即在两个方面进行抽象：在静态方面，关注系统的大粒度（宏观）总体结构（如分层）；在动态方面，关注系统内关键行为的共同特征。

（5）架构具有“基础”性：它通常涉及解决各类关键的重复问题的通用方案（复用性），以及系统设计中影响深远（架构敏感）的各项重要决策（一旦贯彻，更改的代价昂贵）。

（6）架构隐含有“决策”，即架构是由架构设计师根据关键的功能和非功能性需求（质量属性及项目相关的约束）进行设计与决策的结果。不同的架构设计师设计出来的架构是不一样的，为避免架构设计师考虑不周，重大决策应经过评审。特别是架构设计师自身的水平是一种约束，不断学习和积累经验才是摆脱这种约束走向自由王国的必经之路。

在设计软件架构时也必须考虑硬件特性和网络特性，因此，软件架构与系统架构二者间

的区别其实不大。但是，在大多情况下，架构设计师在软件方面的选择性较之硬件方面，其自由度大得多。因此，使用“软件架构”这一术语，也表明了一个观点：架构设计师通常将架构的重点放在软件部分。

将软件架构置于商业背景中进行观察，可以发现软件架构对企业非常重要。

（1）影响架构的因素。软件系统的项目干系人（客户、用户、项目经理、程序员、测试人员、市场人员等）对软件系统有不同的要求开发组织（项目组）有不同的人员知识结构、架构设计师的素质与经验、当前的技术环境等方面都是影响架构的因素。

这些因素通过功能性需求、非功能性需求、约束条件及相互冲突的要求，影响架构设计师的决策，从而影响架构。

（2）架构对上述诸因素具有反作用，例如，影响开发组织的结构。架构描述了系统的大粒度（宏观）总体结构，因此可以按架构进行分工，将项目组为几个工作组，从而使开发有序；影响开发组织的目标，即成功的架构为开发组织提供了新的商机，这归功于：系统的示范性、架构的可复用性及团队开发经验的提升，同时，成功的系统将影响客户对下一个系统的要求等。这种反馈机制构成了架构的商业周期。

9.1.2 软件架构的重要性

从技术角度看，软件架构的重要性表现为如下几方面。

（1）项目关系人之间交流的平台。软件系统的项目关系人分别关注系统的不同特性，而这些特性都由架构所决定，因此，架构提供了一个共同语言（公共的参考点），项目关系人以此作为彼此理解、协商、达成共识或相互沟通的基础。架构分析既依赖于又促进了这个层次上的交流。

（2）早期设计决策。从软件生命周期来看，软件架构是所开发系统的最早设计决策的体现，主要表现为：

架构明确了对系统实现的约束条件：架构是架构设计师对系统实现的各方面进行权衡的结果，是总体设计的体现，因此，在具体实现时必须按架构的设计进行。

架构影响着系统的质量属性：要保证系统的高质量，具有完美的架构是必要的（虽然不充分）。架构可以用来预测系统的质量，例如，可以根据经验对该架构的质量（如性能）作定性的判断。

架构为维护的决策提供根据。在架构层次上能为日后的更改决策提供推理、判断的依据。一

个富有生命力的架构，应该是在最有可能更改的地方有所考虑（架构的柔性），使其在此点最容易进行更改。

架构有助于原型开发。可以按架构构造一个骨架系统（原型），例如，在早期实现一个可执行的特例，确定潜在的性能问题。

借助于架构进行成本与进度的估计。

（3）在较高层面上实现软件复用。软件架构作为系统的抽象模型，可以在多个系统间传递（复用），特别是比较容易地应用到具有相似质量属性和功能需求的系统中。产品线通常共享一个架构。产品线的架构是开发组织的核心资产之一，利用架构及其范例进行多系统的开发，在开发时间、成本、生产率和产品质量方面具有极大的回报。基于架构的开发强调对各元素的组合或装配。系统开发还可以使用其他组织开发的元素，例如购买商业构件。

（4）架构对开发的指导与规范意义不容忽略。架构作为系统的总体设计，它指导后续的详细设计和编码。架构使基于模板的开发成为可能，有利于开发的规范化和一致性，减少开发与维护成本。架构可以作为培训的基础，有利于培养开发团队和培训相关人员。

从软件开发过程来看，如果采用传统的软件开发模型（生命周期模型），则软件架构的建立应位于概要设计之前，需求分析之后。

基于架构的软件开发模型则明确地把整个软件过程划分为架构需求、设计、文档化、评审（评估）、实现、演化等 6 个子过程。本章各节将分别对这些子过程进行讨论。

9.1.3 架构的模型

软件架构作为一个有机的整体，可以分解成多个侧面来认识，每个侧面强调它的不同方面的特征，从而使架构设计师能整体地把握它的重点。我们可以将软件架构归纳成 5 种模型：结构模型、框架模型、动态模型、过程模型和功能模型。最常用的是结构模型和动态模型。

（1）结构模型。这是一个最直观、最普遍的建模方法。这种方法以架构的构件、连接件和其他概念来刻画结构，并力图通过结构来反映系统的重要语义内容，包括系统的配置、约束、隐含的假设条件、风格、性质。研究结构模型的核心是架构描述语言。

（2）框架模型。框架模型与结构模型类似，但它不太侧重描述结构的细节而更侧重于整体的结构。框架模型主要以一些特殊的问题为目标建立只针对和适应该问题的结构。

（3）动态模型。动态模型是对结构或框架模型的补充，研究系统“大颗粒”的行为性

质。例如，描述系统的重新配置或演化。动态可能指系统总体结构的配置、建立或拆除通信通道或计算的过程。

（4）过程模型。过程模型研究构造系统的步骤和过程。因而结构是遵循某些过程脚本的结果。

（5）功能模型。该模型认为架构由一组功能构件按层次组成，且下层向上层提供服务。它可以看作是一种特殊的框架模型。

这 5 种模型各有所长，也许将 5 种模型有机地统一在一起，形成一个完整的模型来刻画软件架构更合适。即将软件架构视为这些模型的统一体，通过这些模型的表述（文档）来完整反映软件架构。例如，Kruchten 在 1995 年提出了一个“4+1”的视图模型。“4+1” 视图模型从 5 个不同的视角包括逻辑视图、进程视图、物理视图、开发视图和场景视图来描述软件架构。每一个视图只关心系统的一个侧面，5 个视图结合在一起才能反映系统的软件架构的全部内容。“4+1”视图模型如图 9-1 所示。

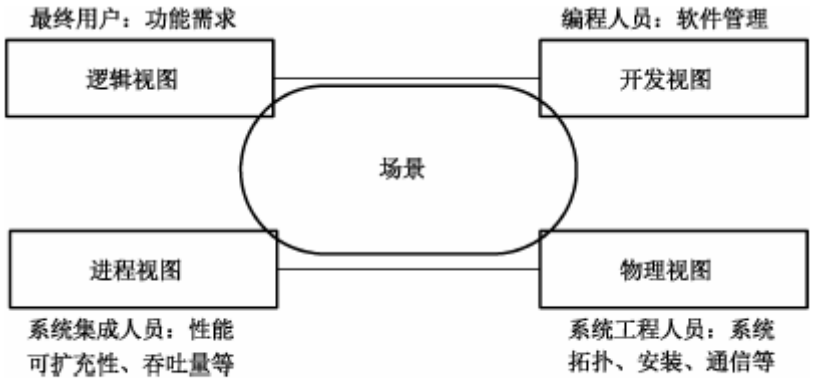


图 9-1 “4+1”视图模型

（1）逻辑视图：主要支持系统的功能需求，即系统提供给最终用户的服务。在逻辑视图中，系统分解成一系列的功能抽象，这些抽象主要来自问题领域。这种分解不但可以用来进行功能分析，而且可用作标识在整个系统的各个不同部分的通用机制和设计元素。在面向对象技术中，通过抽象、封装和继承，可以用对象模型来代表逻辑视图，用类图来描述逻辑视图。逻辑视图中使用的风格为面向对象的风格，逻辑视图设计中要注意的主要问题是要保持一个单一的、内聚的对象模型贯穿整个系统。

（2）开发视图：也称为模块视图，主要侧重于软件模块的组织和管理。软件可通过程序库或子系统进行组织，这样，对于一个软件系统，就可以由不同的人进行开发。开发视图要考虑软件内部的需求，如软件开发的容易性、软件的重用和软件的通用性，要充分考虑由于具体开发工具的不同而带来的局限性。开发视图通过系统输入输出关系的模型图和子系统

图来描述。可以在确定了软件包含的所有元素之后描述完整的开发角度，也可以在确定每个元素之前，列出开发视图原则。

（3）进程视图：侧重于系统的运行特性，主要关注一些非功能性的需求，例如系统的性能和可用性。进程视图强调并发性、分布性、系统集成性和容错能力，以及逻辑视图中的主要抽象的进程结构。它也定义逻辑视图中的各个类的操作具体是在哪一个线程中被执行的。进程视图可以描述成多层抽象，每个级别分别关注不同的方面。

（4）物理视图：主要考虑如何把软件映射到硬件上，它通常要考虑到解决系统拓扑结构、系统安装、通信等问题。当软件运行于不同的节点上时，各视图中的构件都直接或间接地对应于系统的不同节点上。因此，从软件到节点的映射要有较高的灵活性，当环境改变时，对系统其他视图的影响最小。

（5）场景：可以看作是那些重要系统活动的抽象，它使四个视图有机地联系起来，从某种意义上说，场景是最重要的需求抽象。在开发架构时，它可以帮助设计者找到架构的构件和它们之间的作用关系。同时，也可以用场景来分析一个特定的视图，或描述不同视图构件间是如何相互作用的。场景可以用文本表示，也可以用图形表示。

希赛教育专家提示：逻辑视图和开发视图描述系统的静态结构，而进程视图和物理视图描述系统的动态结构。对于不同的软件系统来说，侧重的角度也有所不同。例如，对于管理信息系统来说，比较侧重于从逻辑视图和开发视图来描述系统，而对于实时控制系统来说，则比较注重于从进程视图和物理视图来描述系统。

9.2 架构需求与软件质量属性

架构的基本需求主要是在满足功能属性的前提下，关注软件质量属性，架构设计则是为满足架构需求（质量属性）寻找适当的“战术”。

软件属性包括功能属性和质量属性，但是，软件架构（及软件架构设计师）重点关注的是质量属性。因为，在大量的可能结构中，可以使用不同的结构来实现同样的功能性，即功能性在很大程度上是独立于结构的，架构设计师面临着决策（对结构的选择），而功能性所关心的是它如何与其他质量属性进行交互，以及它如何限制其他质量属性。

9.2.1 软件质量属性

《GB/T16260-1996(idt ISO／IEC9126：1991)信息技术软件产品评价质量特性及其使用指

南》中描述的软件质量特性包括功能性、可靠性、易用性、效率、可维护性、可移植性等 6 个方面，每个方面都包含若干个子特性。

功能性：适合性、准确性、互操作性、依从性、安全性；

可靠性：成熟性、容错性、易恢复性；

易用性：易理解性、易学性、易操作性；

效率：时间特性、资源特性；

可维护性：易分析性、易改变性、稳定性、易测试性；

可移植性：适应性、易安装性、遵循性、易替换性；

正如上述列举与分类，软件的质量属性很多，也有各种不同的分类法和不同的表述。虽然术语没有统一的定义，但其含义可以认为业界已有共识。下面选取常用的质量属性术语，并做逐一说明。

1．运行期质量属性

性能：性能是指软件系统及时提供相应服务的能力。包括速度、吞吐量和持续高速性三方面的要求。

安全性：指软件系统同时兼顾向合法用户提供服务，以及阻止非授权使用的能力。

易用性：指软件系统易于被使用的程度。

可伸缩性：指当用户数和数据量增加时，软件系统维持高服务质量的能力。例如，通过增加服务器来提高能力。

互操作性：指本软件系统与其他系统交换数据和相互调用服务的难易程度。

可靠性：软件系统在一定的时间内无故障运行的能力。

持续可用性：指系统长时间无故障运行的能力。与可靠性相关联，常将其纳入可靠性中。

鲁棒性：是指软件系统在一些非正常情况（如用户进行了非法操作、相关的软硬件系统发生了故障等）下仍能够正常运行的能力。也称健壮性或容错性。

2.开发期质量属性

易理解性：指设计被开发人员理解的难易程度。

可扩展性：软件因适应新需求或需求变化而增加新功能的能力。也称为灵活性。

可重用性：指重用软件系统或某一部分的难易程度。

可测试性：对软件测试以证明其满足需求规范的难易程度。

可维护性：当需要修改缺陷、增加功能、提高质量属性时，定位修改点并实施修改的难易程度；

可移植性：将软件系统从一个运行环境转移到另一个不同的运行环境的难易程度。

在实践中，架构设计师追求质量属性常常陷入“鱼和熊掌”的两难境地，这就需要架构设计师的决策智慧了。表 9-1 反映了质量属性之间的相互制约关系（正相关或负相关），其中“+”代表“行属性”能促进“列属性”；而“-”则相反。例如，第一列符号说明许多属性（行）对性能（列）有副作用，第一行符号说明性能（行）对许多属性（列）有副作用，认识这一点，对于架构决策的权衡很重要。

表 9-1　质量属性关系矩阵

	性能	安全性	持续可用性	可互操作性	可靠性	鲁棒性	易用性	可测试性	可重用性	可维护性	可扩展性	可移植性
性能				-	-	-	-	-		-	-	-
安全性	-			-			-	-	-			
持续可用性					+	+						
可互操作性	-	-									+	+
可靠性	-		+			+	+	+		+	+	
鲁棒性	-		+		+		+					
易用性	-					+		-				
可测试性	-		+		+		+			+	+	
可重用性	-	-		+	-			+		+	+	+
可维护性	-		+		+			+			+	
可扩展性	-	-			+			+		+		+
可移植性	-			+			-	+	+	-	+	

9.2.2 6 个质量属性及实现

本节从架构关注点来研究质量属性实现，将质量属性分为 6 种：可用性、可修改性、性能、安全性、可测试性、易用性。其他的质量属性一般可纳入这几个属性中（在其他文献中为了强调常单列出来），例如，可扩充性可归入可修改性中（修改系统容量），可移植性也可以作为平台的可修改性来获得。对于未能纳入的其他质量属性，可以用本章的方法进行研究。

那么，如何描述质量属性需求呢？采用质量属性场景作为一种描述规范，它由以下 6 个部分组成，如图 9-2 所示。

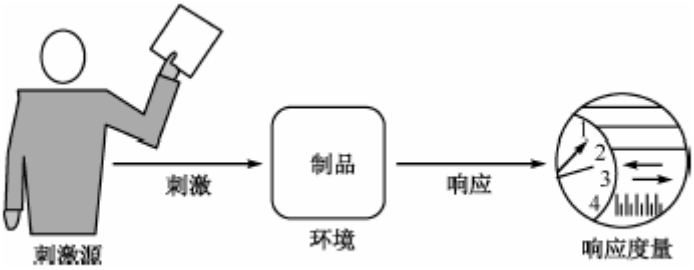


图 9-2　质量属性场景

刺激源：生成该刺激的实体（人、计算机系统或其他激励器）；

刺激：刺激到达系统时可能产生的影响（即需要考虑和关注的情况）；

环境：该刺激在某条件内发生。如系统可能正处于过载情况；

制品：系统中受刺激的部分（某个制品被刺激）；

响应：刺激到达后所采取的行动；

响应度量：当响应发生时，应能够以某种方式对应其度量，用于对是否满足需求的测试。

需要将一般的质量属性场景（一般场景）与具体的质量属性场景（具体场景）区别开来，前者是指独立于具体系统、适合于任何系统的一般性场景；而后者是指适合于正在考虑的某个特定系统的场景，具体场景通常是指从一般场景中抽取特定的、面向具体系统的内容。下面几个小节中为每个质量属性提供一张表，该表中给出了质量属性场景每部分的一些可能取值，整体上形成一个一般场景的表格描述。在实际应用时，根据系统的具体情况，从该表中选取适当的值，就能变成具体场景（可读性强、可应用），可以把具体场景的集合作为系统的质量属性需求。

实现这些质量属性的基本设计决策，称为“战术”，而把战术的集合称为“架构策略”。这些架构策略供架构设计师选择。下面几个小节将对各质量属性的战术进行示例性的总结。

“战术”作为逻辑部件位于图 9-2 的制品中，它旨在控制对刺激的响应。

1．可用性及其实现战术

（1）可用性的描述。可用性的描述如表 9-2 所示。

表 9-2　可用性一般场景的表格描述

场景的部分	可能的值
刺激源	系统内部、系统外部
刺激	错误：疏忽（构件对某输入未做出反映）、崩溃、时间不当（响应时间太早或太迟）、响应不当（以一个不正确的值来响应）
制品	系统的处理器、通信通道、存储器、进程
环境	正常操作、降级模式
响应	系统应检测事件，并进行如下一个或多个活动： • 将其记录下来 • 通知适当的各方，包括用户和其他系统 • 根据规则屏蔽导致错误或故障的事件源 • 不可用（进入修理状态） • 继续在正常或降级模式下运行
响应度量	可用时间、修复时间、各种情况的时间间隔

可用性一般场景可以用图 9-3 表示。

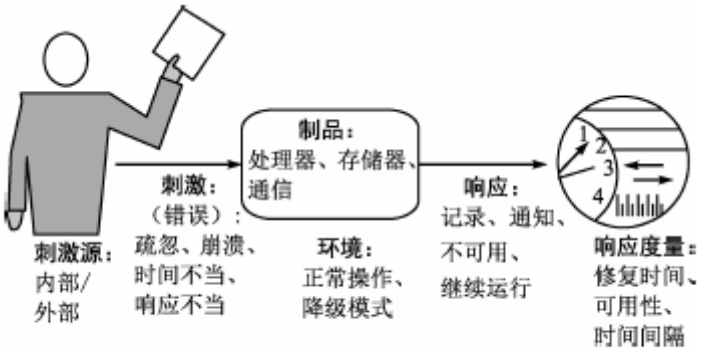


图 9-3　可用性一般场景

对一般场景进行具体化可以得到可用性具体场景，如图 9-4 所示。

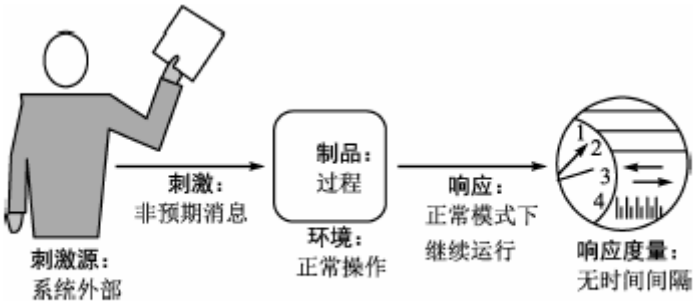


图 9-4　可用性的一个具体场景（示例）

（2）可用性战术。可用性战术的目标是阻止错误发展成故障，至少能够把错误的影响限制在一定范围内，从而使修复成为可能。战术分为：错误检测、错误恢复、错误预防。

① 错误检测

命令/响应：一个构件发出一个命令，并希望在预定义的时间内收到一个来自审查构件的响应，例如远程错误的检测。

心跳（计时器）：一个构件定期发出一个心跳消息，另一个构件收听到消息，如果未收到心跳消息，则假定构件失败，并通知错误纠正构件。

异常：当出现异常时，异常处理程序开发执行。

② 错误恢复

表决：通过冗余构件（或处理器）与表决器连接，构件按相同的输入及算法计算输出值交给表决器，由表决器按表决算法（如多数规则）确定是否有构件出错，表决通常用在控制系统中。

主动冗余（热重启、热备份）：所有的冗余构件都以并行的方式对事件做出响应。它们都处在相同的状态，但仅使用一个构件的响应，丢弃其余构件的响应。错误发生时通过切换的方式使用另一个构件的响应。

被动冗余（暖重启/双冗余/三冗余）：一个构件（主构件）对事件做出响应，并通知其他构件（备用的）必须进行的状态更新（同步）。当错误发生时，备用构件从最新同步点接替主构件的工作。

备件：备件是计算平台配置用于更换各种不同的故障构件。

状态再同步：主动和被动冗余战术要求所恢复的构件在重新提供服务前更新其状态。更新方法取决于可以承受的停机时间、更新的规模及更新的内容多少。

检查点/回滚：检查点就是使状态一致的同步点，它或者是定期进行，或者是对具体事件做出响应。当在两检查点之间发生故障时，则以这个一致状态的检查点（有快照）和之后发生的事务日志来恢复系统（数据库中常使用）。

③ 错误预防

从服务中删除：如删除进程再重新启动，以防止内存泄露导致故障的发生。

事务：使用事务来保证数据的一致性，即几个相关密切的步骤，要么全成功，要么都不成功。

进程监视器：通过监视进程来处理进程的错误。

2．可修改性及其实现战术

（1）可修改性的描述。可修改性的描述如表 9-3 所示。

表 9-3　可修改性一般场景的表格描述

场景的部分	可能的值
刺激源	最终用户、开发人员、系统管理员
刺激	增加/删除/修改/改变：功能、质量属性、容量
制品	用户界面、平台、环境或关联系统
环境	运行时、编译时、构建时、设计时
响应	查找要修改的位置，进行修改（不影响其他功能），进行测试，部署所修改的内容
响应度量	对修改的成本进行度量，对修改的影响进行度量

对于可修改性一般场景的图示及可修改性具体场景，读者可仿照前面可用性的描述方式，自行练习。

（2）可修改性战术。包括局部化修改、防止连锁反应、推迟绑定时间。

① 局部化修改。在设计期间为模块分配责任，以便把预期的变更限制在一定的范围内，从而降低修改的成本。

维持语义的一致性：语义的一致性指的是模块中责任之间的关系，使这些责任能够协同工作，不需要过多地依赖其他模块。耦合和内聚指标反映一致性，应该根据一组预期的变更来度量语义一致性。使用“抽象通用服务”（如应用框架的使用和其他中间软件的使用）来支持可修改性是其子战术。

预期期望的变更：通过对变更的预估，进行预设、准备，从而使变更的影响最小。

泛化该模块：使一个模块更通用、更广泛的功能。

限制可能的选择：如在更换某一模块（如处理器）时，限制为相同家族的成员。

② 防止连锁反应。由于模块之间有各种依赖性，因此，修改会产生连锁反应。防止连锁反应的战术如下。

信息隐藏：就是把某个实体的责任分解为更小的部分，并选择哪些信息成为公有的，哪些成为私有的，通过接口获得公有责任。

维持现有的接口：尽可能维持现有接口的稳定性。例如通过添加接口（通过新的接口提供新的服务）可以达到这一目的。

限制通信路径：限制与一个给定的模块共享数据的模块。这样可以减少由于数据产生/使用引入的连锁反应。

仲裁者的使用：在具有依赖关系的两个模块之间插入一个仲裁者，以管理与该依赖相关的活动。仲裁者有很多种类型，例如：桥、调停者、代理等就是可以提供把服务的语法从一种形式转换为另一种形式的仲裁者。

③ 推迟绑定时间。系统具备在运行时进行绑定并允许非开发人员进行修改（配置）。

运行时注册：支持即插即用。

配置文件：在启动时设置参数。

多态：在方法调用的后期绑定。

构件更换：允许载入时绑定。

3．性能及其实现战术

（1）性能的描述。性能描述如表 9-4 所示。

表 9-4　性能一般场景的表格描述

场景的部分	可能的值
刺激源	系统外部或内部
刺激	定期事件、随机事件、偶然事件
制品	系统
环境	正常模式、超载模式
响应	处理刺激、改变服务级别
响应度量	度量等待、期限、吞吐量、缺失率、数据丢失等

对于性能一般场景的图示及性能具体场景，读者可仿照前面可用性的描述方式，自行练习。

（2）性能战术。性能与时间相关，影响事件的响应时间有两个基本因素。

资源消耗：事件到达后进入一系列的处理程序，每一步处理都要占用资源，而且在处理过程中消息在各构件之间转换，这些转换也需要占用资源。

闭锁时间：指对事件处理时碰到了资源争用、资源不可用或对其他计算的依赖等情况，就产生了等待时间。

性能的战术有如下几种。

① 资源需求

减少处理事件流所需的资源：提高计算效率（如改进算法）、减少计算开销（如在可修改性与性能之间权衡，减少不必要的代理构件）。

减少所处理事件的数量：管理事件率、控制采样频率。

控制资源的使用：限制执行时间（如减少迭代次数）、限制队列大小。

② 资源管理

引入并发：引入并发对负载平衡很重要。

维持数据或计算的多个副本：C/S 结构中客户机 C 就是计算的副本，它能减少服务器计算的压力；高速缓存可以存放数据副本（在不同速度的存储库之间的缓冲）。

增加可用资源：在成本允许时，尽量使用速度更快的处理器、内存和网络。

③ 资源仲裁

资源仲裁战术是通过如下调度策略来实现的。

先进/先出（FIFO）；

固定优先级调度：先给事件分配特定的优先级，再按优先级高低顺序分配资源；

动态优先级调度：轮转调度、时限时间最早优先；

静态调度：可以离线确定调度。

4．安全性及其实现战术

（1）安全性的描述。安全性的描述如表 9-5 所示。

表 9-5　安全性一般场景的表格描述

场景的部分	可能的值
刺激源	对敏感资源进行访问的人或系统（合法的、非法的）
刺激	试图：显示数据、改变/删除数据、访问系统服务、降低系统服务的可用性
制品	系统服务、系统中的数据
环境	在线或离线、联网或断网、有或无防火墙
响应	对用户身份验证；阻止或允许对数据或服务的访问；授予可回收访问权；加密信息；限制服务的可用性；通知用户或系统
响应度量	增加安全性的成本；检测或确定攻击的可能性；降低服务级别后的成功率；恢复数据/服务

对于安全性一般场景的图示及安全性具体场景，读者可仿照前面可用性的描述方式，自行练习。

（2）安全性战术：包括抵抗攻击、检测攻击和从攻击中恢复。

① 抵抗攻击

对用户进行身份验证：包括动态密码、一次性密码、数字证书及生物识别等；

对用户进行授权：即对用户的访问进行控制管理；

维护数据的机密性：一般通过对数据和通信链路进行加密来实现；

维护完整性：对数据添加校验或哈希值；

限制暴露的信息；

限制访问：如用防火墙、DMZ 策略。

② 检测攻击。一般通过“入侵检测”系统进行过滤、比较通信模式与历史基线等方法。

③ 从攻击中恢复。

恢复：与可用性中的战术相同；

识别攻击者：作为审计追踪，用于预防性或惩罚性目的。

5．可测试性及其实现战术

（1）可测试性的描述。可测试性的描述如表 9-6 所示。

表 9-6　可测试性一般场景的表格描述

场景的部分	可能的值
刺激源	各类测试人员（单元测试、集成测试、验收、用户）
刺激	一种测试
制品	设计、代码段、完整的应用
环境	设计时、开发时、编译时、部署时
响应	提供测试的状态值、测试环境与案例的准备
响应度量	测试成本、出现故障的概率、执行时间等

对于可测试性一般场景的图示及可测试性具体场景，读者可仿照前面可用性的描述方式，自行练习。

（2）可测试性战术：包括输入/输出和内部监控。

① 输入/输出

记录/回放：指捕获跨接口的信息，并将其作为测试专用软件的输入；

将接口与实现分离：允许使用实现的替代（模拟器）来支持各种测试目的；

优化访问线路/接口：用测试工具来捕获或赋予构件的变量值。

② 内部监控。当监视器处于激活状态时，记录事件（如通过接口的信息）。

6．易用性及其实现战术

（1）易用性的描述。易用性的描述如表 9-7 所示。

表 9-7　易用性一般场景的表格描述

场景的部分	可能的值
刺激源	最终用户
刺激	学习系统特性、有效使用系统、使错误的影响最低、适配系统、对系统满意
制品	系统
环境	运行时或配置时
响应	支持“学习系统特性”的响应：界面为用户所熟悉或使用帮助系统 支持“有效使用系统”的响应：数据/命令聚合或复用；界面是导航；操作的一致性；多个活动同时进行 支持“使错误的影响最低”的响应：撤销/取消；从故障中恢复；识别并纠正用户错误；验证系统资源 支持“适配系统”的响应：定制能力；国际化 支持“对系统满意”的响应：显示系统状态；与用户的节奏合拍
响应度量	从最终用户的角度进行度量，如：学习成本、错误数量、解决问题的数量、满意度等

对于易用性一般场景的图示及易用性具体场景，读者可仿照前面可用性的描述方式，自行练习。

（2）易用性战术：包括运行时战术、设计时战术和支持用户主动操作。

① 运行时战术

任务的模型：维护任务的信息，使系统了解用户试图做什么，并提供各种协助；

用户的模型：维护用户的信息，例如使系统以用户可以阅读页面的速度滚动页面；

系统的模型：维护系统的信息，它确定了期望的系统行为，并向用户提供反馈。

② 设计时战术。将用户接口与应用的其余部分分离开来，预计用户接口会频繁发生变化，因此，单独维护用户接口代码将实现变更局部化。这与可修改性相关。

③ 支持用户主动操作。支持用户的主动操作，如支持“取消”、“撤销”、“聚合”和 “显示多个视图”。

9.3 软件架构风格

软件架构设计的一个核心问题是能否使用重复的软件架构模式，即能否达到架构级别的软件重用。也就是说，能否在不同的软件系统中，使用同一架构。基于这个目的，学者们开始研究和实践软件架构的风格和类型问题。

软件架构风格是描述某一特定应用领域中系统组织方式的惯用模式（ idiomatic paradigm）。架构风格定义了一个系统家族，即一个架构定义一个词汇表和一组约束。词汇表中包含一些构件和连接件类型，而这组约束指出系统是如何将这些构件和连接件组合起来的。架构风格反映了领域中众多系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成一个完整的系统。按这种方式理解，软件架构风格定义了用于描述系统的术语表和一组指导构建系统的规则。

对软件架构风格的研究和实践促进了对设计的重用，一些经过实践证实的解决方案也可以可靠地用于解决新的问题。架构风格不变的部分使不同的系统可以共享同一个实现代码。只要系统是使用常用的、规范的方法来组织，就可使别的设计者很容易地理解系统的架构。例如，如果某人把系统描述为客户/服务器模式，则不必给出设计细节，我们立刻就会明白系统是如何组织和工作的。

软件架构风格为大粒度的软件重用提供了可能。然而，对于应用架构风格来说，由于视点的不同，系统设计师有很大的选择余地。要为系统选择或设计某一个架构风格，必须根据特定项目的具体特点，进行分析比较后再确定，架构风格的使用几乎完全是特定的。

9.3.1 软件架构风格分类

讨论架构风格时要回答的问题是：

（1）设计词汇表是什么？

（2）构件和连接件的类型是什么？

（3）可容许的结构模式是什么？

（4）基本的计算模型是什么？

（5）风格的基本不变性是什么？

（6）其使用的常见例子是什么？

（7）使用此风格的优缺点是什么？

（8）其常见的特例是什么？

这些问题的回答包括了架构风格的最关键的四要素内容，即提供一个词汇表、定义一套配置规则、定义一套语义解释原则和定义对基于这种风格的系统所进行的分析。Garlan 和 Shaw 根据此框架给出了通用架构风格的分类：

（1）数据流风格：批处理序列；管道/过滤器。

（2）调用/返回风格：主程序/子程序；面向对象风格；层次结构。

（3）独立构件风格：进程通信；事件系统。

（4）虚拟机风格：解释器；基于规则的系统。

（5）仓库风格：数据库系统；超文本系统；黑板系统。

9.3.2 数据流风格

数据流风格的软件架构是一种最常见，结构最为简单的软件架构。这样的架构下，所有的数据按照流的形式在执行过程中前进，不存在结构的反复和重构，就像工厂中的汽车流水线一样，数据就像汽车零部件一样在流水线的各个节点上被加工，最终输出所需要的结果（一部完整的汽车）。在流动过程中，数据经过序列间的数据处理组件进行处理，然后将处理结果向后传送，最后进行输出。

数据流风格架构主要包括两种具体的架构风格：批处理序列和管道-过滤器。

1. 批处理序列

批处理风格的每一步处理都是独立的，并且每一步是顺序执行的。只有当前一步处理完，后一步处理才能开始。数据传送在步与步之间作为一个整体。（组件为一系列固定顺序的计算单元，组件间只通过数据传递交互。每个处理步骤是一个独立的程序，每一步必须在前一步结束后才能开始，数据必须是完整的，以整体的方式传递）批处理的典型应用：

（1）经典数据处理；

（2）程序开发；

（3）Windows 下的 BAT 程序就是这种应用的典型实例。

2. 管道和过滤器

在管道/过滤器风格的软件架构中，每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。因此，这里的构件被称为过滤器，这种风格的连接件就像是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。此风格特别重要的过滤器必须是独立的实体，它不能与其他的过滤器共享数据，而且一个过滤器不知道它上游和下游的标识。一个管道/过滤器网络输出的正确性并不依赖于过滤器进行增量计算过程的顺序。

图 9-5 是管道/过滤器风格的示意图。一个典型的管道/过滤器架构的例子是以 UNIX shell 编写的程序。UNIX 既提供一种符号，以连接各组成部分（UNIX 的进程），又提供某种进程运行时机制以实现管道。另一个著名的例子是传统的编译器。传统的编译器一直被认为是一种管道系统，在该系统中，一个阶段（包括词法分析、语法分析、语义分析和代码生成）的输出是另一个阶段的输入。

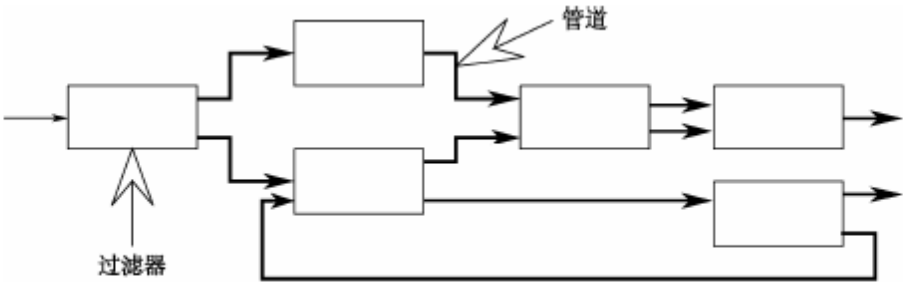


图 9-5 管道/过滤器风格的架构

管道/过滤器风格的软件架构具有许多很好的特点：

（1）使得软构件具有良好的隐蔽性和高内聚、低耦合的特点；

（2）允许设计者将整个系统的输入/输出行为看成是多个过滤器的行为的简单合成；

（3）支持软件重用。只要提供适合在两个过滤器之间传送的数据，任何两个过滤器都可被连接起来；

（4）系统维护和增强系统性能简单。新的过滤器可以添加到现有系统中来；旧的可以被改进的过滤器替换掉；

（5）允许对一些如吞吐量、死锁等属性的分析；

（6）支持并行执行。每个过滤器是作为一个单独的任务完成，因此可与其他任务并行执行。

但是，这样的系统也存在着若干不利因素。

（1）通常导致进程成为批处理的结构。这是因为虽然过滤器可增量式地处理数据，但它们是独立的，所以设计者必须将每个过滤器看成一个完整的从输入到输出的转换；

（2）不适合处理交互的应用。当需要增量地显示改变时，这个问题尤为严重；

（3）因为在数据传输上没有通用的标准，每个过滤器都增加了解析和合成数据的工作，这样就导致了系统性能下降，并增加了编写过滤器的复杂性。

3. 批处理序列风格与管道过滤器风格对比

把批处理序列风格与管道过滤器风格比较：

共同点：把任务分成一系列固定顺序的计算单元（组件）。组件间只通过数据传递交互。

区别：批处理是全部的、高潜伏性的，输入时可随机存取，无合作性、无交互性。而管道过滤器是递增的，数据结果延迟小，输入时处理局部化，有反馈、可交互。批处理强调数据传送在步与步之间作为一个整体，而管理过滤器无此要求。

9.3.3 调用/返回风格

调用返回风格顾名思义，就是指在系统中采用了调用与返回机制。利用调用-返回实际上是一种分而治之的策略，其主要思想是将一个复杂的大系统分解为一些子系统，以便降低复杂度，并且增加可修改性。程序从其执行起点开始执行该构件的代码，程序执行结束，将控制返回给程序调用构件。

调用/返回风格架构主要包括三种具体的架构风格：主程序/子程序；面向对象风格；层次结构。

1. 主程序/子程序

主程序/子程序风格是结构化开发时期的经典架构风格。这种风格一般采用单线程控制，把问题划分为若干处理步骤，构件即为主程序和子程序。子程序通常可合成为模块。过程调用作为交互机制，即充当连接件。调用关系具有层次性，其语义逻辑表现为子程序的正确性，取决于它调用的子程序的正确性。

2. 面向对象风格

抽象数据类型概念对软件系统有着重要作用，目前软件界已普遍使用面向对象系统。这种风格建立在数据抽象和面向对象的基础上，数据的表示方法和它们的相应操作封装在一个抽象数据类型或对象中。这种风格的构件是对象，或者说是抽象数据类型的实例。对象是一

种被称作管理者的构件，因为它负责保持资源的完整性。对象是通过函数和过程的调用来交互的。

图 9-6 是数据抽象和面向对象风格的示意图。

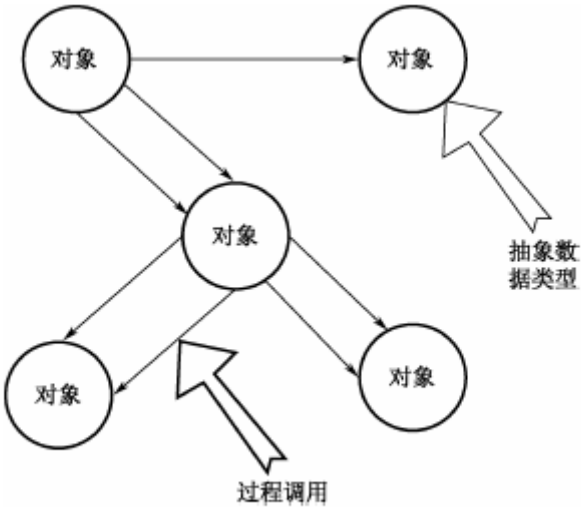


图 9-6　数据抽象和面向对象风格的架构

这种风格的两个重要特征为：

（1）对象负责维护其表示的完整性；

（2）对象的表示对其他对象而言是隐蔽的。因为一个对象对它的客户隐藏了自己的表示，所以这些对象可以不影响它的客户就能改变其实现方法。

面向对象的系统有许多优点，并早已为人所知：

（1）因为对象对其他对象隐藏它的表示，所以可以改变一个对象的表示，而不影响其他的对象；

（2）设计者可将一些数据存取操作的问题分解成一些交互的代理程序的集合。

但是，面向对象的系统也存在着某些问题：

（1）为了使一个对象和另一个对象通过过程调用等进行交互，必须知道对象的标识。只要一个对象的标识改变了，就必须修改所有其他明确调用它的对象；

（2）必须修改所有显式调用它的其他对象，并消除由此带来的一些副作用。例如，如果 A 使用了对象 B，C 也使用了对象 B，那么，C 对 B 的使用所造成的对 A 的影响可能是料想不到的。

2.　层次结构风格

层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。在一些层次系统中，除了一些精心挑选的输出函数外，内部的层只对相邻的层可见。这样的系统中构件在一

些层实现了虚拟机（在另一些层次系统中层是部分不透明的）。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。

这种风格支持基于可增加抽象层的设计。这样，允许将一个复杂问题分解成一个增量步骤序列的实现。由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。

图 9-7 是层次系统风格的示意图。层次系统最广泛的应用是分层通信协议。在这一应用领域中，每一层提供一个抽象的功能，作为上层通信的基础。较低的层次定义低层的交互，最低层通常只定义硬件物理连接。

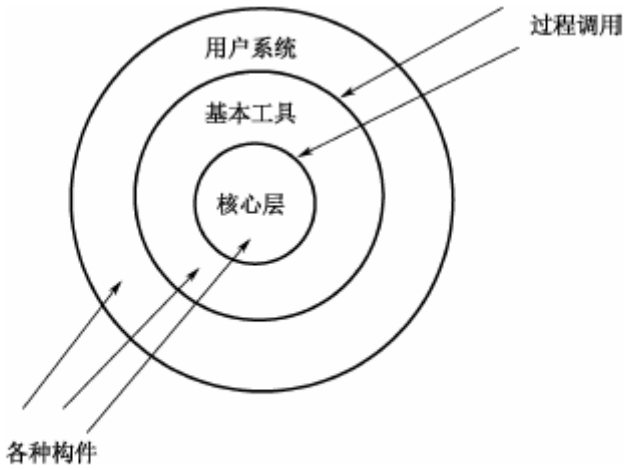


图 9-7　层次系统风格的架构

层次系统有许多可取的属性：

（1）支持基于抽象程度递增的系统设计，使设计者可以把一个复杂系统按递增的步骤进行分解；

（2）支持功能增强，因为每一层至多和相邻的上下层交互，因此功能的改变最多影响相邻的上下层；

（3）支持重用。只要提供的服务接口定义不变，同一层的不同实现可以交换使用。这样，就可以定义一组标准的接口，而允许各种不同的实现方法。

但是，层次系统也有其不足之处：

（1）并不是每个系统都可以很容易地划分为分层的模式，甚至即使一个系统的逻辑结构是层次化的，出于对系统性能的考虑，系统设计师不得不把一些低级或高级的功能综合起来；

（2）很难找到一个合适的、正确的层次抽象方法。

9.3.4 独立构件风格

独立构件风格主要强调系统中的每个构件都是相对独立的个体，它们之间不直接通信，以降低耦合度，提升灵活性。独立构件风格主要包括：进程通讯和事件系统子风格。

1. 进程通信架构风格进程通信架构风格：构件是独立的过程，连接件是消息传递。这种风格的特点是构件通常是命名过程，消息传递的方式可以是点到点、异步和同步方式及远过程调用等。

2. 事件系统风格基于事件的隐式调用风格的思想是构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中的其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。

从架构上说，这种风格的构件是一些模块，这些模块既可以是一些过程，又可以是一些事件的集合。过程可以用通用的方式调用，也可以在系统事件中注册一些过程，当发生这些事件时，过程被调用。

基于事件的隐式调用风格的主要特点是事件的触发者并不知道哪些构件会被这些事件影响。这样不能假定构件的处理顺序，甚至不知道哪些过程会被调用，因此，许多隐式调用的系统也包含显式调用作为构件交互的补充形式。

支持基于事件的隐式调用的应用系统很多。例如，在编程环境中用于集成各种工具，在数据库管理系统中确保数据的一致性约束，在用户界面系统中管理数据，以及在编辑器中支持语法检查。例如在某系统中，编辑器和变量监视器可以登记相应 Debugger 的断点事件。当 Debugger 在断点处停下时，它声明该事件，由系统自动调用处理程序，如编辑程序可以卷屏到断点，变量监视器刷新变量数值。而 Debugger 本身只声明事件，并不关心哪些过程会启动，也不关心这些过程做什么处理。

隐式调用系统的主要优点有：

（1）为软件重用提供了强大的支持。当需要将一个构件加入现存系统中时，只需将它注册到系统的事件中。

（2）为改进系统带来了方便。当用一个构件代替另一个构件时，不会影响到其他构件的接口。

隐式调用系统的主要缺点有：

（1）构件放弃了对系统计算的控制。一个构件触发一个事件时，不能确定其他构件是

否会响应它。而且即使它知道事件注册了哪些构件的过程，它也不能保证这些过程被调用的顺序。

（2）数据交换的问题。有时数据可被一个事件传递，但另一些情况下，基于事件的系统必须依靠一个共享的仓库进行交互。在这些情况下，全局性能和资源管理便成了问题。

（3）既然过程的语义必须依赖于被触发事件的上下文约束，关于正确性的推理存在问题。

9.3.5 虚拟机风格

虚拟机风格的基本思想是人为构建一个运行环境，在这个环境之上，可以解析与运行自定义的一些语言，这样来增加架构的灵活性，虚拟机风格主要包括解释器和规则为中心两种架构风格。

1.解释器

一个解释器通常包括完成解释工作的解释引擎，一个包含将被解释的代码的存储区，一个记录解释引擎当前工作状态的数据结构，以及一个记录源代码被解释执行进度的数据结构。

具有解释器风格的软件中含有一个虚拟机，可以仿真硬件的执行过程和一些关键应用。解释器通常被用来建立一种虚拟机以弥合程序语义与硬件语义之间的差异。其缺点是执行效率较低。典型的例子是专家系统。

2. 规则为中心

基于规则的系统包括规则集、规则解释器、规则/数据选择器及工作内存。

9.3.6 仓库风格

在仓库（repository）风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据存储上执行，仓库与外构件间的相互作用在系统中会有大的变化。

仓库风格包括的子风格有：数据库系统、超文本系统、黑板风格。

数据库架构是库风格最常见的形式。构件主要有两大类，一个是中央共享数据源，保存当前系统的数据状态；另一个是多个独立处理元素，处理元素对数据元素进行操作。而超文本系统的典型代表，就是早期的静态网页。三种架构子风格中，最复杂的是黑板系统。

黑板系统是在抽象与总结语言理解系统 HEARSAY-11 的基础上产生的，适合于解决复杂的非结构化的问题，能在求解过程中综合运用多种不同知识源，使得问题的表达、组织和求

解变得比较容易。黑板系统是一种问题求解模型，是组织推理步骤、控制状态数据和问题求解之领域知识的概念框架。它将问题的解空间组织成一个或多个应用相关的分级结构。分级结构的每一层信息由一个唯一的词汇来描述，它代表了问题的部分解。领域相关的知识被分成独立的知识模块，它将某一层次中的信息转换成同层或相邻层的信息。各种应用通过不同知识表达方法、推理框架和控制机制的组合来实现。影响黑板系统设计的最大因素是应用问题本身的特性，但是支撑应用程序的黑板体系结构有许多相似的特征和构件。

对于特定应用问题，黑板系统可通过选取各种黑板、知识源和控制模块的构件来设计；也可以利用预先定制的黑板体系结构的编程环境。图 9-8 是黑板系统的组成。黑板系统的传统应用是信号处理领域，如语音和模式识别。另一应用是松耦合代理数据共享存取。

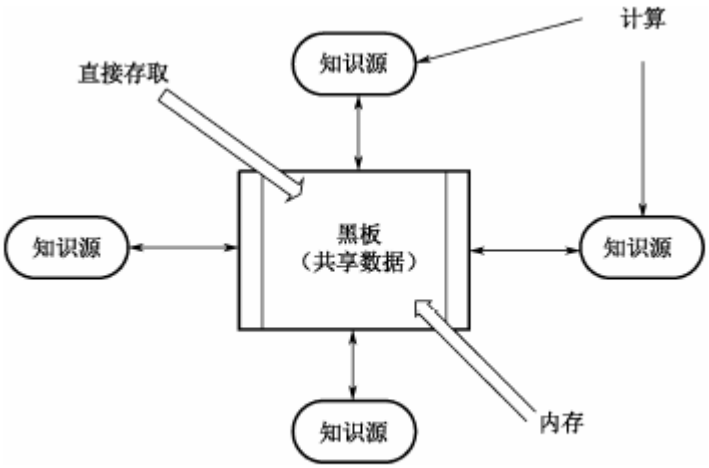


图 9-8　黑板系统的组成

我们从图 9-8 中可以看出，黑板系统主要由三部分组成：

（1）知识源。知识源中包含独立的、与应用程序相关的知识，知识源之间不直接进行通信，它们之间的交互只通过黑板来完成。

（2）黑板数据结构。黑板数据是按照与应用程序相关的层次来组织的解决问题的数据，知识源通过不断地改变黑板数据来解决问题。

（3）控制。控制完全由黑板的状态驱动，黑板状态的改变决定使用的特定知识。

9.4 层次系统架构风格

本章 9.3.3 节中已对层次系统架构风格进行了初步的介绍，下面将对几种经典的层次风格进行详细说明。

9.4.1 二层及三层 C/S 架构风格

C/S 架构是基于资源不对等，且为实现共享而提出来的，是 20 世纪 90 年代成熟起来的技术，C/S 结构将应用一分为二，服务器（后台）负责数据管理，客户机（前台）完成与用户的交互任务。

C/S 软件架构具有强大的数据操作和事务处理能力，模型思想简单，易于人们理解和接受。但随着企业规模的日益扩大，软件的复杂程度不断提高，传统的二层 C/S 结构存在以下几个局限：

二层 C/S 结构为单一服务器且以局域网为中心，所以难以扩展至大型企业广域网或 Internet；

软、硬件的组合及集成能力有限；

服务器的负荷太重，难以管理大量的客户机，系统的性能容易变坏；

数据安全性不好。因为客户端程序可以直接访问数据库服务器，那么，在客户端计算机上的其他程序也可想办法访问数据库服务器，从而使数据库的安全性受到威胁。

正是因为二层 C/S 有这么多缺点，因此，三层 C/S 结构应运而生。三层 C/S 结构是将应用功能分成表示层、功能层和数据层三个部分，如图 9-9 所示。

表示层是应用的用户接口部分，它担负着用户与应用间的对话功能。它用于检查用户从键盘等输入的数据，并显示应用输出的数据。在变更用户接口时，只需改写显示控制和数据检查程序，而不影响其他两层。检查的内容也只限于数据的形式和取值的范围，不包括有关业务本身的处理逻辑。

功能层相当于应用的本体，它是将具体的业务处理逻辑编入程序中。而处理所需的数据则要从表示层或数据层取得。表示层和功能层之间的数据交往要尽可能简洁。

数据层就是数据库管理系统，负责管理对数据库数据的读写。数据库管理系统必须能迅速执行大量数据的更新和检索。因此，一般从功能层传送到数据层的要求大都使用 SQL 语言。

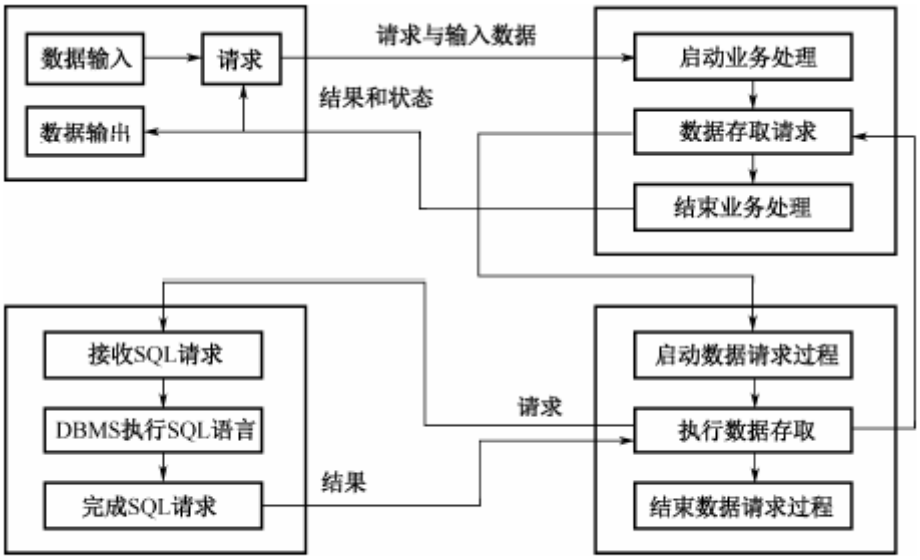


图 9-9　三层 C/S 结构示意图

三层 C/S 的解决方案是：对这三层进行明确分割，并在逻辑上使其独立。原来的数据层作为数据库管理系统已经独立出来，所以，关键是要将表示层和功能层分离成各自独立的程序，并且还要使这两层间的接口简洁明了。

一般情况是只将表示层配置在客户机中，如果将功能层也放在客户机中，与二层 C/S 结构相比，其程序的可维护性要好得多，但是其他问题并未得到解决。客户机的负荷太重，其业务处理所需的数据要从服务器传给客户机，所以系统的性能容易变差。

如果将功能层和数据层分别放在不同的服务器中，则服务器和服务器之间也要进行数据传送。但是，由于在这种形态中三层是分别放在各自不同的硬件系统上的，所以灵活性很高，能够适应客户机数目的增加和处理负荷的变动。例如，在追加新业务处理时，可以相应增加装载功能层的服务器。因此，系统规模越大这种形态的优点就越显著。

9.4.2　B/S 架构风格

在三层 C/S 架构中，表示层负责处理用户的输入和向客户的输出（出于效率的考虑，它可能在向上传输用户的输入前进行合法性验证）。功能层负责建立数据库的连接，根据用户的请求生成访问数据库的 SQL 语句，并把结果返回给客户端。数据层负责实际的数据库存储和检索，响应功能层的数据处理请求，并将结果返回给功能层。

浏览器/服务器（Browser/Server，简称 B/S）风格就是上述三层应用结构的一种实现方式，其具体结构为：浏览器/Web 服务器/数据库服务器。采用 B/S 结构的计算机应用系统的基本框架如图 9-10 所示。

B/S 架构主要是利用不断成熟的 WWW 浏览器技术，结合浏览器的多种脚本语言，用通用浏览器就实现了原来需要复杂的专用软件才能实现的强大功能，并节约了开发成本。从某种程度上来说，B/S 结构是一种全新的软件架构。

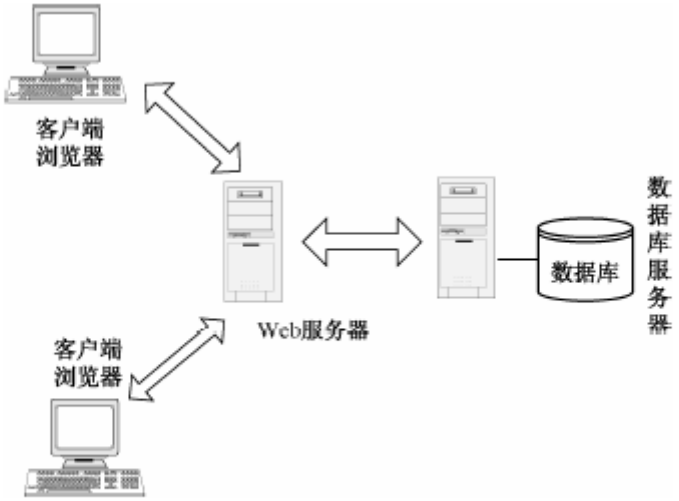


图 9-10　B/S 模式结构

在 B/S 结构中，除了数据库服务器外，应用程序以网页形式存放于 Web 服务器上，用户运行某个应用程序时只需在客户端上的浏览器中键入相应的网址，调用 Web 服务器上的应用程序并对数据库进行操作完成相应的数据处理工作，最后将结果通过浏览器显示给用户。可以说，在 B/S 模式的计算机应用系统中，应用（程序）在一定程度上具有集中特征。

基于 B/S 架构的软件，系统安装、修改和维护全在服务器端解决。用户在使用系统时，仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。B/S 架构还提供了异种机、异种网、异种应用服务的联机、联网、统一服务的最现实的开放性基础。

B/S 结构出现之前，管理信息系统的功能覆盖范围主要是组织内部。B/S 结构的“零客户端”方式，使组织的供应商和客户（这些供应商和客户有可能是潜在的，也就是说可能是事先未知的!）的计算机方便地成为管理信息系统的客户端，进而在限定的功能范围内查询组织相关信息，完成与组织的各种业务往来的数据交换和处理工作，扩大了组织计算机应用系统的功能覆盖范围，可以更加充分利用网络上的各种资源，同时应用程序维护的工作量也大大减少。另外，B/S 结构的计算机应用系统与 Internet 的结合也使新近提出的一些新的企业计算机应用（如电子商务，客户关系管理）的实现成为可能。

与 C/S 架构相比，B/S 架构也有许多不足之处，例如:

（1）B/S 架构缺乏对动态页面的支持能力，没有集成有效的数据库处理功能。

（2）采用 B/S 架构的应用系统，在数据查询等响应速度上，要远远地低于 C/S 架构。

（3）B/S 架构的数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理（OnLine Transaction Processing，简称 OLTP）应用。

9.4.3 MVC 架构风格

MVC 全名是 Model ViewController，是模型(model)－视图(view)－控制器(controller)的缩写，它是分层架构风格的一种。MVC 是由挪威的计算机专家 Trygve M.H. Reenskau 于 1979 年提出的软件架构模式，MVC 最初用于 SmallTalk。MVC 提出的基本思想是进行关注点分离。一个典型的人机交互应用具有三个主要的关注点：数据在可视化界面上的呈现、UI 处理逻辑和业务逻辑。如果按传统的自治视图模式（即将与 UI 相关的逻辑都定义在针对视图的相关元素的事件上），将三者混合在一起，势必会带来一系列问题：

（1）业务逻辑是与 UI 无关的，应该最大限度地被重用。由于业务逻辑定义在自治视图中，相当于完全与视图本身绑定在一定，如果我们能够将 UI 的行为抽象出来，基于抽象化 UI 的处理逻辑也是可以被共享的。但是定义在自治视频中的 UI 处理逻辑完全丧失了重用的可能。

（2）业务逻辑具有最强的稳定性，UI 处理逻辑次之，而可视化界面上的呈现最差（比如我们经常会为了更好地呈现效果来调整 HTML）。如果将具有不同稳定性的元素融为一体，那么具有最差稳定性的元素决定了整体的稳定性。

（3）任何涉及 UI 的组件都不易测试。UI 是呈现给人看的，并且用于人机交互，用机器来模拟活生生的人来对组件实施自动化测试不是一件容易的事，自治视图严重损害了组件的可测试性。

正是为了解决以上的问题，所以我们需要采用关注点分离的方针来将可视化界面呈现、UI 处理逻辑和业务逻辑三者分离出来，并且采用合理的交互方式将它们之间的依赖降到最低。

MVC 中各个部分的分工与协作是这样的：

（1）Model 是对应用状态和业务功能的封装，我们可以将它理解为同时包含数据和行为的领域模型。Model 接受 Controller 的请求并完成相应的业务处理，在状态改变的时候向 View 发出相应的通知。

（2）View 实现可视化界面的呈现并捕捉最终用户的交互操作（例如鼠标和键盘的操作）。

（3）View 捕获到用户交互操作后会直接转发给 Controller，后者完成相应的 UI 逻辑。如果需要涉及业务功能的调用，Controller 会直接调用 Model。在完成 UI 处理后，Controller 会根据需要控制原 View 或者创建新的 View 对用户交互操作予以响应。

MVC 模式的基本结构如图 9-11 所示。

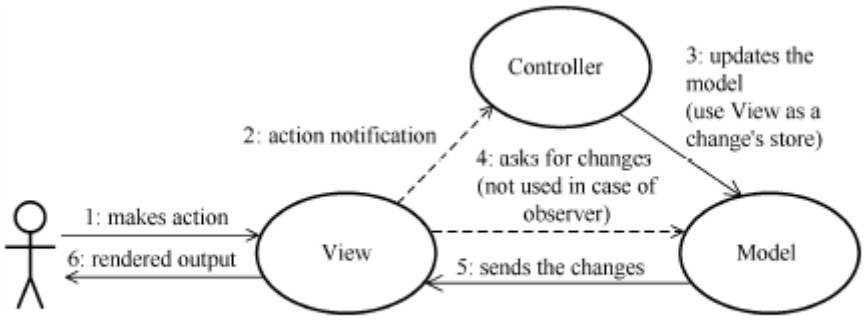


图 9-11　MVC 结构

9.4.4 MVP 架构风格

MVP 的全称为 Model-View-Presenter，Model 提供数据，View 负责显示，Controller/Presenter 负责逻辑的处理。MVP 是从经典的模式 MVC 演变而来，它们的基本思想有相通的地方：Controller/Presenter 负责逻辑的处理，Model 提供数据，View 负责显示。当然 MVP 与 MVC 也有一些显著的区别，MVC 模式中元素之间“混乱”的交互主要体现在允许 View 和 Model 直接进行“交流”，这在 MVP 模式中是不允许的。在 MVP 中 View 并不直接使用 Model，它们之间的通信是通过 Presenter （MVC 中的 Controller）来进行的，所有的交互都发生在 Presenter 内部，而在 MVC 中 View 会直接从 Model 中读取数据而不是通过 Controller。

MVP 不仅仅避免了 View 和 Model 之间的耦合，还进一步降低了 Presenter 对 View 的依赖。Presenter 依赖的是一个抽象化的 View，即 View 实现的接口 IView，这带来的最直接的好处，就是使定义在 Presenter 中的 UI 处理逻辑变得易于测试。由于 Presenter 对 View 的依赖行为定义在接口 IView 中，我们只需要一个实现了这个接口的 View 就能对 Presenter 进行测试。MVP 的结构如图 9-12 所示。

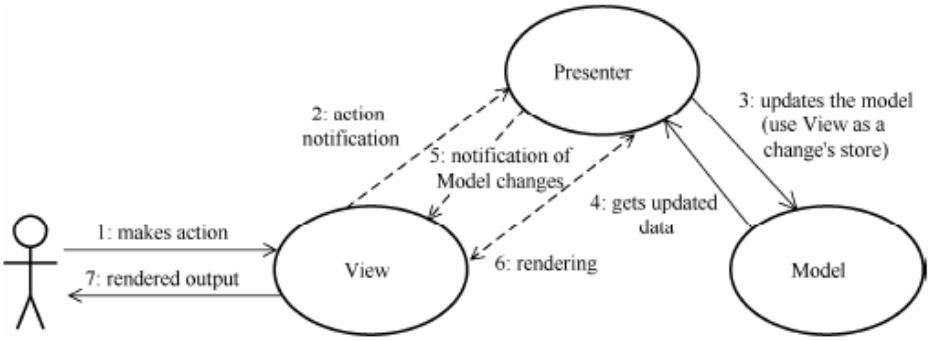


图 9-12　MVP 结构

下面我们来分析 MVP 的优缺点。MVP 的优点包括：

（1）模型与视图完全分离，我们可以修改视图而不影响模型。

（2）可以更高效地使用模型，因为所有的交互都发生在一个地方—Presenter 内部。

（3）我们可以将一个 Presenter 用于多个视图，而不需要改变 Presenter 的逻辑。这个特性非常的有用，因为视图的变化总是比模型的变化频繁。

（4）如果我们把逻辑放在 Presenter 中，那么我们就可以脱离用户接口来测试这些逻辑（单元测试）。

MVP 的缺点包括：

由于对视图的渲染放在了 Presenter 中，所以视图和 Presenter 的交互会过于频繁。还有一点需要明白，如果 Presenter 过多地渲染了视图，往往会使得它与特定的视图的联系过于紧密。一旦视图需要变更，那么 Presenter 也需要变更了。比如说，原本用来呈现 HTML 的 Presenter 现在也需要用于呈现 PDF 了，那么视图很有可能也需要变更。

9.5 面向服务的架构

迄今为止，对于面向服务的架构（Service-Oriented Architecture，SOA）还没有一个公认的定义。许多组织从不同的角度和不同的侧面对 SOA 进行了描述，较为典型的有以下三个：

（1）W3C 的定义：SOA 是一种应用程序架构，在这种架构中，所有功能都定义为独立的服务，这些服务带有定义明确的可调用接口，能够以定义好的顺序调用这些服务来形成业务流程。

（2）Service-architecture.com 的定义：服务是精确定义、封装完善、独立于其他服务所处环境和状态的函数。SOA 本质上是服务的集合，服务之间彼此通信，这种通信可能是简单的数据传送，也可能是两个或更多的服务协调进行某些活动。服务之间需要某些方法进行连接。

（3）Gartner 的定义：SOA 是一种 C/S 架构的软件设计方法，应用由服务和服务使用者组成，SOA 与大多数通用的 C/S 架构模型不同之处，在于它着重强调构件的松散耦合，并使用独立的标准接口。

9.5.1 SOA 概述

SOA 是一种在计算环境中设计、开发、部署和管理离散逻辑单元（服务）模型的方法。SOA 并不是一个新鲜事物，而只是面向对象模型的一种替代。虽然基于 SOA 的系统并不排除使用 OOD 来构建单个服务，但是其整体设计却是面向服务的。由于 SOA 考虑到了系统内的对象，所以虽然 SOA 是基于对象的，但是作为一个整体，它却不是面向对象的。

SOA 系统原型的一个典型例子是 CORBA，它已经出现很长时间，其定义的概念与 SOA 相似。SOA 建立在 XML 等新技术的基础上，通过使用基于 XML 的语言来描述接口，服务已经转到更动态且更灵活的接口系统中，CORBA 中的 IDL 无法与之相比。图 9-13 描述了一个完整的 SOA 模型。

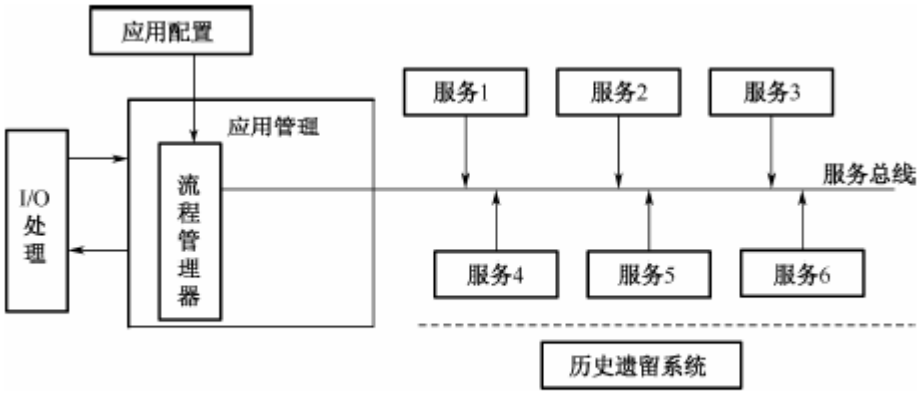


图 9-13　SOA 模型示例

在 SOA 模型中，所有的功能都定义成了独立的服务。服务之间通过交互和协调完成业务的整体逻辑。所有的服务通过服务总线或流程管理器来连接。这种松散耦合的架构使得各服务在交互过程中无需考虑双方的内部实现细节，以及部署在什么平台上。

1. 服务的基本结构

一个独立的服务基本结构如图 9-14 所示。

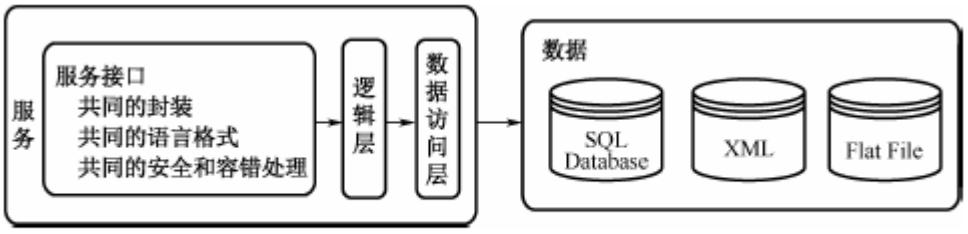


图 9-14　单个服务内部结构

由图 9-14 可以看出，服务模型的表示层从逻辑层分离出来，中间增加了服务对外的接口层。通过服务接口的标准化描述，使得服务可以提供给在任何异构平台和任何用户接口使用。这允许并支持基于服务的系统成为松散耦合、面向构件和跨技术实现，服务请求者很可能根本不知道服务在哪里运行、是由哪种语言编写的，以及消息的传输路径，而是只需要提出服务请求，然后就会得到答案。

2.SOA 设计原则

在 SOA 架构中，继承了来自对象和构件设计的各种原则，例如，封装和自我包含等。那些保证服务的灵活性、松散耦合和复用能力的设计原则，对 SOA 架构来说同样是非常重要的。关于服务，一些常见的设计原则如下：

（1）明确定义的接口。服务请求者依赖于服务规约来调用服务，因此，服务定义必须长时间稳定，一旦公布，不能随意更改；服务的定义应尽可能明确，减少请求者的不适当使用；不要让请求者看到服务内部的私有数据。

（2）自包含和模块化。服务封装了那些在业务上稳定、重复出现的活动和构件，实现服务的功能实体是完全独立自主的，独立进行部署、版本控制、自我管理和恢复。

（3）粗粒度。服务数量不应该太多，依靠消息交互而不是远程过程调用，通常消息量比较大，但是服务之间的交互频度较低。

（4）松耦合。服务请求者可见的是服务的接口，其位置、实现技术、当前状态和私有数据等，对服务请求者而言是不可见的。

（5）互操作性、兼容和策略声明。为了确保服务规约的全面和明确，策略成为一个越来越重要的方面。这可以是技术相关的内容，例如，一个服务对安全性方面的要求；也可以是与业务有关的语义方面的内容，例如，需要满足的费用或者服务级别方面的要求，这些策略对于服务在交互时是非常重要的。

3. 服务构件与传统构件

服务构件架构（Service Component Architecture，SCA）是基于 SOA 的思想描述服务之

间组合和协作的规范，它描述用于使用 SOA 构建应用程序和系统的模型。它可简化使用 SOA 进行的应用程序开发和实现工作。SCA 提供了构建粗粒度构件的机制，这些粗粒度构件由细粒度构件组装而成。SCA 将传统中间件编程从业务逻辑分离出来，从而使程序员免受其复杂性的困扰。它允许开发人员集中精力编写业务逻辑，而不必将大量的时间花费在更为底层的技术实现上。

SCA 服务构件与传统构件的主要区别在于，服务构件往往是粗粒度的，而传统构件以细粒度居多；服务构件的接口是标准的，主要是服务描述语言接口，而传统构件常以具体 API 形式出现；服务构件的实现与语言是无关的，而传统构件常绑定某种特定的语言；服务构件可以通过构件容器提供 QoS 的服务，而传统构件完全由程序代码直接控制。

9.5.2 SOA 的关键技术

SOA 伴随着无处不在的标准，为企业的现有资产或投资带来了更好的复用性。SOA 能够在最新的和现有的系统之上创建应用，借助现有的应用产生新的服务，为企业提供更好的灵活性来构建系统和业务流程。SOA 是一种全新的架构，为了支持其各种特性，相关的技术规范不断推出。与 SOA 紧密相关的技术主要有 UDDI、WSDL、SOAP 和 REST 等，而这些技术都是以 XML 为基础而发展起来的。

1. UDDI

UDDI（Universal DescriptionDiscovery and Integration，统一描述、发现和集成）提供了一种服务发布、查找和定位的方法，是服务的信息注册规范，以便被需要该服务的用户发现和使用它。UDDI 规范描述了服务的概念，同时也定义了一种编程接口。通过 UDDI 提供的标准接口，企业可以发布自己的服务供其他企业查询和调用，也可以查询特定服务的描述信息，并动态绑定到该服务上。

在 UDDI 技术规范中，主要包含以下三个部分的内容：

（1）数据模型。UDDI 数据模型是一个用于描述业务组织和服务的 XML Schema。

（2）API。UDDI API 是一组用于查找或发布 UDDI 数据的方法，UDDI API 基于 SOAP。

（3）注册服务。UDDI 注册服务是 SOA 中的一种基础设施，对应着服务注册中心的角色。

2.WSDL

WSDL（Web ServiceDescription Language，Web 服务描述语言）是对服务进行描述的语

言，它有一套基于 XML 的语法定义。WSDL 描述的重点是服务，它包含服务实现定义和服务接口定义，如图 9-15 所示。

图 9-15　基本服务描述

采用抽象接口定义对于提高系统的扩展性很有帮助。服务接口定义就是一种抽象的、可重用的定义，行业标准组织可以使用这种抽象的定义来规定一些标准的服务类型，服务实现者可以根据这些标准定义来实现具体的服务。

服务实现定义描述了给定服务提供者如何实现特定的服务接口。服务实现定义中包含服务和端口描述。一个服务往往会包含多个服务访问入口，而每个访问入口都会使用一个端口元素来描述，端口描述的是一个服务访问入口的部署细节，例如，通过哪个地址来访问，应当使用怎样的消息调用模式来访问等。

3.SOAP

SOAP（Simple ObjectAccess Protocol，简单对象访问协议）定义了服务请求者和服务提供者之间的消息传输规范。SOAP 用 XML 来格式化消息，用 HTTP 来承载消息。通过 SOAP，应用程序可以在网络中进行数据交换和远程过程调用（Remote Procedure Call， RPC）。SOAP 主要包括以下四个部分：

（1）封装。SOAP 封装定义了一个整体框架，用来表示消息中包含什么内容，谁来处理这些内容，以及这些内容是可选的还是必需的。

（2）编码规则。SOAP 编码规则定义了一种序列化的机制，用于交换系统所定义的数据类型的实例。

（3）RPC 表示。SOAP RPC 表示定义了一个用来表示远程过程调用和应答的协议。

（4）绑定。SOAP 绑定定义了一个使用底层传输协议来完成在节点之间交换 SOAP 封装的约定。

SOAP 消息基本上是从发送端到接收端的单向传输，但它们常常结合起来执行类似于请求/应答的模式。所有的 SOAP 消息都使用 XML 进行编码。SOAP 消息包括以下三个部分：

（1）封装（信封）。封装的元素名是 Envelope，在表示消息的 XML 文档中，封装是顶层元素，在 SOAP 消息中必须出现。

（2）SOAP 头。SOAP 头的元素名是 Header，提供了向 SOAP 消息中添加关于这条 SOAP 消息的某些要素的机制。SOAP 定义了少量的属性用来表明这项要素是否可选以及由谁来处理。SOAP 头在 SOAP 消息中可能出现，也可能不出现。如果出现的话，必须是 SOAP 封装元素的第一个直接子元素。

（3）SOAP 体。SOAP 体的元素名是 Body，是包含消息的最终接收者想要的信息的容器。SOAP 体在 SOAP 消息中必须出现且必须是 SOAP 封装元素的直接子元素。如果有头元素，则 SOAP 体必须直接跟在 SOAP 头元素之后；如果没有头元素，则 SOAP 体必须是 SOAP 封装元素的第一个直接子元素。

4.REST

REST（RepresentationalState Transfer，表述性状态转移）是一种只使用 HTTP 和 XML 进行基于 Web 通信的技术，可以降低开发的复杂性，提高系统的可伸缩性。它的简单性和缺少严格配置文件的特性，使它与 SOAP 很好地隔离开来，REST 从根本上来说只支持几个操作（POST、GET、PUT 和 DELETE），这些操作适用于所有的消息。REST 提出了如下一些设计概念和准则：

（1）网络上的所有事物都被抽象为资源。

（2）每个资源对应一个唯一的资源标识。

（3）通过通用的连接件接口对资源进行操作。

（4）对资源的各种操作不会改变资源标识。

（5）所有的操作都是无状态的。

9.5.3 SOA 的实现方法

SOA 只是一种概念和思想，需要借助于具体的技术和方法来实现它。从本质上来看，SOA 是用本地计算模型来实现一个分布式的计算应用，也有人称这种方法为“本地化设计，分布式工作”模型。CORBA、DCOM 和 EJB 等都属于这种解决方式，也就是说，SOA 最终可以基于这些标准来实现。有关这些标准的知识，已经在 13.1.1 节中详细介绍。另外，这些标准分别使用的 ORB、RPC 和 RMI（Remote Method Invocation，远程方法调用）等技术，将在 17.1.2 节中介绍，此处不再赘述。

从逻辑上和高层抽象来看，目前，实现 SOA 的方法也比较多，其中主流方式有 Web Service、企业服务总线和服务注册表。

1.Web Service

在 Web Service（Web 服务）的解决方案中，一共有三种工作角色，其中服务提供者和服务请求者是必需的，服务注册中心是一个可选的角色。它们之间的交互和操作构成了 SOA 的一种实现架构，如图 9-16 所示。

（1）服务提供者。服务提供者是服务的所有者，该角色负责定义并实现服务，使用 WSDL 对服务进行详细、准确、规范地描述，并将该描述发布到服务注册中心，供服务请求者查找并绑定使用。

（2）服务请求者。服务请求者是服务的使用者，虽然服务面向的是程序，但程序的最终使用者仍然是用户。从架构的角度看，服务请求者是查找、绑定并调用服务，或与服务进行交互的应用程序。服务请求者角色可以由浏览器来担当，由人或程序（例如，另外一个服务）来控制。

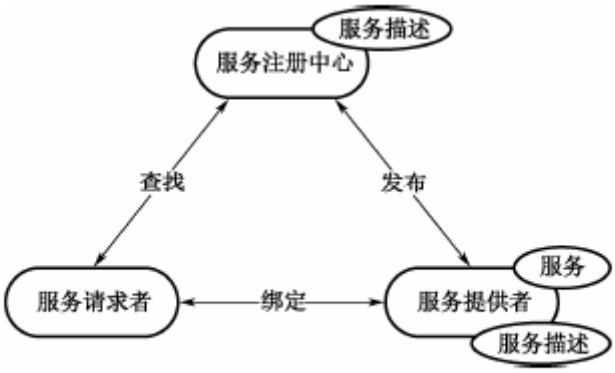


图 9-16　Web Service 模型

（3）服务注册中心。服务注册中心是连接服务提供者和服务请求者的纽带，服务提供者在此发布他们的服务描述，而服务请求者在服务注册中心查找他们需要的服务。不过，在某些情况下，服务注册中心是整个模型中的可选角色。例如，如果使用静态绑定的服务，服务提供者则可以把描述直接发送给服务请求者。

Web Service 模型中的操作包括发布、查找和绑定，这些操作可以单次或反复出现。

（1）发布。为了使用户能够访问服务，服务提供者需要发布服务描述，以便服务请求者可以查找它。

（2）查找。在查找操作中，服务请求者直接检索服务描述或在服务注册中心查询所要求的服务类型。对服务请求者而言，可能会在生命周期的两个不同阶段中涉及查找操作，首

先是在设计阶段，为了程序开发而查找服务的接口描述；其次是在运行阶段，为了调用而查找服务的位置描述。

（3）绑定。在绑定操作中，服务请求者使用服务描述中的绑定细节来定位、联系并调用服务，从而在运行时与服务进行交互。绑定可以分为动态绑定和静态绑定。在动态绑定中，服务请求者通过服务注册中心查找服务描述，并动态地与服务交互；在静态绑定中，服务请求者已经与服务提供者达成默契，通过本地文件或其他方式直接与服务进行绑定。

在采用 Web Service 作为 SOA 的实现技术时，应用系统大致可以分为六个层次，分别是底层传输层、服务通信协议层、服务描述层、 服务层、业务流程层和服务注册层。

（1）底层传输层。底层传输层主要负责消息的传输机制，HTTP、JMS（Java Messaging Service，Java 消息服务）和 SMTP 都可以作为服务的消息传输协议，其中 HTTP 使用最广。

（2）服务通信协议层。服务通信协议层的主要功能是描述并定义服务之间进行消息传递所需的技术标准，常用的标准是 SOAP 和 REST 协议。

（3）服务描述层。服务描述层主要以一种统一的方式描述服务的接口与消息交换方式，相关的标准是 WSDL。

（4）服务层。服务层的主要功能是将遗留系统进行包装，并通过发布的 WSDL 接口描述被定位和调用。

（5）业务流程层。业务流程层的主要功能是支持服务发现，服务调用和点到点的服务调用，并将业务流程从服务的底层调用抽象出来。

（6）服务注册层的主要功能是使服务提供者能够通过 WSDL 发布服务定义，并支持服务请求者查找所需的服务信息。相关的标准是 UDDI。

2. 服务注册表

服务注册表（service registry）虽然也具有运行时的功能，但主要在 SOA 设计时使用。它提供一个策略执行点（Policy Enforcement Point，PEP），在这个点上，服务可以在 SOA 中注册，从而可以被发现和使用。服务注册表可以包括有关服务和相关构件的配置、依从性和约束文件。从理论上来说，任何帮助服务注册、发现和查找服务合约、元数据和策略的信息库、数据库、目录或其他节点都可以被认为是一个注册表。大多数商用服务注册产品支持服务注册、服务位置和服务绑定功能。

（1）服务注册。服务注册是指服务提供者向服务注册表发布服务的功能（服务合约），包括服务身份、位置、方法、绑定、配置、方案和策略等描述性属性。使用服务注册表实现 SOA 时，要限制哪些新服务可以向注册表发布、由谁发布以及谁批准和根据什么条件批准

等，以便使服务能够有序的注册。

（2）服务位置。服务位置是指服务使用者，帮助它们查询已注册的服务，寻找符合自身要求的服务。这种查找主要是通过检索服务合约来实现的，在使用服务注册表实现 SOA 时，需要规定哪些用户可以访问服务注册表，以及哪些服务属性可以通过服务注册表进行暴露等，以便服务能得到有效的、经过授权的使用。

（3）服务绑定。服务使用者利用查找到的服务合约来开发代码，开发的代码将与注册的服务进行绑定，调用注册的服务，以及与它们实现互动。可以利用集成的开发环境自动将新开发的服务与不同的新协议、方案和程序间通信所需的其他接口绑定在一起。

3. 企业服务总线

ESB 的概念是从 SOA 发展而来的，它是一种为进行连接服务提供的标准化的通信基础结构，基于开放的标准，为应用提供了一个可靠的、可度量的和高度安全的环境，并可帮助企业对业务流程进行设计和模拟，对每个业务流程实施控制和跟踪、分析并改进流程和性能。

在一个复杂的企业计算环境中，如果服务提供者和服务请求者之间采用直接的端到端的交互，那么随着企业信息系统的增加和复杂度的提高，系统之间的关联会逐渐变得非常复杂，形成一个网状结构，这将带来昂贵的系统维护费用，同时也使得 IT 基础设施的复用变得困难重重。ESB 提供了一种基础设施，消除了服务请求者与服务提供者之间的直接连接，使得服务请求者与服务提供者之间进一步解耦。

ESB 是由中间件技术实现并支持 SOA 的一组基础架构，是传统中间件技术与 XML、Web Service 等技术结合的产物，是在整个企业集成架构下的面向服务的企业应用集成机制。具体来说，ESB 具有以下功能：

（1）支持异构环境中的服务、消息和基于事件的交互，并且具有适当的服务级别和可管理性。

（2）通过使用 ESB，可以在几乎不更改代码的情况下，以一种无缝的非侵入方式使现有系统具有全新的服务接口，并能够在部署环境中支持任何标准。

（3）充当缓冲器的 ESB（负责在诸多服务之间转换业务逻辑和数据格式）与服务逻辑相分离，从而使不同的系统可以同时使用同一个服务，不用在系统或数据发生变化时，改动服务代码。

（4）在更高的层次，ESB 还提供诸如服务代理和协议转换等功能。允许在多种形式下通过像 HTTP、SOAP 和 JMS 总线的多种传输方式，主要是以网络服务的形式，为发表、注册、发现和使用企业服务或界面提供基础设施。

（5）提供可配置的消息转换翻译机制和基于消息内容的消息路由服务，传输消息到不同的目的地。

（6）提供安全和拥有者机制，以保证消息和服务使用的认证、授权和完整性。

在企业应用集成方面，与现存的、专有的集成解决方案相比，ESB 具有以下优势：

（1）扩展的、基于标准的连接。ESB 形成一个基于标准的信息骨架，使得在系统内部和整个价值链中可以容易地进行异步或同步数据交换。ESB 通过使用 XML、SOAP 和其他标准，提供了更强大的系统连接性。

（2）灵活的、服务导向的应用组合。基于 SOA，ESB 使复杂的分布式系统（包括跨多个应用、系统和防火墙的集成方案）能够由以前开发测试过的服务组合而成，使系统具有高度可扩展性。

（3）提高复用率，降低成本。按照 SOA 方法构建应用，提高了复用率，简化了维护工作，进而减少了系统总体成本。

（4）减少市场反应时间，提高生产率。ESB 通过构件和服务复用，按照 SOA 的思想简化应用组合，基于标准的通信、转换和连接来实现这些优点。

9.5.4 微服务

微服务顾名思义，就是很小的服务，所以它属于面向服务架构的一种。通俗一点来说，微服务类似于古代著名的发明，活字印刷术，每个服务都是一个组件，通过编排组合的方式来使用，从而真正做到了独立、解耦、组件化、易维护、可复用、可替换、高可用、最终达到提高交付质量、缩短交付周期的效果。

从专业的角度来看，微服务架构是一种架构模式，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合，为用户提供最终价值。每个服务运行在其独立的进程中，服务与服务间采用轻量级的通信机制互相沟通（通常是基于 HTTP 协议的 RESTful API）。每个服务都围绕着具体业务进行构建，并且能够被独立的部署到生产环境、类生产环境等。另外，应当尽量避免统一的、集中式的服务管理机制，对具体的一个服务而言，应根据业务上下文，选择合适的语言、工具对其进行构建。

所以总结起来，微服务的核心特点为：小，且专注于做一件事情、轻量级的通信机制、松耦合、独立部署。

1.微服务的优势

微服务之所以能盛行，必然是有它独特优势的，下面我们来分析微服务有哪些方面的优势。

（1）技术异构性

在微服务架构中，每个服务都是一个相对独立的个体，每个服务都可以选择适合于自身的技术来实现。如，要开发一个社交平台，此时，我们可能使用文档型数据库来存储帖 子的内容，使用图数据来存储朋友圈的这些关系等，这样可以把每一块的性能都充分发挥 出来。

同时，在应用新技术时，微服务架构也提供了更好的试验场。因为对于单块的系统而 言，采用一个新的语言、数据库或者框架都会对整个系统产生巨大的影响，这样导致我们 想尝试新技术时，望而却步。但微服务不同，我们完全可以只在一个微服务中采用新技术， 待技术使用熟练之后，再推广到其他服务。

（2）弹性

弹性主要讲的是系统中一部分出现故障会引起多大问题。在单块系统中，一个部分出现问题，可能导致整体系统的问题。而微服务架构中，每个服务可以内置可用性的解决方 案与功能降级方案，所以比单块系统强。

（3）扩展

单块系统中，我们要做扩展，往往是整体进行扩展。而在微服务架构中，可以针对单个服务进行扩展。

（4）简化部署

在大型单块系统中，即使修改一行代码，也需要重新部署整个应用系统。这种部署的影响很大、风险很高，因此不敢轻易地重新部署。而微服务架构中，每个服务的部署都是 独立的，这样就可以更快地对特定部分的代码进行部署。

（5）与结织结构相匹配

我们都知道，团队越大越难管理，同时团队越大也代表系统规模越大代码库越大，这样容易引起一系列的问题。且当团队是分布式的时候，问题更严重。微服务架构就能很好地解决这个问题，微服务架构可以将架构与组织结构相匹配，避免出现过大的代码库，从而获得理想的团队大小及生产力。服务的所有权也可以在团队之 间迁移，从而避免异地团队的出现。

（6）可组合性

在微服务架构中，系统会开放很多接口供外部使用。当情况发生改变时，可以使用不同

的方式构建应用，而整体化应用程序只能提供一个非常粗粒度的接口供外部使用。

（7）对可替代性的优化

在单块系统中如果删除系统中的上百行代码，也许不知道会发生什么，引起什么样的问题，因为单块系统中关联性很强。但在微服务架构中，我们可以在需要时轻易地重写服务，或者删除不再使用的服务。

2. 微服务面临的挑战

软件开发业内有一句名言“软件开发没有银弹”，虽然前面介绍了微服务很多方面的优势，但微服务并不能解决所有问题。下面我们来分析在使用微服务架构时可能面临的一些挑战。

（1）分布式系统的复杂度

使用微服务实现分布式系统的复杂度要比单块系统高。这表现在多个方面，如：性能方面微服务是拆分成多个服务进行部署，服务间的通信都是通过网络，此时的性能会受影响。同时可靠性也会受影响。数据一致性也需要严格控制，其成本也比单块系统高。

（2）运维成本

相比传统的单块架构应用，微服务将系统分成多个独立的部分，每个部分都是可以独立部署的业务单元。这就意味着，原来适用于单块架构的集中式的部署、配置、监控或者日志收集等方式，在微服务架构下，随着服务数量的增多，每个服务都需要独立的配置、部署、监控、日志收集等，因此成本呈指数级增长。

（3）部署自动化

传统单块系统部署往往是以月、周为单位，部署频度很低，在这种情况下，手动部署是可以满足需求的。而对于微服务架构而言，每个服务都是一个独立可部署的业务单元，每个服务的修改都需要独立部署。这样部署的成本是比较高的，如亚马逊，每天都要执行数十次、甚至上百次的部署，此时仍用人工部署是行不通的，要使用自动化部署。如何有效地构建自动化部署流水线，降低部署成本、提高部署频率，是微服务架构下需要面临的一个挑战。

（4）DevOps 与组织结构

传统单块架构中，团队通常是按技能划分，如开发部、测试部、运维部，并通过项目的方式协作，完成系统交付。而在微服务架构的实施过程中，除了如上所述的交付、运维上存在的挑战，在组织或者团队层面，如何传递 DevOps 文化的价值，让团队理解 DevOps 文化的价值，并构建全功能团队，也是一个不小的挑战。

微服务不仅表现出一种架构模型，同样也表现出一种组织模型。这种新型的组织模型意

味着开发人员和运维的角色发生了变化，开发者将承担起服务整个生命周期的责任，包括部署和监控，而运维也越来越多地表现出一种顾问式的角色，尽早考虑服务如何部署。因此，如何在微服务的实施中，按需调整组织架构，构建全功能的团队，是一个不小的挑战。

（5）服务间依赖测试

由于微服务架构是把系统拆分为若干个可独立部署的服务，所以需要进行服务间的依赖测试。在服务数量较多的情况下，如何有效地保证服务之间能有效按照接口的约定正常工作，成为微服务实施过程中必须面临的巨大挑战。

（6）服务间依赖管理

传统的单块系统，功能实现比较集中，大部分功能都运行在同一个应用中，同其他系统依赖较少。而微服务架构则不同，在将系统功能拆分成相互协作的独立服务之后，随着微服务个数的增多，如何清晰有效地展示服务之间的依赖关系，成为了一个挑战。

3.微服务与 SOA

微服务可以讲是 SOA 的一种，但仔细分析与推敲，我们又能发现他们的一些差异。这种差异表现在多个方面，具体如表 9-8 所示。

表 9-8　微服务与 SOA

微服务	SOA
能拆分的就拆分	是整体的，服务能放一起的都放一起
纵向业务划分	是水平分多层
由单一组织负责	按层级划分不同部门的组织负责
细粒度	粗粒度
两句话可以解释明白	几百字只相当于 SOA 的目录
独立的子公司	类似大公司里面划分了一些业务单元（BU）
组件小	存在较复杂的组件
业务逻辑存在于每一个服务中	业务逻辑横跨多个业务领域
使用轻量级的通信方式，如 HTTP	企业服务产总线（ESB）充当了服务之间通信的角色

这些差异自然影响到其实现，在实现方面的主要差异如表 9-9 所示。

表 9-9　微服务与 SOA 实现对比

微服务架构实现	SOA 实现
团队级，自底向上开展实施	企业级，自顶向下开展实施
一个系统被拆分成多个服务，粒度细	服务由多个子系统组成，粒度大
无集中式总线，松散的服务架构	企业服务总线，集中式的服务架构
集成方式简单（HTTP/REST/JSON）	集成方式复杂（ESB/WS/SOAP）
服务能独立部署	单块架构系统，相互依赖，部署复杂

9.6 架构设计

9.2 节讨论了实现软件质量属性的战术，这些战术可以看做设计的基本“构建块”，通过这些构建块，就可以精心设计系统的软件架构了。

架构模式也称为架构风格，它是适当地选取战术的结果，这些固定的结果（模式）在高层抽象层次上具有普遍实用性和复用性。

通过架构模式，架构设计师可以借鉴和复用他人的经验，看看类似的问题别人是如何解决的。但不要把模式看成是一个硬性的解决方法，它只是一种解决问题的思路。Martin Fowler 曾说：“模式和业务构件的区别就在于模式会引发你的思考。”

在本章的后续部分将进一步讨论架构模式。

1．演变交付生命周期

业界已开发出各种软件生命周期模型，其中把架构放在一个适当位置的模型中，典型的有演变交付生命周期模型，如图 9-17 所示。

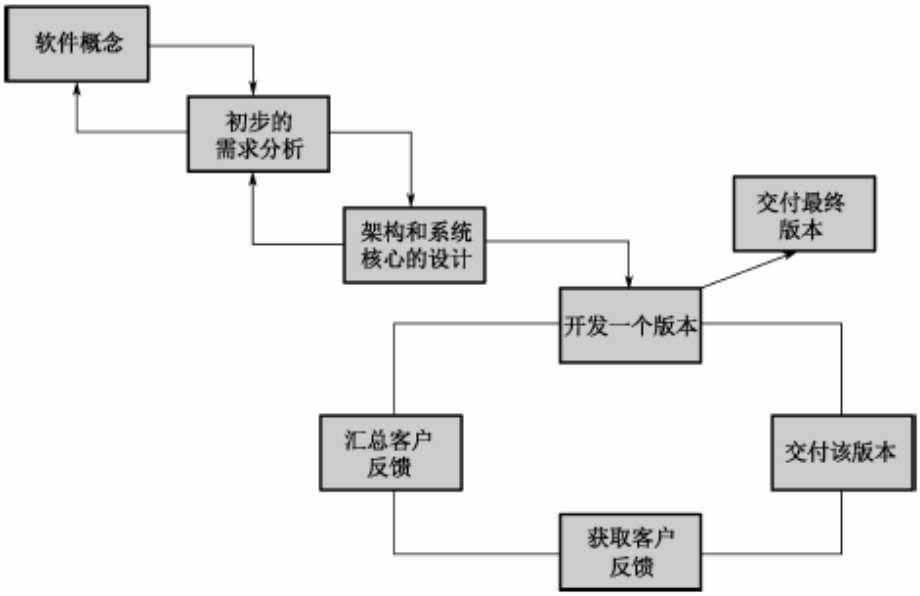


图 9-17　演变交付生命周期模型

在生命周期模型中，架构设计就是从初步的需求分析开始逐步进行循环迭代（图 9-17 中的反向箭头说明了这一点）。即：一方面在了解系统需求前，不能开始设计架构；另一方面，刚开始进行设计架构时并不需要等到全部需求都收集到。架构是由“架构驱动”因素“塑造”的，架构因素是指少数关键的、优先级别最高的业务目标质量需求。架构由少数关键需求决定并在循环迭代中处于基本稳定状态，它作为演变的基础设施。

2．属性驱动设计法

上述模型强调先建立软件架构，再把架构作为骨架，在骨架上循环迭代，逐步长出有血有肉的系统之躯。属性驱动设计法（Attribute-Driven Design，ADD）就是一种定义软件架构的方法，该方法将分解过程建立在软件必须满足的质量属性之上。ADD 的输入为：功能需求（一般表示为用例）、限制条件和质量需求（一组特定于系统的质量场景）。

ADD 的步骤如下。

（1）选择要分解的模块。通常是整个系统，要求上述输入都是可获得的（限制条件、功能需求、质量需求）。从系统开始，然后分解为子系统，进一步将子系统分解为子模块。

（2）根据如下步骤对模块进行求精：

从具体的质量场景和功能需求集合中选择架构驱动因素。并不是同等看待所有需求，而是在满足了最重要的需求的条件下，才满足不太重要的需求，即针对架构需求有优先级。

选择满足架构驱动因素的架构模式，根据前面的战术创建（或选择）模式。其目标是建立一个由模块类型组成的总体架构模式。

实例化模块并根据用例分配功能，使用多个视图进行表示。

定义子模块的接口。

验证用例和质量场景，并对其进行求精，使它们成为子模式的限制。

（3）对需要进一步分解的每个模块重复上述步骤。

3．按架构组织开发团队

在架构的模块分解结构的最初几个层次相对稳定之后，就可以把这些模块分配给开发小组，其结果就是工作视图。像软件系统一样，开发小组也应该努力做到松耦合、高内聚。即每个模块都构成自己的小领域（专门知识或专门技术），并与其他模块的接口清晰，这样，不同的模块分到不同的开发小组中，就能减少各开发小组之间的沟通成本，而在各开发小组内部，由于是处理小领域的问题，容易建立起有效的沟通机制，如成员有这个小领域的背景知识（或培训获得）、共享决策信息。

同时，项目计划在架构确定之后可以结合分工进一步明细化，特别要规划好接口提供的时间点，保证项目开发的整体协调性。

4．开发骨架系统

演变交付生命周期模型中有两个循环，第一个循环是通过迭代的方式开发出软件架构，第二个循环是在架构的基础上通过迭代的方式开发出交付的最终版本。开发骨架系统就是第二个循环的第一步。这一步就是以架构为指导，开发一个可运行的原型（骨架系统）。在骨

架系统开发过程中要注意对接口进行充分协商，避免先开发的部分强制随后部分满足其不合理的接口要求。骨架系统完成后，就可以在其上进行增量开发，直到软件开发完成。

5．利用商用构件进行开发

模式本来就是针对特定问题的解，因此，针对需求的特点，也可以选用相应的模式来设计架构，并利用对应于该模式的商用构件进行软件开发。例如可以使用 J2EE/EJB 进行开发面向对象的分布式系统。

9.7 软件架构文档化

记录软件架构的活动就是架构编档过程，也就是架构的文档化。它包含两个方面：一是过程，编档过程能促使架构设计师进一步思考，使得架构更加完善；二是结果，描述架构的文档将作为架构开发的成果，供项目关系人使用。

1．架构文档的使用者

架构文档的使用者是架构的项目关系人。编写技术文档（尤其是软件架构文档）最基本的原则之一是要从读者的角度来编写，易于编写但很难阅读的文档是不受欢迎的。

架构的主要用途是充当项目关系人之间进行交流的工具，文档则促进了这种交流—— 架构项目关系人希望从架构文档中获得自己所关心的架构信息，如：系统实现人员希望文档提供关于开发活动的不能违反的限制及可利用的自由；测试人员和集成人员希望能从文档中得到必须组合在一起的各部分，并以此得到一个正确的测试黑箱；项目经理希望根据所确定的工作任务组建开发小组，规划和分配项目资源。

2．合理的编档规则编写架构文档和编写其他文档一样，必须遵守一些基本规则，这里将任何软件编档（包括软件架构编档）的规则归纳为 7 条：

（1）从读者的角度编写文档。

（2）避免出现不必要的重复。

（3）避免歧义。

（4）使用标准结构。

（5）记录基本原理。

（6）使文档保持更新，但更新频率不要过高。

（7）针对目标的适宜性对文档进行评审。

3．视图编档

视图是最重要的软件架构编档概念，本书将在 9.11 节中专门讨论架构的视图。视图的概念为架构设计师提供了进行软件架构编档的基本原则。架构文档化就是将相关视图编成文档，并补充多个视图的关联关系。

视图编档的组织结构（内容及编排次序、大纲）虽然目前还没有工业标准模板，研究者（Bachmann 等人）提出的文档组织结构包含 7 个部分，如图 9-18 所示。

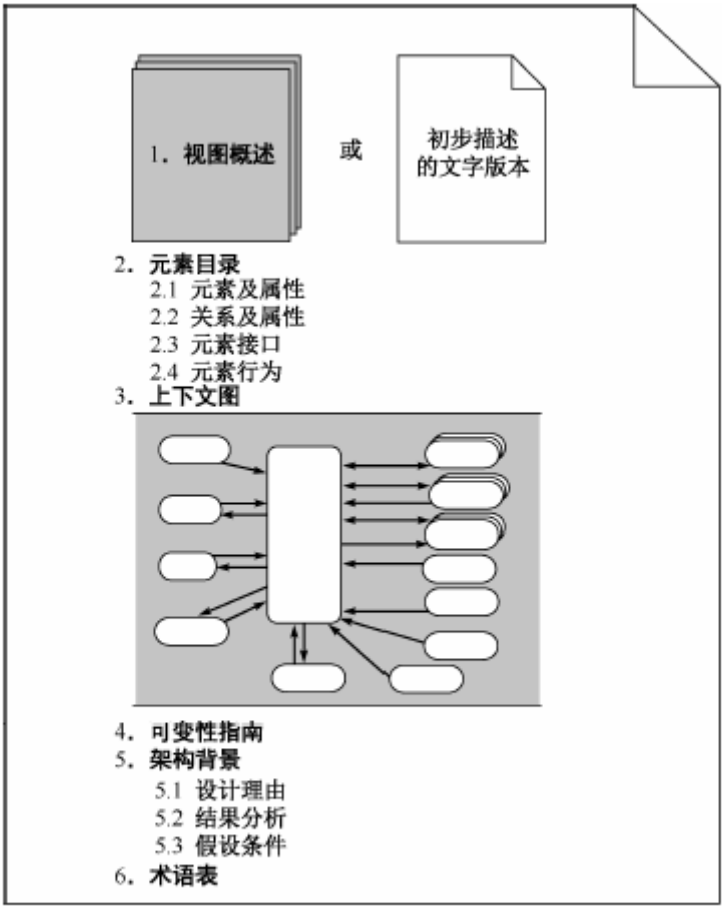


图 9-18　文档组织结构

（1）视图概述：对系统进行概括性的描述，包含视图的主要元素和元素间的关系（但并不包含所有元素和元素间的关

系，如：与错误处理相关的内容可以放在支持文档中）。主要表示可用多个形式：图形、表格、文本，通常用图形形式，使用 UML 语言来描述。

（2）元素目录：对主要表示中所描述的元素及其关系进行详细描述，包括：元素及其 属性、关系及其属性、元素接口、元素行为。

这部分是文档的主要组成部分，其中要注意：

—对元素及其协同工作的行为进行编档，如用 UML 的顺序图和状态图描述行为；

一对接口进行编档，图 9-19 说明了这部分的内容。

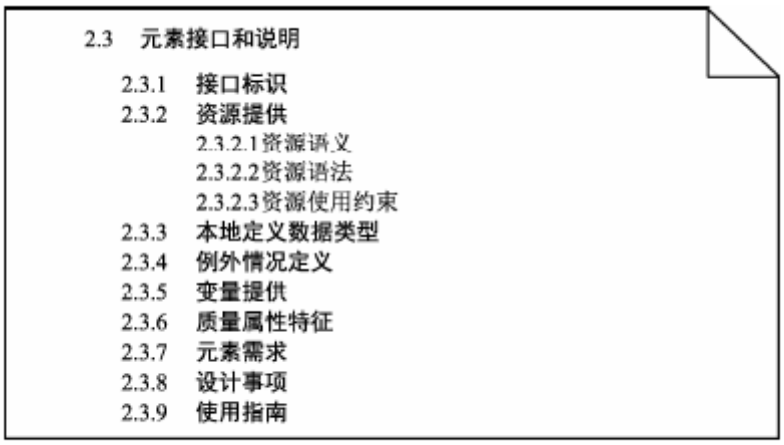


图 9-19 元素接口文档结构

（3）上下文图：用图形展示系统如何与其环境相关。

（4）可变性指南：描述架构的可变化点，如在软件产品线中，产品线架构通过变化，适用于多个系统，因此，文档中应包含这些变化点，如各系统要做出选择的选项、做出选择的时间。

（5）架构背景：为架构的合理性提供足够的、令人信服的论据。包括：基本原理、分析结果及设计中所反映的假定。

（6）术语表：对文档中每个术语进行简要说明。

（7）其他信息：描述不属于架构方面的必要信息，如管理信息（创作者、配置控制数据及变更历史）。

4．跨视图文档

软件架构由多个视图文档来反映，按前面所述的要求完成每个视图的文档后，需要对这些文档进行一个整体的“打包”工作，这就是跨视图文档。它包括如下内容：

（1）文档有哪些内容，它们是如何组织的：视图目录（含哪些视图）；视图模板（即前面描述的视图文档，企业可以通过规范化来定义统一的、公共的视图模板）。

（2）架构概述：它描述系统的目的、视图之间的关联、元素表及索引、项目词汇。

（3）为什么架构是这样的（基本原理）：跨视图基本原理解释了整体架构实际上是其需求的一个解决方案。即解释了做出决策的原因、方案的限制、改变决策时的影响及意义。

5．使用 UML

UML 已经成为对软件架构进行文档化的事实上的标准表示法。在视图文档的组织结构中，UML 主要用于表示元素或元素组的行为。

有关 UML 的详细内容，请阅读 8.4.3 节。

6．软件架构重构

前面已论述了架构编档，即在架构设计时完成编档工作。但是还有另外一种情况：系统已经存在，但不知其架构，即架构没有通过文档很好地保留下来（文档的缺失/失效）。如何维护这样的系统并管理其演变？其关键就是要找到软件架构，软件架构重构就是研究解决这一问题的方法，它是反向工程之一。

架构重构需要工具的支持，但任何一个工具或工具集对架构重构都是不够的。因为：

工具往往是面向特定语言的；

数据提取工具经常返回不完整的或错误的结果，因此，应在多个工具提供的结果间进行补充、验证和判断；

重构的目的（文档的用途）不同，决定了需要提取什么数据，这反过来影响了工具的选择。以此为原则，就是架构重构的工作台方法，如 SEI 开发的 Dali。

软件架构重构由以下活动组成，这些活动以迭代方式进行，如图 9-20 所示。

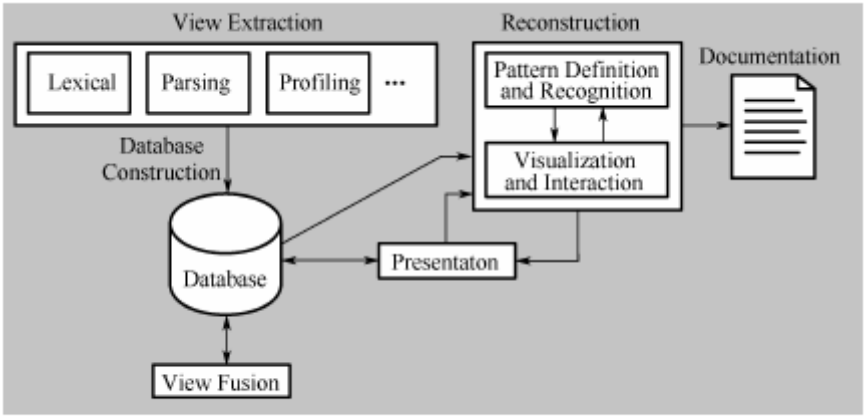


图 9-20　软件架构重构活动

（1）信息提取（View Extraction）。可以使用各种工具进行信息提取，如解析器、语法分析器等；可以利用 build 和 makefile 文件中关于模块的依赖关系；可以从源代码、编译时制品和设计制品中提取静态信息；可以使用分析工具提取动态信息。

（2）数据库构造（Database Construction）：将提取的信息转化为标准的形式，并置于数据库中。

（3）视图融合（View Fusion）：将数据库中的信息组合在一起，生成该架构的一个内聚的视图。

（4）重构（Reconstruction）：构建数据抽象和各种表示以生成架构表示，主要由两个

活动组成：可视化和交互、模式定义和识别。最后生成需要的架构文档（Documentation）。

上述过程中，架构是由重构人员通过对系统做出一组假定来获得，为了最有效地生成这些假定并对其进行验证，必须让熟悉系统的人参与此项工作，包括过去参与系统开发的人员或现在正在对其进行维护的人员。

9.8 软件架构评估

软件架构评估是在对架构分析、评估的基础上，对架构策略的选取进行决策。它也可以灵活地运用于对软件架构进行评审等工作中。

9.8.1 软件架构评估的方法

业界已开发出多种软件架构评估的方法，按基于的技术手段来看，可以分为三类：基于调查问卷或检查表的方式、基于场景的方式和基于度量的方式。

（1）基于调查问卷或检查表的方式：该方式的关键是要设计好问卷或检查表，它充分利用系统相关人员的经验和知识，获得对架构的评估。其缺点是在很大程度上依赖于评估人员的主观推断。

（2）基于场景的方式：基于场景的方式由 SEI 首先提出并应用在架构权衡分析法（Architecture Tradeoff Analysis Method，ATAM）和软件架构分析方法（Software Architecture Analysis Method，SAAM）中。它是通过分析软件架构对场景（也就是对系统的使用或修改活动）的支持程度，从而判断该架构对这一场景所代表的质量需求的满足程度。下节将对 ATAM 进行重点介绍。

（3）基于度量的方式：它是建立在软件架构度量的基础上的，涉及三个基本活动，首先需要建立质量属性和度量之间的映射原则，即确定怎样从度量结果推出系统具有什么样的质量属性；然后从软件架构文档中获取度量信息；最后根据映射原则分析推导出系统的质量属性。它能提供更为客观和量化的质量评估，但它对评估人员及其使用的技术有较高的要求。ATAM 中也使用了度量的思想（度量效用）。

9.8.2 架构的权衡分析法

从技术角度对软件架构进行评估，旨在通过分析来预见软件的质量；通过分析来创建、选择、评估与比较不同的架构。例如，Kazman 等人在 2000 年提出的架构的 ATAM 方法。

ATAM 方法不但能够揭示架构如何满足特定的质量需求（例如，性能和可修改性），而且还提供了分析这些质量需求之间交互作用的方法。使用 ATAM 方法评价一个软件架构的目的是理解架构设计满足系统质量需求的结果。

ATAM 产生如下结果。

（1）一个简洁的架构表述：ATAM 的一个要求是在一小时内表述架构，这样就得到了一个简洁、可理解的、面向普通项目关系人的架构表述。它是从架构文档中提炼形成的。

（2）表述清楚的业务目标。

（3）用场景集合捕获质量需求。

（4）架构决策到质量需求的映射。

（5）所确定的敏感点和权衡点集合：这个集合是一些对一个或多个质量属性具有显著影响的架构决策。如：备份数据库就是这样一个架构决策，它对可靠性产生正面影响，而对系统性能产生负面影响，因此需要进行权衡。

（6）有风险决策和无风险决策。

（7）风险主题的集合。找到这些风险主题旨在采取相应的措施。

（8）产生一些附属结果。评估过程形成的文档（经受住了评估的考验）可以作为经验保留下来。

（9）还产生一些无形结果，如能够使项目关系人产生“团队感”，提供了一个交流平台和沟通渠道，使大家更好地理解架构（优势及弱点）。

ATAM 的 9 个步骤如下。

（1）ATAM 方法的表述：评估负责人向参加会议的项目代表介绍 ATAM（简要描述 ATAM 步骤和评估的结果）。

（2）商业动机的表述。项目决策者从商业角度介绍系统的概况。

（3）架构的表述。对架构进行详略适当的介绍。设计师要描述用来满足需求的架构方法或模式，还应描述技术约束条件及与其他系统的交互等。

（4）对架构方法进行分类。通过研究架构文档及倾听上一步的表述，了解系统使用的架构模式和方法（进行明确命名）。

（5）生成质量属性效用树。可以选取这样一棵树：根——质量属性——属性求精（细分）——场景（叶）。修剪这棵树，保留重要场景（不超过 50 个），再对场景按重要性给定优先级（用 H/M/L 的形式），再按场景实现的难易度来确定优先级（用 H/M/L 的形式），这样对所选定的每个场景就有一个优先级对（重要度，难易度），如（H，L）表示该场景重

要且易实现。

（6）分析架构方法。评估小组按优先级对上述效用树的场景进行分析（小组成员提问，设计师回答、解释），探查实现场景的架构方法。评估小组把相关架构决策编成文档，确定其有风险决策、无风险决策、敏感点、权衡点，并对其进行分类（分别用表格列出）。

（7）集体讨论并确定场景的优先级。由于项目关系人的不同角色及所关心的场景不一致，因此，应鼓励项目关系人考虑效用树中尚未分析过的场景。集体讨论后，可通过投票的方式获得各场景的优先级。通过把集体讨论确定了优先级的一组场景与效用树中的那组场景进行比较，能发现设计师所想的与项目关系人实际所要的是否存在差距，这一差距是否导致风险。

（8）分析架构方法。类似于第 6 步，这时，评估小组引导设计师实现在第 7 步中得到的优先级最高的场景。

（9）结果的表述。把在 ATAM 分析中得到的各种信息进行归纳总结，并呈现给项目关系人。主要有：

已编写了文档的架构方法；

经过讨论得到的场景集合及其优先级；

效用树；

所发现的有风险决策；

已编成文档的无风险决策；

所发现的敏感点和权衡点。

9.8.3 成本效益分析法

在大型复杂系统中最大的权衡通常必须考虑经济性，因此，需要从经济角度建立成本、收益、风险和进度等方面软件的“经济”模型。成本效益分析法（the Cost Benefit Analysis Method，CBAM）是在 ATAM 上构建，用来对架构设计决策的成本和收益进行建模，是优化此类决策的一种手段。CBAM 的思想就是架构策略影响系统的质量属性，反过来这些质量属性又会为系统的项目关系人带来一些收益（称为“效用”），CBAM 协助项目关系人根据其投资回报（ROI）选择架构策略。CBAM 在 ATAM 结束时开始，它实际上使用了 ATAM 评估的结果。

CBAM 的步骤如下。

（1）整理场景。整理 ATAM 中获取的场景，根据商业目标确定这些场景的优先级，并选取优先级最高的 1/3 的场景进行分析。

（2）对场景进行求精。为每个场景获取最坏情况、当前情况、期望情况和最好情况的质量属性响应级别。

（3）确定场景的优先级。项目关系人对场景进行投票，其投票是基于每个场景“所期望的”响应值，根据投票结果和票的权值，生成一个分值（场景的权值）。

（4）分配效用。对场景的响应级别（最坏情况、当前情况、期望情况和最好情况）确定效用表。

（5）架构策略涉及哪些质量属性及响应级别，形成相关的策略—场景—响应级别的对应关系。

（6）使用内插法确定“期望的”质量属性响应级别的效用。即根据第 4 步的效用表以及第 5 步的对应关系，确定架构策略及其对应场景的效用表。

（7）计算各架构策略的总收益。根据第 3 步的场景的权值及第 6 步的架构策略效用表，计算出架构策略的总收益得分。

（8）根据受成本限制影响的 ROI（Return On Investment，投资报酬率）选择架构策略。根据开发经验估算架构策略的成本，结合第 7 步的收益，计算出架构策略的 ROI，按 ROI 排序，从而确定选取策略的优先级。

9.9 构件及其复用

软件企业为了提高开发效率，越来越注重软件元素的复用（也称重用），因此，架构设计师在进行架构设计时，必须关注复用，例如，考虑丰富企业构件和充分使用已有的构件。本节从构件角度研究如何使用软件复用技术，下一节重点讨论基于产品线的软件复用。

与复用技术密切相关的概念是构件（component，组件），业界对构件还没有公认的定义，如下为几种常见的定义。

定义 1：构件是指软件系统中可以明确辨识的构成成分。而可复用构件（reusable component）是指具有相对独立的功能和可复用价值的构件。

定义 2：构件是一个组装单元，它具有约定式规范的接口及明确的依赖环境。

定义 3：构件是软件系统中具有相对独立功能、可以明确辨识、接口由契约指定、和语境有明显依赖关系、可独立部署的可组装软件实体。

对构件更广义的理解是把所有种类的工作成品（例如，各类文档、方案、计划、测试案例、代码）都看成是可复用的构件。

9.9.1 商用构件标准规范

当前，主流的商用构件标准规范包括 OMG（Object Management Group，对象管理组织）的 CORBA、SUN 的 J2EE 和 Microsoft 的 DNA。

1．CORBA

CORBA（Common ObjectRequest Broker Architecture，公共对象请求代理架构）主要分为 3 个层次：对象请求代理、公共对象服务和公共设施。最底层的对象请求代理（Object Request Broker，ORB），规定了分布对象的定义（接口）和语言映射，实现对象间的通信和互操作，是分布对象系统中的“软总线”；在 ORB 之上定义了很多公共服务，可以提供诸如并发服务、名字服务、事务（交易）服务、安全服务等各种各样的服务；最上层的公共设施则定义了构件框架，提供可直接为业务对象使用的服务，规定业务对象有效协作所需的协定规则。

CORBA CCM（CORBA ComponentModel，CORBA 构件模型）是 OMG 组织制定的一个用于开发和配置分布式应用的服务器端构件模型规范，它主要包括如下 3 项内容。

（1）抽象构件模型：用以描述服务器端构件结构及构件间互操作的结构。

（2）构件容器结构：用以提供通用的构件运行和管理环境，并支持对安全、事务、持久状态等系统服务的集成。

（3）构件的配置和打包规范：CCM 使用打包技术来管理构件的二进制、多语言版本的可执行代码和配置信息，并制定了构件包的具体内容和文档内容标准。

2.J2EE

在 J2EE 中，SUN 给出了完整的基于 Java 语言开发面向企业分布的应用规范，其中，在分布式互操作协议上，J2EE 同时支持 RMI（Remote Method Invocation，远程方法调用）和 IIOP（Internet Inter-ORB Protocol，互联网内部对象请求代理协议），而在服务器端分布式应用的构造形式，则包括了 Java Servlet、JSP、EJB 等多种形式，以支持不同的业务需求，而且 Java 应用程序具有跨平台的特性，使得 J2EE 技术在发布计算领域得到了快速发展。其中，EJB 给出了系统的服务器端分布构件规范，这包括了构件、构件容器的接口规范，以及构件打包、构件配置等的标准规范内容。EJB 技术的推出，使得用 Java 基于构件方法开发服务器端分布式应用成为可能。从企业应用多层结构的角度，EJB 是业务逻辑层的中间件

技术，与 JavaBeans 不同，它提供了事务处理的能力，自从三层结构提出以后，中间层，也就是业务逻辑层，是处理事务的核心，从数据存储层分离，取代了存储层的大部分地位。从 Internet 技术应用的角度，EJB 和 Servlet，JSP 一起成为新一代应用服务器的技术标准，EJB 中的 Bean 可以分为会话 Bean 和实体 Bean，前者维护会话，后者处理事务，通常由 Servlet 负责与客户端通信，访问 EJB，并把结果通过 JSP 产生页面传回客户端。

3．DNA 2000

Microsoft DNA 2000 是 Microsoft 在推出 Windows 2000 系列操作系统平台的基础上，在扩展了分布计算模型，以及改造 Back Office 系列服务器端分布计算产品后发布的新的分布计算架构和规范。在服务器端，DNA 2000 提供了 ASP、COM、Cluster 等的应用支持。

DNA 2000 融合了当今最先进的分布计算理论和思想，例如，事务处理、可伸缩性、异步消息队列、集群等内容。DNA 可以开发基于 Microsoft 平台的服务器构件应用，其中，如数据库事务服务、异步通信服务和安全服务等，都由底层的分布对象系统提供。

Microsoft 的 DCOM/COM/COM+技术，在 DNA 2000 分布计算结构基础上，展现了一个全新的分布构件应用模型。首先，DCOM/COM/COM+的构件仍然采用普通的 COM（Component Object Model，构件对象模型）模型。COM 最初作为 Microsoft 桌面系统的构件技术，主要为本地的 OLE（Object Linking and Embedding，对象连接与嵌入）应用服务，但是随着 Microsoft 服务器操作系统 Windows NT 和 DCOM（Distributed Component Object Model，分布式构件对象模型）的发布，COM 通过底层的远程支持使得构件技术延伸到了分布应用领域。DCOM/COM/COM+更将其扩充为面向服务器端分布应用的业务逻辑中间件。通过 COM+的相关服务设施，如负载均衡、内存数据库、对象池、构件管理与配置等，DCOM/COM/COM+将 COM、DCOM、MTS（Microsoft Transaction Server，微软事物处理服务器）的功能有机地统一在一起，形成了一个概念、功能强的构件应用架构。

通过购买商用构件（平台）并遵循其开发标准来进行应用开发，是提高应用软件开发效率的常见选择

9.9.2 应用系统簇与构件系统

除专门开发构件的企业外，开发应用系统的企业也会发展自己的构件应用体系：通常是随着企业的不断成熟，逐步从已开发的应用系统中整理出来一些构件，反过来，将这些构件复用到优化与整合已有应用系统中或复用于开发新的应用系统。

一个应用系统中的复用率毕竟有限，通常在应用系统簇中进行软件构件复用。当要开发若干相关的应用系统时，可以先按复用的要求，界定这一组应用系统的共同“特性”，根据这些共同特性，建立模型，并按照复用的要求，将模型分解成恰当规模和结构的构件，对这些构件进行设计、实现、打包、编写文档，形成方便使用的可复用构件。这批可复用构件将用于支持该应用簇的各个应用系统的开发工作。这里的构件特指一个封装的代码模块或大粒度的运行模块。

软件企业将相关的构件有机地组织在一起，形成构件系统（较构件库层次更高），实施复用的软件企业通常拥有多个构件系统，有的是购置的，有的是自己开发的。

应用系统和构件系统都是系统产品（而不是工作产品）。它们都可以采用模型和结构的类型定义出来。一般情况下，构件系统只在开发单位内部使用，而应用系统提供给外部客户，与应用系统相比，构件系统具有通用性，可复用性，这就要求构件系统的开发过程应当实施更为严格的工程规范。

一个构件系统是能提供一系列可复用特性的系统产品。构件系统中的构件应当是高内聚、低耦合的，但构件之间应有若干种关系；可以发送消息给其他构件；可以与其他构件联合，支持协同工作。构件系统应当是易于理解和易于使用的，对构件应当是仔细地进行建模、实现、制作文档、测试等，便于以后的有效维护和改进。通常为支持构件的复用，应开发与构件系统相配的工具箱。

应用系统可以向构件系统输入构件（构件的需求源于应用系统或应用系统中的模块），反过来，构件系统向应用系统输出构件。这就是构件系统如何获得构件和如何提供构件的方式。

9.9.3 基于复用开发的组织结构

基于复用的开发组织与传统的开发组织结构不同，它需要有一部分用于开发可复用资产的资源，这部分资源应同具体应用系统的开发资源分开，以确保不被占用。

一种较平衡的组织结构如图 9-21 所示，它有三类职能部门：一是构件系统开发部门，它开发可复用资产；二是应用系统项目开发部（多个），它复用资产；三是支持部门，这个部门是可选的，它进一步隔离上述两主体部门，虽然牺牲了一些效率，但保证了构件的规范性。它的主要职责是对构件开发部门所提供的可复用资产进行确认、对构件库进行分类编目、向开发应用系统的工程师们发通告和分发可复用资产、提供必要的文档、从复用者处收集反馈信息和缺陷报告。可以这样理解：构件系统开发部门开发的构件系统由支持部门向应用开

发部门推广，即支持部门是企业内的推广部。外部购买的商用构件也由支持部门维护与管理。

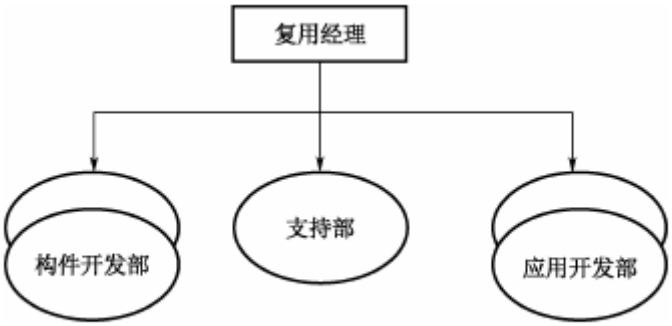


图 9-21　一种软件复用组织结构

这三个平行部门之上有一个高层经理（复用经理），他关注总目标，协调相互关系。

一方面，构件开发者应当尽量接近应用开发者，以使其开发出的构件能尽量符合实际需要；另一方面，构件开发者与应用开发者分属两个并列的部门，使构件开发者能摆脱应用项目的日常压力，保证可复用资产的开发和持续改进。复用经理应当在构件开发和应用项目开发利益之间进行权衡，保证长期目标不受近期项目压力的影响。

9.10　产品线及系统演化

软件企业追求长远的发展，通常采用产品线模型及系统演化策略，它实质上是用架构技术构建产品线，并在此基础上借助复用技术持续演化，不断地推出新产品，满足市场追求产品升级换代的需求。

9.10.1 复用与产品线

软件产品线是指一组软件密集型系统，它们共享一个公共的、可管理的特性集，满足某个特定市场或任务的具体需要，是以规定的方式用公共的核心资产集成开发出来的。即围绕核心资产库进行管理、复用、集成新的系统。核心资产库包括软件架构及其可剪裁的元素，更广泛地，它还包括设计方案及其文档、用户手册、项目管理的历史记录（如预算和进度）、软件测试计划和测试用例。复用核心资产（特别是软件架构），更进一步采用产品线将会惊人地提高生产效率、降低生产成本和缩短上市时间。

创建一个成功的产品线取决于软件工程、技术管理和组织管理等多个方面的协调，这里只对软件架构方面进行讨论。

基于产品间共性的“软件”产品线代表了软件工程中一个创新的、不断发展的概念。软

件产品线的本质是在生产产品家族时，以一种规范的、策略性的方法复用资产。可复用的资产非常广，包括以下几点。

需求：许多需求与早期开发的系统相同或部分相同，如网上银行交易与银行柜面交易。

架构设计：原系统在架构设计方面花费了大量的时间与精力，系统成功验证了架构的合理性，如果新产品能复用已有的架构，将会取得很好的效益。

元素：元素复用不只是简单的代码复用，它旨在捕获并复用设计中的可取之处，避免（不要重复）设计失败的地方。

建模与分析：各类分析方法（如性能分析）及各类方案模型（如容错方案、负载均衡方案）都可以在产品中得到复用。

测试：采用产品线可积累大量的测试资源，即在考虑测试时不是以项目为单位，而是以产品线为单位，这样，整个测试环境都可以得到复用，如测试用例、测试数据、测试工具，甚至测试计划、过程、沟通渠道都可以得到复用。

项目规划：利用经验对项目的成本、预算、进度及开发小组的安排等进行预测，即不必每次都建立工作分解结构。

过程、方法和工具：有了产品线这面旗帜，企业就可以建立产品线级的工作流程、规范、标准、方法和工具环境，供产品线中所有产品复用。如编码标准就是一例。

人员：以产品线来培训的人员，适应于整个系列的各个产品的开发。

样本系统：将已部署（投产）的产品作为高质量的演示原型和工程设计原型。

缺陷消除：产品线开发中积累的缺陷消除活动，可使新系统受益，特别是整个产品家族中的性能、可靠性等问题的一次性解决，能取得很高的回报。同时也使得开发人员和客户心中有“底”。

9.10.2 基于产品线的架构

软件产品线架构是针对一系列产品而设计的通用架构，并在此基础上，进一步将系列产品共用的模块事先实现，供直接重用；将架构用框架的形式予以实现，供定制使用。这就是通常所说的“平台”。

产品线架构较之单个产品架构，有如下三点特别之处：

（1）产品线架构必须考虑一系列明确许可的变化；

（2）产品线架构一定要文档化；

（3）产品线架构必须提供“产品创建者指南”（开发指南），描述架构的实例化过程。

产品线的软件架构应将不变的方面提出来（正因为有不变，才产生了产品线），同时，识别允许的变化，并提供实现它们的机制。通常应考虑三个方面。

（1）确定变化点：确定变化是一项需要持续进行的活动，可以在开发过程的任何时间确定变化。

（2）支持变化点：对变化的支持可以有多种形式，举例如下。

包含或省略元素：如条件编译。

包含不同数量的复制元素：如通过添加更多的服务器产生高容量的变体。

具有相同的接口但具有不同的行为或质量特性的元素版本的选择：如静态库和动态链接库的使用。

在面向对象的系统中，通过特化或泛化特定的类实现变化。

通过参数配置来实现变化。

（3）对产品线架构的适宜性进行评估。

评审方法：见下节的软件架构评估。

评估的内容：必须把评估的重点放在变化点上，以确保它（变化点）提供了足够的灵活性，从而能覆盖产品线的预期范围；它们支持快速构建产品。

评估的时间：应该对用于构建产品线中的一个或多个产品的架构的实例或变体进行评估。产品架构评估的结果通常能提供有用的反馈，推动架构的改进。当提议开发的一个新产品不在最初的产品线的范围内时，则可以重新对产品线架构进行评估，看如何使其容纳新的产品或是否有必要产生一个新的产品线。

9.10.3 产品线的开发模型

开发（确定）产品线的方法有两种模型：

（1）“前瞻性”产品线：利用在应用领域的经验、对市场和技术发展趋势的了解及商业判断力等进行产品线设计，它反映了企业的战略决策。通常是自上而下地采用产品线方法。

（2）“反应性”模型：企业根据以前的产品构建产品家族，并随着新产品的开发，扩展架构和设计方案，它的核心资产库是根据“已经证明”为共有、而非“预先计划”为共有的元素构建的。通常是自下而上地采用产品线方法。

一旦确定了产品线，就进入其演变阶段，它是产品线不断向前的发展过程。

9.10.4 特定领域软件架构

架构的本质在于其抽象性。它包括两个方面的抽象：业务抽象和技术抽象。其中业务抽象面向特定的应用领域。

特定领域软件架构（Domain Specific Software Architecture，DSSA）可以看做开发产品线的一个方法（或理论），它的目标就是支持在一个特定领域中有多个应用的生成。DSSA 的必备特征有：

（1）一个严格定义的问题域或解决域；

（2）具有普遍性，使其可以用于领域中某个特定应用的开发；

（3）对整个领域的合适程度的抽象；

（4）具备该领域固定的、典型的在开发过程中的可复用元素。

从功能覆盖的范围角度理解 DSSA 中领域的含义有两种方法：

（1）垂直域。定义了一个特定的系统族，导出在该领域中可作为系统的可行解决方案的一个通用软件架构。

（2）水平域。定义了在多个系统和多个系统族中功能区域的共有部分，在子系统级上涵盖多个系统（族）的特定部分功能。

DSSA 的活动阶段如下。

（1）领域分析：主要目标是获得领域模型。即通过分析领域中系统的需求（领域需求），确定哪些需求是被领域中的系统广泛共享的，从而建立领域模型。

（2）领域设计：这个阶段的目标是获得 DSSA，它是一个能够适应领域多个系统的需求的一个高层次的设计。由于领域模型中的领域需求具有一定的变化性，DSSA 也要相应地具有变化性，它可以通过表示多选一的、可选的解决方案等来做到这一点。

（3）领域实现：主要目标是依据领域模型和 DSSA 开发与组织可复用信息。这些复用信息可以是从现有系统中提取得到的，也可能通过新的开发得到。这个阶段可以看作复用基础设施的实现阶段。

在上述工作中，获得领域模型是基础也是关键，领域建模专注于分析问题领域本身，发掘重要的业务领域概念，并建立业务领域概念之间的关系。通常领域模型可用 UML 的类图和状态图表示。

希赛教育专家提示：对于中等复杂度的项目，应该在系统的领域模型中找到大约 50 到 100 个类。

领域模型的主要作用如下：

（1）领域模型为需求定义了领域知识和领域词汇，这较之单一的项目需求更有较好的大局观；

（2）软件界面的设计往往和领域模型关系密切；

（3）领域模型的合理性将严重影响软件系统的可扩展性；

（4）在分层架构的指导下，领域模型精化后即成为业务层的骨架；

（5）领域模型也是其数据模型的基础；

（6）领域模型是团队交流的基础，因为它规定了重要的领域词汇表，并且这些词汇的定义是严格的、大家共同认可的。

9.10.5 架构及系统演化

架构虽然为系统的变化提供了一定的自由度，但是系统的较大变化必然导致架构的改变。架构（系统）演化是指向既定的方向、可控地改变。架构（系统）演化可以形成产品线，反过来，架构（系统）可以在规划的产品线中进行演化。

架构（系统）演化过程包含 7 个步骤，如图 9-22 所示。

（1）需求变动归类。首先，必须对用户需求的变化进行归类，使变化的需求与已有构件对应。对找不到对应构件的变动，也要做好标记，在后续工作中，将创建新的构件，以对应这部分变化的需求。

（2）制订架构演化计划。在改变原有结构之前，开发组织必须制订一个周密的架构演化计划，作为后续演化开发工作的指南。

（3）修改、增加或删除构件。在演化计划的基础上，开发人员可根据在第（1）步得到的需求变动的归类情况，决定是否修改或删除存在的构件、增加新构件。最后，对修改和增加的构件进行功能性测试。

（4）更新构件的相互作用。随着构件的增加、删除和修改，构件之间的控制流必须得到更新。

（5）构件组装与测试。通过组装支持工具把这些构件的实现体组装起来，完成整个软件系统的连接与合成，形成新的架构。然后，对组装后的系统整体功能和性能进行测试。

（6）技术评审。对以上步骤进行确认，进行技术评审。评审组装后的架构是否反映需求变动，符合用户需求。如果不符合，则需要在第（2）到第（6）步之间进行迭代。

（7）产生演化后的架构。在原来系统上所作的所有修改必须集成到原来的架构中，完成一次演化过程。

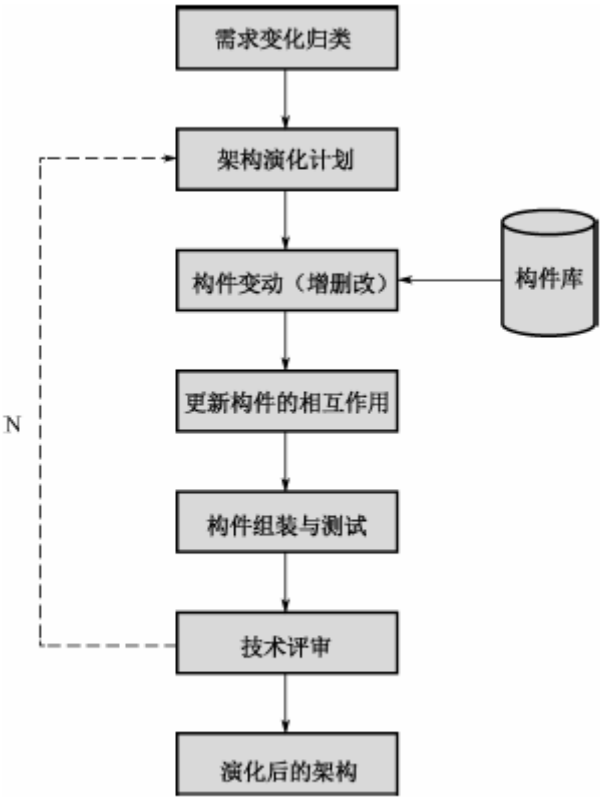


图 9-22　架构演化过程

9.11 软件架构视图

9.10 节从软件架构本身的特点出发讨论了架构建模及与特定应用领域密切相关的架构风格。本节将从对架构编档的角度对软件架构视图及其风格进行讨论。

9.11.1 软件视图的分类

现代软件系统非常复杂，通常在某个具体的时间内只需将注意力集中在某几个结构上（就像看病时，医生只是将注意力集中在某方面的人体结构上，骨科医生与心血管科医生关心不同的结构），结构是元素本身的集合，而视图则是捕获和表达结构（文档描述），虽然它们有区别，但在实际使用时则不严格区分，即从系统体系的角度说是结构，从文档角度说是视图，因此，本节将不再区分结构和视图术语。

软件架构是一种无法以简单的一维方式进行说明的复杂实体，从不同侧面的描述就是视

图。架构的优势也在于使用视图：每个视图强调系统的某一个方面，同时忽视系统的其他方面，以便有助于处理或理解当前问题，描述完整的系统架构必须具备完整的视图集，“4+1”方法就是一类完备视图集。

软件视图通常分为三种类型：

（1）模块视图类型：为系统的主要模块实现单元编档。

（2）构件和连接件视图类型：为系统的构件和连接件执行单元编档。

（3）分配视图类型：为软件的开发和执行环境之间的关系编档。每一视图类型中，又有一些常用的形态，可以把这些形态归纳成架构风格（简称风格），

大量的架构风格供架构设计师选用，例如客户机/服务器是一种常见的架构风格，它是构件和连接件视图类型中的一员。架构风格是对元素和关系类型的特化，它还包括如何使用这些元素和关系类型的一组限制条件。架构结构/视图分类如表 9-10 所示。下面各小节中再分别对这三种类型及其风格从元素、关系及特征方面做进一步总结。

表 9-10　系统的架构视图

组　别	架构风格	说　明	应用于
模块 视图类型	分解	大模块分解为小模块，小到容易理解	资源分配、项目结构化和规划；信息隐蔽、封装；配置控制
	使用	一个单元的正确性依赖于另一个单元的正确性（如版本）	设计子集；设计扩展（增量开发）
	分层	上层使用下层的服务；实现隐藏细节的抽象	增量式开发；基于“虚拟机”上的可移植性
	类或泛化	“继承自”或“是一个实例”；共享访问方法	面向对象的设计（使用公共模板）
构件-连接器 视图类型	客户机-服务器	构件是客户机和服务器，连接件是协议及共享消息	分布式操作；关注点分离（支持可修改性）；负载均衡
	进程或通信进程	通过通信、同步或排除操作形成进程或线程之间的关联	调度分析；性能分析
	并发	在相同的“逻辑线程”上运行	确定资源争用；分析线程
	共享数据	运行时产生数据、使用数据（共享数据存储库）	性能；数据完整理；可修改性
分配 视图类型	部署	软件功能分配给软件（进程）、硬件（处理器）和通信路径	性能、可用性、安全性说明。尤其在分布式或并行系统中
	实现	模块映射到开发活动中	配置控制、集成、测试活动
	工作分配	将责任分配到适当的开发小组，特别是公共部分不是每个人去实现	项目管理、管理通用性，最好的专业技术安排

9.11.2 模块视图类型及其风格

模块将遵循某种方式将软件系统分解成可管理的功能单元。架构模块视图是通过文档来

枚举系统的主要实现单元或模块，及这些单元之间的关系。

任务完整的架构文档必须包含有模块视图，它为源代码提供蓝图。该类型如表 9-11 所示。

表 9-11　模块视图类型总结

元素	其元素就是模块，它是一种能提供内聚功能单元的软件实现单元
关系	其关系表现为： • “部分关系”，即模块间部分-整体关系，如分解关系； • “依赖关系”，如“共享数据”、“调用”； • “特化关系”，定义了较为特殊的模块和一般模块之间的关系，如面向对象中的继承
元素特征	• 可能必须遵守命名空间规则的名称； • 模块责任，应该使用责任特性定义模块功能； • 实现信息
关系特征	• “部分关系”拥有相关的可见性特征，这种特性确定子模块在聚集模块之外是否可见； • “依赖关系”拥有分配的约束条件，以便详细规定两个模块之间的依赖性关系； • “特化关系”拥有实现特性，如特殊模块继承一般模块的实现方案

下面对模块视图的四种风格进行总结。

（1）分解风格能展示向模块分配责任的方式。该风格总结如表 9-12 所示。

表 9-12　模块分解风格总结

元素	模块，见表 9-11。有时将具有独立完整功能的模块称为子系统
关系	分解关系，它是“部分关系”的精化形式，文档必须定义分解的标准
元素特征	见表 9-11
关系特征	可见性，模块被其父模块之外的模块了解的程度及其功能对于这些外部模块的可用程度
其他	• 分解视图中不允许出现循环； • 在一个视图中，一个模块不能同时属于多个模块

（2）使用风格能展示模块相互依赖的方式。该风格总结如表 9-13 所示。

表 9-13　模块使用风格总结

元素	模块，见表 9-11
关系	使用关系，它是“依赖关系”的精化形式，如果模块 A 依赖于功能正常的模块 B 的存在来满足自己的需求，那么，模块 A 就是在使用模块 B
元素特征	见表 9-11
关系特征	描述一个模块会以哪种方式使用另一个模块

（3）分层风格能将系统分割成一组虚拟机，通过“允许使用”关系相互关联，分层风格能帮助实现可移植性和可修改性。该风格总结如表 9-14 所示。

表 9-14　模块分层风格总结

元素	层
关系	“允许使用”，它是模块视图类型一般的“依赖关系”的特化。如果 P1 的正确性依赖于当前 P2 的正确实现，就说 P1 使用 P2
元素特征	层的名称。 层包容的软件单元。 允许层使用的软件。一是层间和层内的使用规则，如“高层可以使用较低层次的软件”和“不允许软件使用同一层中的其他软件”等；二是这些规则可以容许的例外情况。 层的内聚：对层所表示的虚拟机的描述
关系特征	见表 9-11
其他	每一部分软件只能分配给一个层

（4）泛化风格能展示一个模块如何成为另一个模块的泛化或特化，从而使模块之间产生关联。它广泛应用于面向对象的系统，能展示继承性，并能用来使用模块之间的共性。该风格总结如表 9-15 所示。

表 9-15　模块泛化风格总结

元素	模块，见表 9-11
关系	泛化关系，即模块视图类型中的“特化关系”
元素特征	除了模块视图类型中为模块定义的特性外，模块还能拥有“抽象”特性，抽象特性能定义拥有接口但没有实现方案的模块
关系特征	泛化关系能拥有一种区别接口和实现继承的特性
其他	• 不允许出现循环，子模块不能是父模块的泛化； • 模块能拥有多个父模块，但这种多重继承一般不提倡

9.11.3 C&C 视图类型及其风格

C&C 视图能定义由具有某种运行时存在的元素模型，这些元素包括进程、对象、客户机、服务器及数据存储器等。此外，它还包含作为元素的交互路径，如通信链路和协议、信息流及共享存储器访问。通常，可利用复杂的基础结构（如中间件框架、分布式通信信道和进程调度）来执行这些交互操作。该类型总结如表 9-16 所示。

表 9-16　C&C 视图类型总结

元素	• 构件类型：主要处理单元和数据存储器； • 连接件类型：交互机制
关系	构件且有接口，这种接口被称为端口。连接件具有接口，这种接口被称为角色。 连接：构件端口与特定的连接件角色相关联
元素特征	（1）构件。 • 名称：应反映构件功能； • 类型：定义一般功能、端口数量及类型以及所需特征； • 其他：包括性能和可靠性值等（取决于构件类型）。

元素特征	（2）连接件。 • 名称：应反映连接件的交互功能； • 类型：定义交互性质、角色数量、类型及所需特征； • 其他：包括交互协议和性能值等（取决于连接件类型）

C&C 视图风格是 C&C 视图类型的特化，C&C 视图风格为数不少，下面对 C&C 视图的几种风格进行总结。

（1）管道和过滤器风格中的交互模式表现出数据流连续变换的特征。数据抵达过滤器并经过转换后由管理传送给下一个过滤器。该风格总结如表 9-17 所示。

表 9-17　管道和过滤器风格总结

元素	• 构件——过滤器。过滤器端口必须是输入端口或输出端口； • 连接件——管道。管道扮演数据输入和数据输出角色
关系	连接关系能使过滤器输出端口与某个管道的数据输入角色相关联，使过滤器输入端口与多个管道的数据输出角色相关联，并能确定交互过滤器的图形
计算模型	• 过滤器是从其输入端口读取数据并将数据流写入其输出端口的数据转换器； • 管道能将数据流从一个过滤器传送到另一个过滤器
特征	见表 8-16

（2）共享数据风格通过保留持久数据来支配交互模式，持久数据由多个数据存取器和至少一个储存库保留。该风格总结如表 9-18 所示。

表 9-18　共享数据风格总结

元素	• 构件——共享数据储存库和数据存取器； • 连接件——数据读写
关系	连接关系能确定哪些数据存取器将连接到哪些数据储存库
计算模型	数据存取器之间的通信经由共享数据存储库来完成，控制过程由数据存取器或数据储存库来启动
特征	见表 9-16。可精化为：存储数据的类型、面向性能的数据特征和数据分配

（3）发布-订阅风格用于向一组未知接受者发送事件和消息。可在不修改生产者的情况下添加新的接受者（订阅者）。在发布-订阅风格中，构件通过事件发布进行交互。构件可订阅一组事件。该风格总结如表 9-19 所示。

表 9-19　发布-订阅风格总结

元素	• 构件——任何具有能发布和订阅事件的接口的 C&C 构件； • 连接件——发布—订阅
关系	连接关系能将构件与发布-订阅连接件关联起来
计算模型	宣布事件并能对其他已宣布事件做出反应的独立构件系统
特征	见表 9-16。可精化为：哪些事件由哪些构件宣布，哪些事件由哪些构件订阅，什么时候允许构件订阅事件
其他	所有构件连接到一个事件分配器，可将该分配器视为总线（连接件）或构件

（4）客户机-服务器风格能展示构件通过请求其他构件的服务进行交互的过程，将功能划分成客户机和服务器后即可基于运行时准则把它们单独分配给各个级。该风格总结如表 9-20 所示。

点击查看大图

（5）对等连接系统能通过构件之间的直接交换支持服务交换。它是一种调用/返回风格。该风格总结如表 9-21 所示。

表 9-20　客户机-服务器风格总结

元素	• 构件——请求其他构件服务的客户机和向其他构件提供服务的服务器； • 连接件——请求/应答，即客户机对服务器的非对称调用
关系	连接关系使客户机与连接件的请求角色相关联，使服务器与连接件的应答角色相关联，并确定哪些服务能由哪些客户机请求
计算模型	客户机能启动各项活动，向服务器请求所需服务，并等待这些请求的结果
特征	见表 9-16。可精化为：可连接的客户机数量和类型及性能特性
其他	可施加以下限制： • 与给定端口或角色的连接数量； • 服务器之间允许存在的关系； • 是层级的

（6）通信-进程风格的特征表现在通过各种连接件机制并发执行构件的交互，如通过同步、消息传递、数据交换、启动和停止等进行交互。该风格总结如表 9-22 所示。

表 9-22　通信-进程风格总结

元素	构件——并发单元，如任务、进程和线程； 连接件——数据交换、消息传递、同步、控制和其他类型的通信
关系	连接关系，见表 9-16
计算模型	通过特定连接件机制进行交互的并发执行构件
特征	并发单元：“可抢占性”，它表示并发单元的执行可被另一个并发单元抢占，或并发单元将继续执行，直到它自愿中止自己的执行；“优先性”，它能影响调度；“时间参数”定义周期和最后期限等。 数据交换：“缓冲”，它表示如果不能立即处理消息就会先把消息保存起来；“协议”用于通信

9.11.4　分配视图类型及其风格

硬件、文件系统和团队结构都会与软件架构进行交互，将软件架构映射到其环境的一般形式称为“分配视图类型”。该类型总结如表 9-23 所示。

表 9-23　分配视图类型总结

元素	软件元素和环境元素
关系	“分配到……”。软件元素被分配到环境元素
元素特征	软件元素拥有“要求的”特征。环境元素拥有“提供的”特性，前者必须与后者匹配
关系特征	取决于特定的风格

分配视图类型的三种常见风格为：

部置风格：能描述构件和连接件对硬件的映射，硬件是软件执行的场所。

实现风格：能描述模块对包含它们的文件系统的映射。

工作任务风格：能描述模块对承担模块开发任务的人员、团队或小组的映射。

（1）部置风格体现为 C&C 风格（如通信-进程风格）的元素被分配到执行平台。该风格总结如表 9-24 所示。

表 9-24　部置风格总结

元素	软件元素：通常是 C&C 视图类型中的进程； 环境元素：计算硬件，如处理器、内存、磁盘和网络等
关系	“分配到……”。表示软件元素驻留在哪些物理单元上。 分配可以是动态的
元素特征	软件元素所要求的特性：重要的硬件特征，如处理器、内存、容量需求和容错性； 环境元素提供的特性：影响分配决策的重要硬件特征
关系特征	“分配到……”

（2）实现风格能将模块视图类型中的模块映射到开发基础结构。实现一个模块总会产生许多独立文件，必须对这些文件进行组织，以免失去对系统的控制及系统的完整性。通常利用配置管理技术进行文件管理。该风格总结如表 9-25 所示。

表 9-25　实现风格总结

元素	软件元素：模块； 环境元素：配置条目，如文件或目录
关系	包含关系，规定一个配置条目由另一个配置条目包容； “分配到……”关系，描述将模块分配到配置条目
元素特征	如果有的话，就是软件元素所要求的特性：例如对数据库等开发环境的需求； 环境元素提供的特性：对开发环境提供的特征的指示
关系特征	无
其他	分层配置条目：“包容在……”

（3）软件项目的时间和预算估计取决于工作分解结构（WBS），而工作分解结构则取决于软件架构。工作任务风格将软件架构映射到由人组成的团队之中，实现这一项目管理的目的。该风格总结如表 9-26 所示。

表 9-26　工作任务风格总结

元素	软件元素：模块； 环境元素：组织单元，如人员、团队、部门和分包商
关系	“分配到……”关系，描述将模块分配到配置条目
元素特征	技能集：所需的技能和提供的技能
关系特征	无
其他	分层配置条目：“包容在……”

工作任务风格与模块分解风格关系密切，它能将模块分解风格用作其分配映射的基础。这种风格能通过添加与开发工具、测试工具和配置管理系统等对应的模块分解进行扩展。工作任务风格还通常与其他风格联合使用，例如，团队工作任务可以是模块分解风格中的模块，可以是分层图中的层，也可以是多进程系统中的任务或进程。

9.11.5 各视图类型间的映射关系

为了完整地描述一个架构，必须使用多个视图，这些视图必须遵守一定的映射关系。

（1）模块视图类型中的视图通常会映射到构件和连接件视图类型中的视图。模块实现单元将映射到运行时构件。

（2）系统的构件和连接件视图和模块视图之间的关系可能会非常复杂。同样的代码模块可由 C&C 视图的许多元素执行。反之，C&C 视图的单一构件可执行由许多模块定义的代码。同样，C&C 构件可能会拥有许多与环境进行交互的点，每个交互点由同一模块接口定义。

（3）分配视图类型是为有效地实现软件架构的辅助性视图，它将其他视图类型中的软件元素映射到软件环境中，即反映其他视图与软件环境之间的关系。

第 10 章：设计模式

面向对象技术为软件技术带来新的发展。人们运用面向对象的思想分析系统、为系统建模并设计系统，最后使用面向对象的程序语言来实现系统。但是面向对象的设计并不是一件很简单的事情，尤其是要设计出架构良好的软件系统更不容易。为了提高系统的复用性，需要进行一些“额外”的设计（这里的额外并不是无用的，而是指业务领域之外），定义类的接口、规划类的继承结构、建立类与类之间的关系。毋庸置疑，良好的设计可以让系统更容易地被复用、被移植和维护，而如何快速进行良好的设计则是设计模式要讨论的问题。设计模式是软件架构设计师的必修课，设计模式中蕴含的思想是架构设计师必须掌握的。

10.1 设计模式概述

在 20 世纪 70 年代，Christopher Alexander 提出了城市建筑的模式，他认为：模式就是描述一个不断发生的问题和该问题的解决方案。随后，Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 写了一本著名的参考书《设计模式：可复用面向对象软件的基础》。后人也因为这本书称这四个人为四人组，将这本书中描述的模式称为 GoF（Gang of Four）设计模式。在这本书中，四人组将设计模式定义为：对被用来在特定场景下解决一般设计问题的类和互相通信的对象的描述。通俗地说，可以把设计模式理解为对某一类问题的通用解决方案。

10.1.1 设计模式的概念

首先，设计模式解决的是一类问题，例如工厂模式就是为了解决类创建的问题，而适配器模式则是为了解决类接口不匹配的问题。如果把解决 A 问题的设计模式使用在 B 问题上，结果肯定是张冠李戴了。所以在描述设计模式前，首先要描述这个设计模式究竟要解决什么样的问题。

其次，设计模式是一种通用的解决方案，而不是具体的，也不是唯一的。在 GoF 的书中对设计描述主要着重于思想的描述，虽然也给出了 C++的实现方法，但同样也可以使用 Java 甚至非面向对象的语言实现。具体应用时可以根据实际情况进行相应的变化，例如，对于工厂模式就有很多种变化。

需要指出的是，虽然在 GoF 的著作中第一次提出了软件的设计模式，但设计模式并非这四人所创，它来源于很多项目中的成功设计，并将这些优雅的设计方式进行抽象、总结、归纳出来。

在学习设计模式时需要注意以下两点：

（1）学习这些模式是一个方面，另一方面更要了解模式中的思想。设计模式本身是为了提高软件架构的质量，学习设计模式的目的也是为了提高架构设计的水平。虽然设计模式中描述的大多是面向对象的低层设计方案，但其中包含的却是软件设计的思想，同软件架构风格是一致的。例如，MVC 既可以看作一种设计模式也可以看作一种架构风格。掌握这种设计思想是非常有意义的。

（2）设计模式虽然可以使设计变得更精妙，但滥用设计模式会适得其反。在软件设计中使用设计模式，可以优化设计，提高架构质量。但是，首先，设计模式有其应用的场合，不相宜的场合乱用设计模式有害无益；其次，设计模式主要解决对象之间相互通信、相互依赖的结构关系，架构设计师需要把握好使用设计模式的力度，过度的使用设计模式不但不会提高软件的复用性，反而会让架构变得混乱而难以维护。

10.1.2 设计模式的组成

一般的，在描述一个设计模式时，至少需要包含四个方面：模式名称（Pattern name）、问题（Problem）、解决方案（Solution）、效果（Consequence）。这四个方面就是设计模式的四要素。名不正则言不顺，每种设计模式都有自己的名字，也就是模式名称；设计模式都有其应用的场合，即该设计模式意图解决的问题，超出了这个问题就不应该再应用这种模式，

所以问题是设计模式的第二要素；设计模式的目的就是解决问题，所以在描述设计模式时当然要有解决问题的方法描述，这就是设计模式的另外一个要素——解决方案；虽然架构设计师知道应用设计模式可以提高架构质量，提高软件的复用性，但对于每一种设计模式而言，还有其更具体的效果描述，所以设计模式的最后一个要素就是效果。

这四个要素是描述设计模式时必不可少的部分。在本章中，除了描述模式的四要素外，还补充了该模式的 Java 实现和对该模式的引申，也就是说，本章中的模式将按照如下的方式描述：模式名称、意图解决的问题、模式描述、效果、实现、相关讨论。在 10.2 节中，将使用这种方式介绍几种有代表性的设计模式。

10.1.3 GoF 设计模式

GoF 的著作不反第一次总结了设计中的常用模式，还在学术上建立了软件设计模式的地位。因此，人们习惯上将 GoF 提出的 23 个模式统称为 GoF 模式，这 23 个模式分别简述如下。

（1）Factory Method 模式。Factory Method 模式提供了一种延迟创建类的方法，使用这个方法可以在运行期由子类决定创建哪一个类的实例。

（2）Abstract Factory 模式。Abstract Factory 又称为抽象工厂模式，该模式主要为解决复杂系统中对象创建的问题。抽象工厂模式提供了一个一致的对象创建接口来创建一系列具有相似基类或相似接口的对象。抽象工厂模式是一种很有代表性的设计模式，在 9.2 节中将对该模式进行更详细的介绍。

（3）Builder 模式。Builder 模式与 Abstract Factory 模式非常类似，但 Builder 模式是逐步地构造出一个复杂对象，并在最后返回对象的实例。Builder 模式可以把复杂对象的创建与表示分离，使得同样的创建过程可以创建不同的表示。

（4）Prototype 模式。Prototype 模式可以根据原型实例制定创建的对象的种类，并通过深复制这个原型来创建新的对象。Prototype 模式有着同 Abstract Factory 模式和 Builder 模式相同的效果，不过当需要实例化的类是在运行期才被指定的而且要避免创建一个与产品曾是平行的工厂类层次时，可以使用 Prototype 模式。使用 Prototype 模式可以在运行时增加或减少原型，比 Abstract Factory 和 Builder 模式更加灵活。

（5）Singleton 模式。Singleton 模式也是一种很有代表性的模式。使用 Singleton 模式可以保证一个类仅有一个实例，从而可以提供一个单一的全局访问点。将在 9.2 节中对

Singleton 作更详细的介绍。

（6）Adapter 模式。Adapter 模式可以解决系统间接口不相容的问题。通过 Adapter 可以把类的接口转化为客户程序所希望的接口，从而提高复用性。

（7）Bridge 模式。Bridge 模式把类的抽象部分同实现部分相分离，这样类的抽象和实现都可以独立地变化。

（8）Composite 模式。Composite 模式提供了一种以树形结构组合对象的方法，使用 Composite 可以使单个对象和组合后的对象具有一致性以提高软件的复用性。

（9）Decorator 模式。Decorator 模式可以动态地为对象的某一个方法增加更多的功能。在很多时候，使用 Decorator 模式可以不必继承出新的子类从而维护简洁的类继承结构。在 9.2 节中将对 Decorator 模式作更详细的介绍。

（10）Facade 模式。Facade 模式为一组类提供了一致的访问接口。使用 Facade 可以封装内部具有不同接口的类，使其对外提供统一的访问方式。Facade 模式在 J2EE 系统开发中发展为 Session Facade 模式。

（11）Flyweight 模式。Flyweight 模式可以共享大量的细粒度对象，从而节省创建对象所需要分配的空间，不过在时间上的开销会变大。

（12）Proxy 模式。顾名思义，Proxy 模式为对象提供了一种访问代理，通过对象 Proxy 可以控制客户程序的访问。例如：访问权限的控制、访问地址的控制、访问方式的控制等，甚至可以通过 Proxy 将开销较大的访问化整为零，提高访问效率。

（13）Interpreter 模式。定义了一个解释器，来解释遵循给定语言和文法的句子。

（14）Template Method 模式。定义一个操作的模板，其中的一些步骤会在子类中实现，以适应不同的情况。

（15）Chain of Responsibility 模式。Chain of Responsibility 模式把可以响应请求的对象组织成一条链，并在这条对象链上传递请求，从而保证多个对象都有机会处理请求而且可以避免请求方和相应方的耦合。

（16）Command 模式。将请求封装为对象，从而增强请求的能力，如参数化、排队、记录日志等。

（17）Iterator 模式。Iterator 模式提供了顺序访问一个对象集合中的各元素的方法，使用 Iterator 可以避免暴露集合中对象的耦合关系。

（18）Mediator 模式。Mediator 模式可以减少系统中对象间的耦合性。Mediator 模式使用中介对象封装其他的对象，从而使这些被封装的对象间的关系就成了松散耦合。

（19）Memento 模式。Memento 模式提供了一种捕获对象状态的方法，且不会破坏对象的封装。并且可以在对象外部保存对象的状态，并在需要的时候恢复对象状态。

（20）Observer 模式。Observer 模式提供了将对象的状态广播到一组观察者的方式，从而可以让每个观察者随时可以得到对象更新的通知。

（21）State 模式。State 模式允许一个对象在其内部状态改变的时候改变它的行为。

（22）Strategy 模式。使用 Strategy 模式可以让对象中算法的变化独立于客户。

（23）Visitor 模式。表示对某对象结构中各元素的操作，使用 Visitor 模式可以在不改变各元素类的前提下定义作用于这些元素的新操作。

10.1.4 其他设计模式

在 GoF 之后，人们继续对设计模式进行发掘，总结出更多的设计模式。在 J2EE 应用领域，人们也对使用 J2EE 框架开发的应用程序总结出一系列设计模式，本节对几种较有代表性的 J2EE 设计模式进行简要介绍。因为这些设计模式是同 J2EE 技术紧密相关的，所以本节中将会使用一些 J2EE 技术术语。

（1）Intercepting Filter 模式。在 J2EE 的 BPS（Basic Programming System，基本编程系统）应用框架下，在真正响应客户端请求前经常需要进行一些预处理，如客户身份验证、客户 Session 的合法性验证、字符集转码、客户请求记录等。当然可以将这些请求预处理在每一个 Servlet 中，不过很明显，这样的话预处理的代码就“侵入”了真正的处理程序，使得代码变得更加难以维护。Intercepting Filter 模式提供了解决这个问题的方法。它通过截取客户请求，并将请求发送到 Filter 链中，一步一步地进行预处理，直到这些处理结束，请求才会被转发到真正响应客户请求的 Servlet 中。关于 Interception Filter 模式的详细内容，请参考 10.2 节。

（2）Session Facade 模式。Session Facade 模式广泛应用于 EJB 开发的 J2EE 应用程序中。EJB 是一种分布式构件，EJB 的客户端需要通过 EJB 容器调用 EJB，即使 EJB 的客户端同 EJB 部署于同一台机器，对 EJB 的调用也许要通过网络接口进行远程调用。因此，在开发 EJB 时，需要尽量减少对 EJB 调用的次数以提高性能。同时为了提高 EJB 构件的可维护性和复用性，应该尽量将 EJB 构件的接口设计得一致。在 GoF 设计模式中就有 Facade 模式提高接口的一致性，在 J2EE 开发领域，人们把 Session Bean 和 Facade 模式结合起来，封装业务逻辑的接口，形成了 Session Facade 模式。关于 Session Facade 模式的详细内容，

请参考 10.2 节。

10.1.5 设计模式与软件架构

软件架构描述了软件的组成，例如，经典的“4+1”视图，将软件架构通过逻辑视图、开发视图、进程视图、物理视图及场景视图来进行描述。在这些视图中，描述了软件系统中类之间的关系、进程之间的关系、软件和硬件的结合等问题。一般来说，软件架构更倾向于从整体和全局上描述软件的组成。

而设计模式则更侧重于类与类、对象与对象之间的关系。例如在逻辑视图中，可以使用多种设计模式来组织类与类之间的关系。因此，有很多人认为，设计模式和软件架构是面向不同层次问题的解决方案。

同设计模式一样，软件架构也有一些固定的模式，通常称为架构风格。常见的架构风格有分层架构、客户端—服务器架构、消息总线、面向服务的架构（Service-Oriented Architecture，SOA）等。

软件架构风格同设计模式在某种含义上是一致的。设计模式和软件架构中蕴含的很多思想是一致的。无论是架构风格还是设计模式，人们在追求良好设计的过程中，将一些常见解决方案总结、整理出来，形成固定的风格与模式。例如消息总线的架构风格同 Observer 模式就有神似之处。因此，掌握设计模式对于软件架构设计有非常大的帮助。

10.1.6 设计模式分类

可以说，设计模式是面向问题的，即每一种设计模式都是为了解决一种特定类型的问题。因此，根据设计模式要解决的问题将设计模式分为三类，分别为创建型、结构型和行为型。

事实上，面向对象的设计中，需要解决的就是：如何管理系统中的对象、如何组织系统中的类与对象、系统中的类与对象如何相互通信。这三类设计模式分别解决了这三个方面的问题。

创建型设计模式主要解决对象创建的问题。在最简单的情况下，在程序中定义类，在使用时创建一个对象实例。但在实际开发中，对象的创建会变得复杂很多，这时就需要使用创建型设计模式解决创建对象的问题。

随着开发系统的不断扩张，系统功能更加丰富，模块之间的复用越来越多，系统中类与对象的结构变得更加复杂。如果缺乏良好的设计，这些类之间的关系将会变得非常混乱。结

构型设计模式就是为了解决这些问题的。

除了这种分类方法外，GoF 还提出了可以根据设计模式主要应用于类还是对象来对设计模式进行分类，对于这种分类方法就不再赘述了。综合这两种分类方法，可以把 GoF 模式进行分类，如表 10-1 所示。

表 10-1　GoF 模式分类

GoF 模式				
		创建型	结构型	行为型
应用范围	应用于类	Factory Method	Adapter	Interpreter Template Method
	应用于对象	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

随着 GoF 设计模式的提出，后人也总结出了更多的良好设计的范本，并根据其他的方法进行分类。例如，在《Core J2EE Patterns》一书中，作者将书中列举的 Design Pattern 分为表现层模式、业务层模式和综合层模式。根据这种分类方法，可以得到应用于 J2EE 框架的设计模式图谱，如表 10-2 所示。

表 10-2　J2EE 设计模式分类

J2EE 设计模式		
表现层	业务层	综合层
Intercepting Filter Front Controller View Helper Composite View Service to Worker Dispatcher View	Business Delegate Value Object Session Facade Composite Entity Value Object Assembler Value List Handler Service Locator	Data Access Object Service Activator

10.2 设计模式及实现

本节就几种常用的设计模式，讨论其实现问题。

10.2.1 Abstract Factory 模式

1．模式名称

Abstract Factory，也经常称之为抽象工厂模式。

2.意图解决的问题

在程序中创建一个对象似乎是不能再简单的事情，其实不然。在大型系统开发中存在以下问题：

（1）object new ClassName 是最常见的创建对象方法，但这种方法造成类名的硬编码，需要根据不同的运行环境动态加载相同接口但实现不同的类实例，这样的创建方法就需要配合上复杂的判断，实例化为不同的对象。

（2）为了适用于不同的运行环境，经常使用抽象类定义接口，并在不同的运行环境中实现这个抽象类的子类。普通的创建方式必然造成代码同运行环境的强绑定，软件产品无法移植到其他的运行环境。

抽象工厂模式就可以解决这样的问题，根据不同的配置或上下文环境加载具有相同接口的不同类实例。

3．模式描述

Abstract Factory 模式的结构如图 10-1 所示。

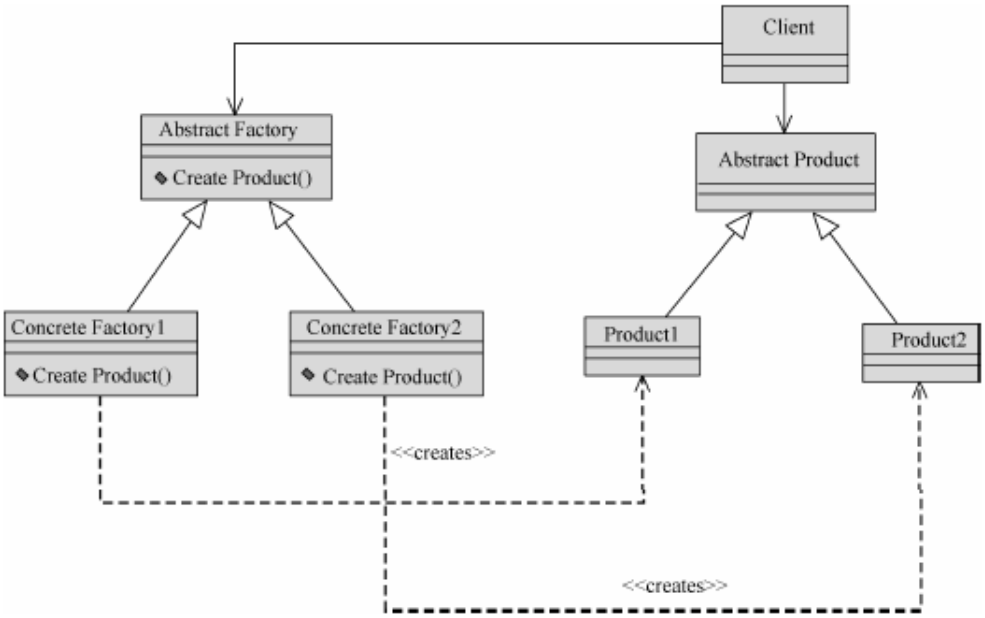


图 10-1　Abstract Factory 结构图

就如同抽象工厂的名字一样，Abstract Factory 类将接受 Client 的“订单”——Client 发送过来的消息，使用不同的“车间”——不同的 Concrete Factory，根据已有的“产品模型”——Abstract Product，生产出特定的“产品”——Product。不同的车间生产出不同的产品供客户使用，车间与产品的关系是一一对应的。由于所有的产品都遵循产品模型——Abstract

Product，具有相同的接口，所以这些产品都可以直接交付客户使用。

在抽象工厂模式中，Abstract Factory 可以有多个类似于 Create Product()的虚方法，就如同一个工厂中有多条产品线一样。Create Product1()创建产品线 1，Create Product2()创建产品线 2。

希赛教育专家提示：在 Abstract Factory 中，Create Product()方法与 Abstract Product 是一一对应的，而 Concrete Factory 的数量同实际的 Product 的数量是一致的。

4．效果

应用 Abstract Factory 模式可以实现对象可配置的、动态的创建。灵活运用 Abstract Factory 模式可以提高软件产品的移植性，尤其是当软件产品运行于多个平台，或有不同的功能配置版本时，抽象工厂模式可以减轻移植和发布时的压力，提高软件的复用性。

5．相关讨论

在实际应用中，Abstract Factory 可以有更灵活的变化。事实上，如果仔细观察 AbstractFactory 模式就可以发现，对于 Client 来说，最关注的就是在不同条件下获得接口一致但实现不同的对象，只要避免类名的硬编码，采用其他方式也可以实现。所以也可以采用其他的方式实现。例如在 Java 中就可以采用接口的方式实现，如图 10-2 所示。

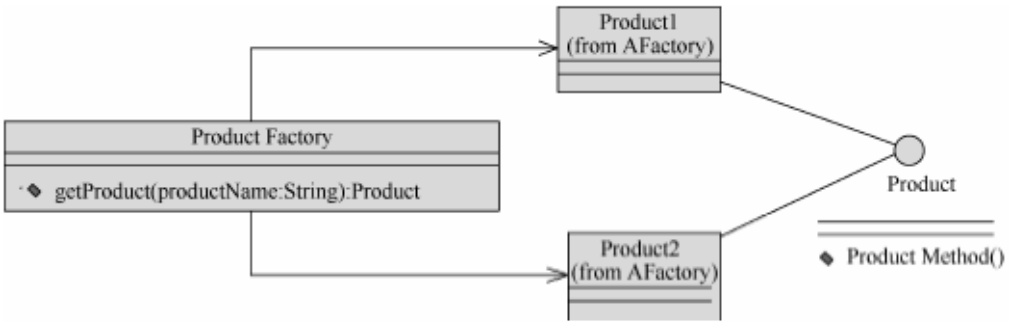


图 10-2　面向接口的简单工厂模式

图 10-2 中所描绘的类就应用了 Abstract Factory 的思想，采用面向接口的方式，简单实现了工厂模式。Product Factory 既可以看作抽象工厂 Abstract Factory，也可以看作具体的工厂（车间） Concrete Factory 。其中提供了一个获得产品的方法： getProduct（productName:String），该方法将根据产品的名称创建特定的产品，并返回。所有的产品都实现了 Product 接口，可以在客户程序中加以使用。同抽象工厂模式一样，工厂中获得对象的方法（getProduct()）与实际的产品线数量是一致的，如果要增加新的产品线——例如定义新的产品接口 ProductX，需要增加相应的 getProduct()方法。由于这种方式把 Abstract Factory 和 Concrete Factory 合并为一个 Product Factory，所以增加新的实现 Product 接口

的 Productn 不需要修改任何代码。

除了这种面向接口的方法外，工厂模式还可以有更多的变化。前面已经说过，学习设计模式最终要的是学习设计思想，学习了工厂模式后，应该知道：

① 可配置的对象创建方法可以提高系统的移植性和复用性。

② 充分利用面向对象多态的特性可以避免对象创建过程的硬编码。

10.2.2 Singleton 模式

1．模式名称

Singleton，也常称之为单件模式或单根模式。

2．意图解决的问题

在软件开发中，开发人员希望一些服务类有且仅有一个实例供其他程序使用。例如，短消息服务程序或打印机服务程序，甚至对于系统配置环境的控制，为了避免并发访问造成的不一致，也希望仅为其他程序提供一个实例。

对于供整个系统使用的对象可以使用一个全局变量，不过全局变量仅能保证正确编码时使用了唯一的实例。但随着系统不断的扩张，开发队伍的扩大，仍然无法保证这个类在系统中有且仅有一个实例。

3．模式描述

Singleton 模式可以解决上面的问题，其结构如图 10-3 所示。

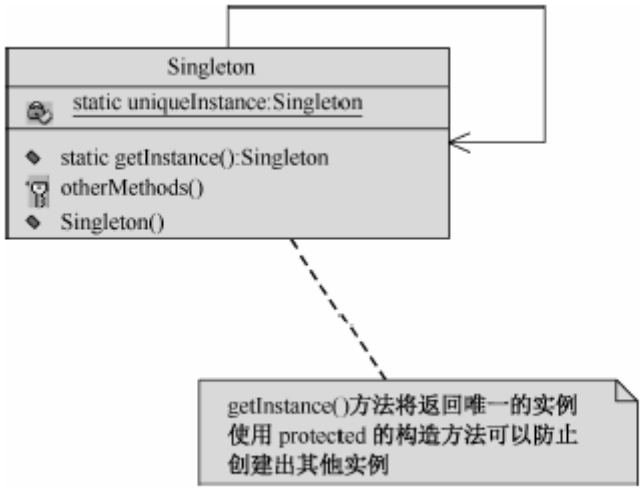


图 10-3 Singleton 结构图

从结构角度而言，Singleton 是最简单的一个模式，不过其用途很广。在 Singleton 中，通过将 Singleton 类的构造函数设为 protected 型（或 private）来防止外部对其直接初始

化。需要访问 Singleton 的程序必须通过 getInstance()方法来获得一个 Singleton。在 getInstance() 中仅创建一次 uniqueInstance 就可以保证系统中的唯一实例。

对于 Singleton 中的 uniqueInstance 有两种不同的初始化策略（Lazy Initialization 和 Early Initialization），在实现中将分别给出这两种初始化策略的代码。

4．效果

使用 Singleton 模式可以保证系统中有且仅有一个实例，这对于很多服务类或者环境配置类来说非常重要。

5．相关讨论

在使用 Singleton 模式时，需要注意：

（1）Singleton 仅能保证在单一系统内的单一实例。如果使用分布式构件，如运行在多个 JVM 下的 EJB，上面的实现方法是不能保持整个系统中的单一实例的。

（2）Singleton 模式仅适用于系统中至多有一个实例的情况，应避免滥用。很多过度设计的 Singleton 同使用了静态方法的工具类一样，没有任何必要，反而可能降低效率。

（3）从 Singleton 的实现就可以看出，Singleton 不支持继承。对于 C++等支持 Template 技术的开发语言，可以定义 Singleton 模板来构造 Singleton，进一步提高该模式的复用性。但在 Java、C#等语言中，则只有采用其他的方法实现。不过 Singleton 不等同于 static 类，所以一般来说，系统中不会也不应该出现很多 Singleton 类，在不支持 Template 的语言中逐一实现也未尝不可。

如果一定要在不支持 Template 的语言中实现批量的 Singleton 类，可以参考有关文献中的方法。

10.2.3 Decorator 模式

1．模式名称

Decorator 模式，又称装饰模式或油漆工模式。

2．意图解决的问题

在开发时，经常会发现为类预先设计的功能并不够强大，需要增强或扩展类的功能。解决这个问题的最简单的办法是继承出一个新的类，并扩展相应的方法。但是这样做会产生大量的子类，让系统中类的层次结构变得复杂且混乱。Decorator 模式通过在原有类的基础上包装一层来解决功能扩展的问题。

3．模式描述

Decorator 模式的结构如图 10-4 所示。

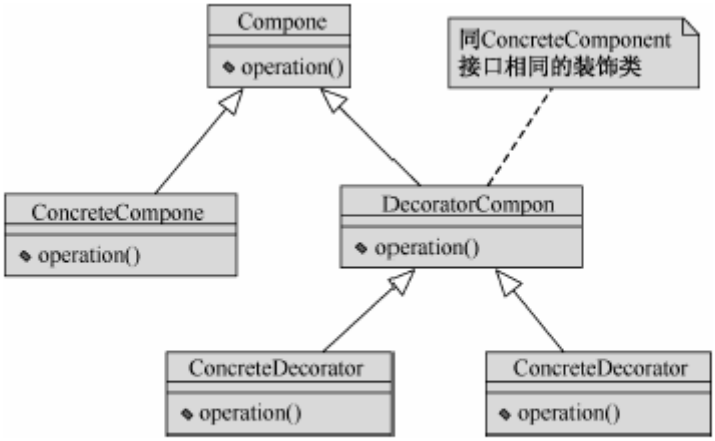


图 10-4　Decorator 模式

DecoratorComponent 是 ConcreteComponent 的装饰类，它们继承自同一个抽象类 Component，拥有相同的接口。对于 ConcreteComponent 的装饰可能不止一种，因此又继承出 ConcreteDecorator1 和 ConcreteDecorator2 来作为具体的装饰类。这个结构在类层次上是很清晰的，比静态继承更具有灵活性。图 10-5 中使用顺序图描述了使用 Decorator 类的方法。

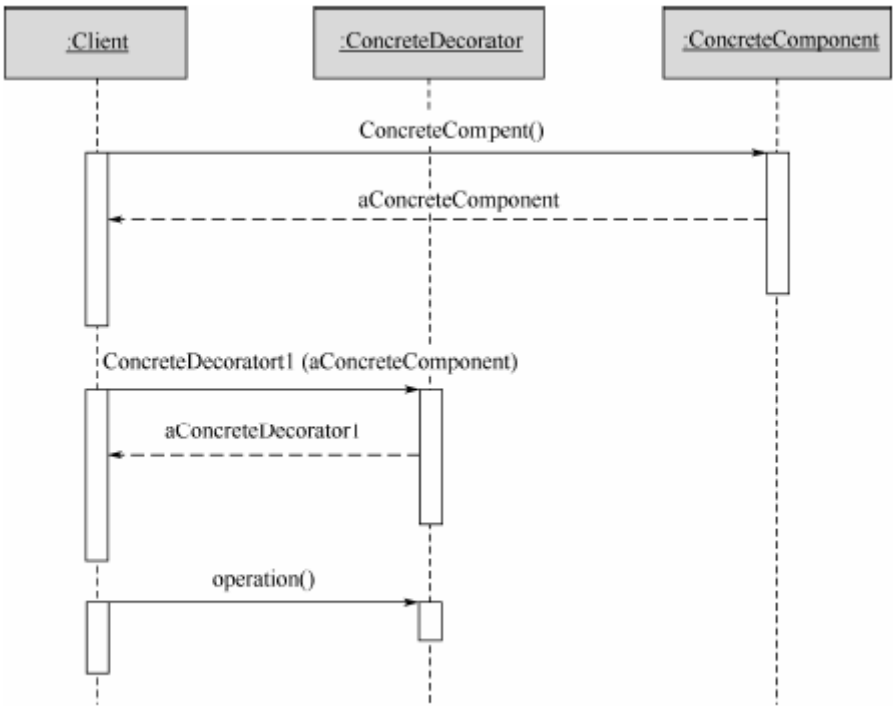


图 10-5　Decorator 类的使用方法

通过图 10-4 的方法，对于接口相同的 operation()，在运行期可以根据客户的选择具有不同的行为特征，实现功能动态的扩展。

4．效果

Decorator 模式可以动态地扩展类的功能，同时又避免继承出大量的子类，比继承的方法更灵活。由于 Decorator 提供了动态的扩展方法，因此可以随时根据具体需求产生新的装饰类，从而可以在一定程度上避免层次较高类的复杂性。但是大量使用 Decorator 模式会造成系统中出现很多接口类似的小装饰对象，这样就造成系统可维护性的下降。

5．相关讨论

Decorator 模式提供了一种动态为类扩展功能的方法，有着较广的应用。例如，在开源项目 displaytag（一个提供 JSP 标签扩展的开源项目，可以简单地在网页中画出美观的表格，http://displaytag.sourceforge.net/）中就提供了以 Decorator 方式扩展功能，进行二次开发的接口，在 JDK 的 I/O API 中，也大量使用了 Decorator 模式。

同其他模式一样，滥用 Decorator 会降低系统的可维护性。如果开发出来的装饰类仅被单一的地方使用或只进行了相当简单的处理，就需要考虑是否有必要使用 Decorator 模式了

10.2.4 Facade/Session Facade 模式

1．模式名称

Facade 模式，又称外观模式，也有人很形象地把它翻译成“门面模式”。

2．意图解决的问题

在程序中，经常会用到其他若干个子系统。在不作任何处理的时候，需要了解每一个子系统的接口，并使用这些接口进行调用，于是系统就如图 10-6 所示一样混乱。

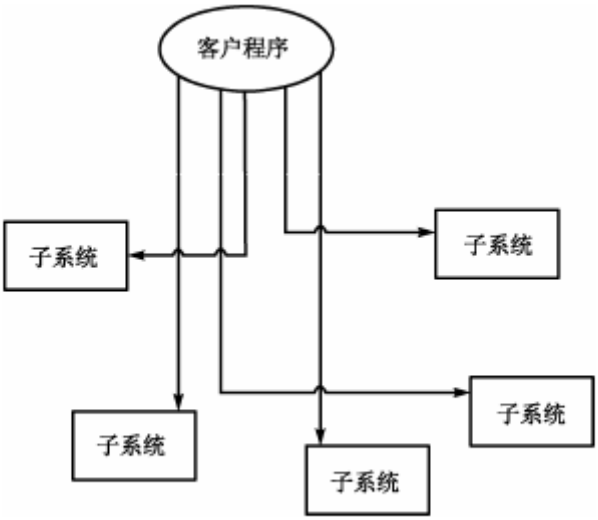


图 10-6　混乱的系统间调用

这些调用不但让结构变得混乱，客户程序和各子系统的耦合性也大大增加，扩展与维护都变得相当困难。

3．模式描述

Facade 模式的结构如图 10-7 所示。

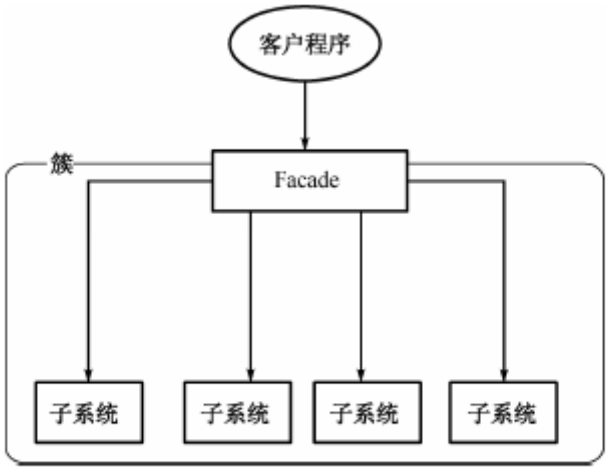


图 10-7　Facade 模式

如图 10-7 所示，Facade 模式通过在原有系统前增加一层的方法，封装这些子系统的接口，对外提供一致的访问接口，解决了上面的问题。

4．效果

Facade 模式屏蔽了子系统的细节，降低了客户程序使用这些子系统的复杂度，同时也降低了客户程序和子系统的耦合程度。这样就从需要让所有的人了解所有的子系统接口变成让个别专家抽象子系统的接口。

Facade 模式应用起来非常灵活，也没有特定的实现，其应用的关键就是抽象出合理的 Facade 接口。

5．相关讨论

Facade 模式很好地体现了封装的思想，它封装的是一个子系统的内部结构。例如，经常对数据访问对象（Data Access Object，DAO）进行封装，使数据访问同具体的数据库相分离。客户程序只要知道最外层的数据库访问构件的接口就可以操纵任何支持的数据库。

在 EJB 开发领域，又根据 Facade 模式引申出 Session Facade 模式。

EJB 是一种分布式构件，J2EE 将对 EJB 的访问定义为远程访问。例如，使用 Servlet 访问 EJB 时，即使 Servlet 和 EJB 部置于同一台主机，但由于 Servlet 和 EJB 分置于不同的容器（Servlet 在 Web 容器中，EJB 在 EJB 容器中），Servlet 的默认访问方式也是通过本机的回播地址（127.0.0.1）访问 EJB，从而造成 EJB 访问的效率问题。而 EJB 中封装了复杂的业务对象，把这些业务对象的全部暴露给外部也不利于系统的维护。所以通常使用 Session Bean 封装这些业务逻辑，提高访问效率，减少内部业务逻辑和外部访问者的耦合，封装服务器中的对象模型，如图 10-8 所示。

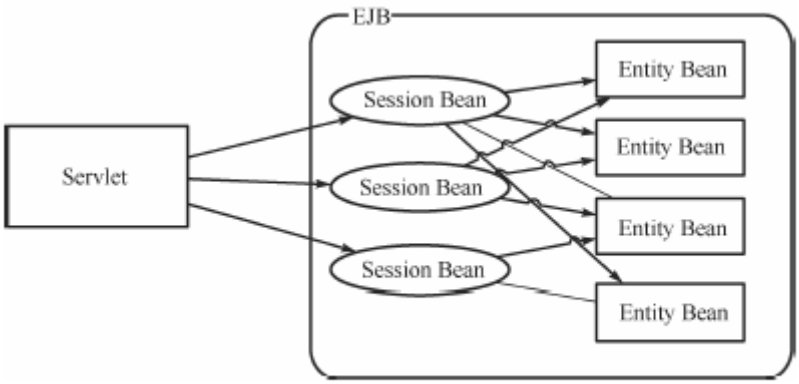


图 10-8　Session Facade 模式

在 Session Facade 模式中，一般使用 Session Bean 作为业务逻辑的接口，在 Session Bean 中进行事务处理，确保客户对 SessionBean 的每一次访问即完成一次业务操作。使用 Session Facade 模式除了获得 Facade 模式的优点外，还可以减少远程调用 EJB 的次数，降低网络接口的负载，提高性能。

10.2.5 Mediator 模式

1．模式名称

Mediator 模式，又称中介者模式。

2．意图解决的问题

在一个复杂系统中会有很多对象，这些对象之间会相互通信，从而造成对象的相互依赖。修改其中一个对象可能会影响到其他若干对象，系统中对象的复杂耦合关系造成系统可维护性的降低，系统显得混乱且难以理解。Mediator 通过封装一组对象间的通信为系统解耦。

3．模式描述

Mediator 模式的结构如图 10-9 所示。

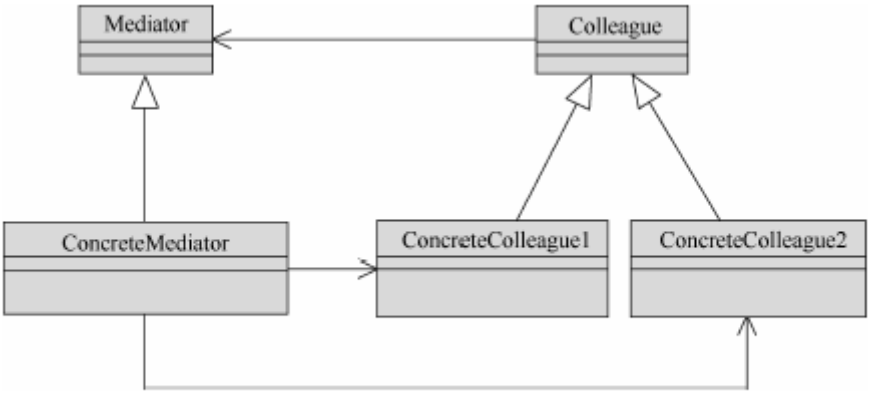


图 10-9　Mediator 模式

在 Mediator 模式中，一组对象 ——抽象类 Colleague 的子类，通过中介类 ——ConcreteMediator 进行通信，中介类 ConcreteMediator 继承了抽象类 Mediator，其中需要实现 Mediator 中定义的通信接口，保证中介类和相互通信类之间接口的一致。

Mediator 模式看上去类似于消息机制，其实和消息机制有很大的区别。Mediaor 不需要负责消息的排队、优先级等处理，但它必须根据 Colleague 发送的消息作出响应，并向特定的 Colleague 类发送消息。换句话说，ConcreteMediator 类中封装了 Colleague 之间的通信，就好比人的大脑，接收到眼睛发送的“前面有一个坑”的消息，就会向脚发送“停止前进”的消息。

4．效果

Mediator 封装了一组对象间的通信，降低了 Colleague 之间的耦合性，单个 Colleague 的变化不会影响到其他的对象。还举前面大脑的例子，在行走的时候，大脑接收到眼睛发送的“前面有状况”的消息会通知脚“停止前进”，而在开车的时候，大脑接收这条消息后则会向脚发送“踩下刹车”的消息。

不过当 Colleague 比较多，且其中的关系复杂时，中介类 ConcreteMediator 会变得非常复杂，难以维护。

5．相关讨论

Mediator 模式在 GUI 方面有较多的使用。当一个窗体或表单或对话框中有多个需要互相通信的对象时——例如显示部门中人员的下拉列表需要根据部门下拉列表不同的选择而变化，使用 Mediator 封装通信可以降低这些对象的耦合性，提高系统的可维护性。

希赛教育专家提示：Mediator 经常会变得非常复杂，不但每一个需要通信的对象都需要知道 Mediator 的存在，而且 Mediator 也需要知道每一个通信的对象的实现。所以，一般仅在较小的范围内使用 Mediator 模式，否则过多的 Colleague 类造成中介类的难以维护会抵消掉该模式带来的优点，反而让系统更难以维护。

10.2.6 Observer 模式

1．模式名称

Observer 模式，又称观察者模式。

2．意图解决的问题

由于对象封装的特性，一般的，在系统中对象状态的变化都是独立的，即 A 对象状态的变化不会立即反映到 B 对象。但是，在系统中很多对象是相互关联的，例如对于一个股票行情系统，股票状态的变化需要同时反映到多个视图中——实时行情、技术分析图表等。对于这个问题，最简单的解决办法就是硬编码这些关联关系。但是这样会造成系统中对象的紧密耦合，系统难以维护和复用。

3．模式描述

Observer 模式的结构如图 10-10 所示。

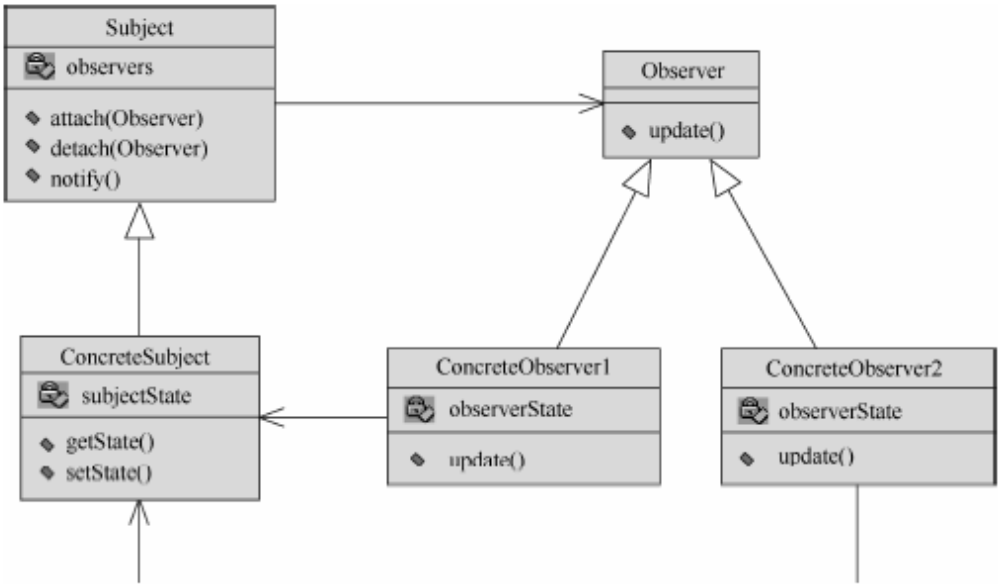


图 10-10　Observer 模式

抽象类 Subject 定义了主题类——也就是被观察者的接口，它的子类 ConcreteSubject 的任何状态改变将会通知全部的观察者——ConcreteObserver。为了保持接口的一致性，这些观察者继承自相同的抽象类 Observer。

抽象类在 Subject 中保存有观察者的列表——observers，并通过 attach 和 detach 方法来动态地添加或移出一个特定的观察者。这种采用订阅方法来添加的观察者并不知道还有其他的观察者，它仅仅能够接收被观察的主题对象的状态改变。在 notify 方法中，主题类将根据目前加入订阅的观察者列表 observers 来向每一个观察者发出状态改变的消息。

ConcreteSubject 中的 setState()方法表明对象的状态发生了变化，该方法会调用 notify 方法对所有的观察者进行更新，如图 10-11 所示。

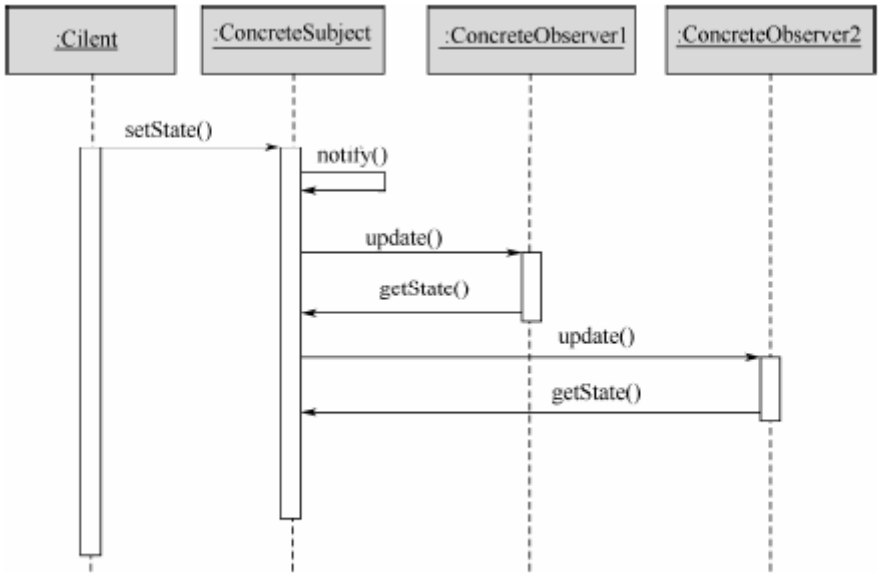


图 10-11　Observer 模式的工作方式

4．效果

在 Observer 模式中，实现了具体对象和观察者之间的抽象耦合。虽然 Subject 了解目前有哪些观察者需要捕获自己状态的变化，但它并不了解这些观察者要做什么；而每一个观察者仅仅知道自己捕获到对象的变化，但并不清楚目前有多少观察者在观察对象的状态，也不需要通知其他的观察者。这样，动态的增加和移除观察者是非常简单的。

通过 Observer 模式可以实现对象间的消息广播，这在很多处理中非常有用。不过广播的副作用是增加了系统的负载，任何消息都将被自动地发送到每一个观察者中，每一个观察者都将根据这些消息作出响应，这些响应未必都是必要的。

5．相关讨论

Observer 模式将一对多的对象依赖转化为抽象耦合，提高系统的复用价值，得到了广泛的应用。Smalltalk 的 MVC 架构中就应用了 Observer 模式，将 Model 的变化传递到 View 中。

但是 Observer 使用不当会造成很多问题。在图 9-9 中描述的更新方法是在 setState()后由 Subject 自动执行 nofity()，向所有的 Observer 发送通知，这种方法可以保证所有的更新都能够通知到观察者。但是对象状态并不总是孤立的，客户程序很可能在执行了 setState1()方法之后紧接着执行 setState2()方法，甚至要连续对对象的状态作若干次的更改。这样，每一次更改都会带来一个 noifty()事件，这些更改被放大到每一个 Observer 对象中。并发执行观察者的 update()方法很可能造成程序的错误，而独占的访问 update()方法又会造

成程序阻塞。

因此有时会采用如图 10-12 所示的方法发送对象状态变化的通知。

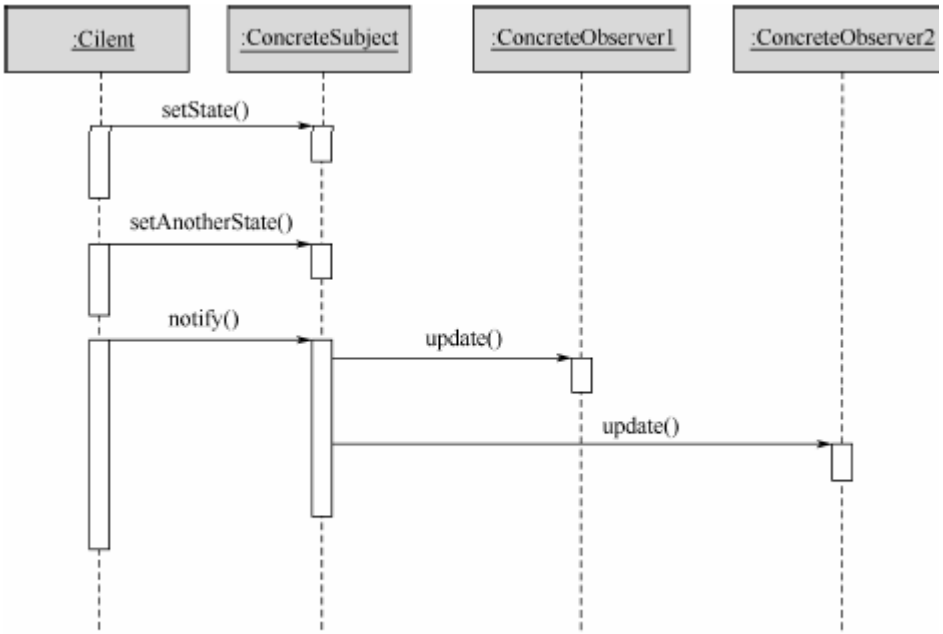


图 10-12　对象状态变化的通知方法

这种方法在客户程序执行了一系列改变对象状态的方法后调用 notify()方法，避免了上面的问题，不过这种方法首先不能保证对象状态的变化一定会被通知到各个观察者，其次还造成观察者对客户程序的不透明。

所以，在使用 Observer 模式时一定要注意精确地定义对象状态变化的依赖关系，以避免这些问题。

前面也提到使用 Observer 可以实现消息的广播。广播是一柄双刃剑，虽然可以简化一对多的消息发送但也将带来一系列的问题。了解计算机网络的读者知道以太网中的广播风暴问题，在 Observer 模式中需要尽力避免广播的扩大，在 update()对象中避免更新 Subject 类的状态，尤其不能更新自己观察的 Subject 类的状态。否则，可能会造成对象间的消息无穷循环下去。

10.2.7 Intercepting Filter 模式

1．模式名称

Intercepting Filter 模式，又称筛选器模式。

2．意图解决的问题

在使用 MVC 架构进行 Web 应用开发时，通常需要对来自于客户的请求进行一些预处理，如验证客户身份、验证请求来源、对请求解码等，然后再传递给控制器。如果把这些预处理都交由控制器来完成，将增加控制器的复杂度，而且难以维护和扩展。

3．模式描述

Intercepting Filter 模式的结构如图 10-13 所示。

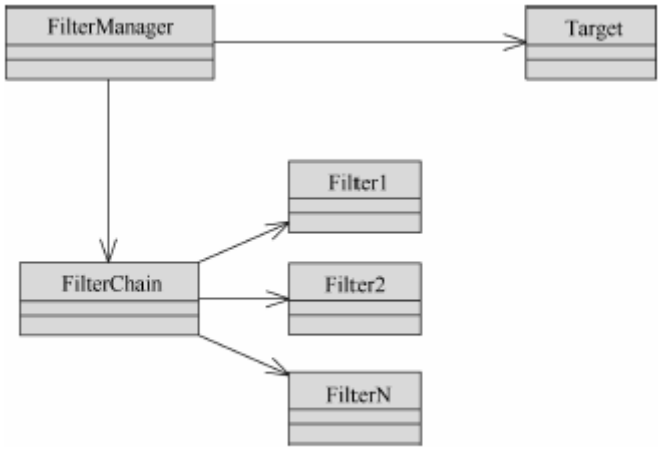


图 10-13　Intercepting Filter 模式

FilterManager 负责调度整个 FilterChain，它将在请求到达 Target 前拦截请求，并传递给 FilterChain，由 FilterChain 中的过滤器依次进行预处理。直到请求经过最后一个过滤器后才完成了全部的预处理，然后由 FilterManger 把请求转发给实际的目标。使用 InterceptingFilter 模式时的时序关系如图 10-14 所示。

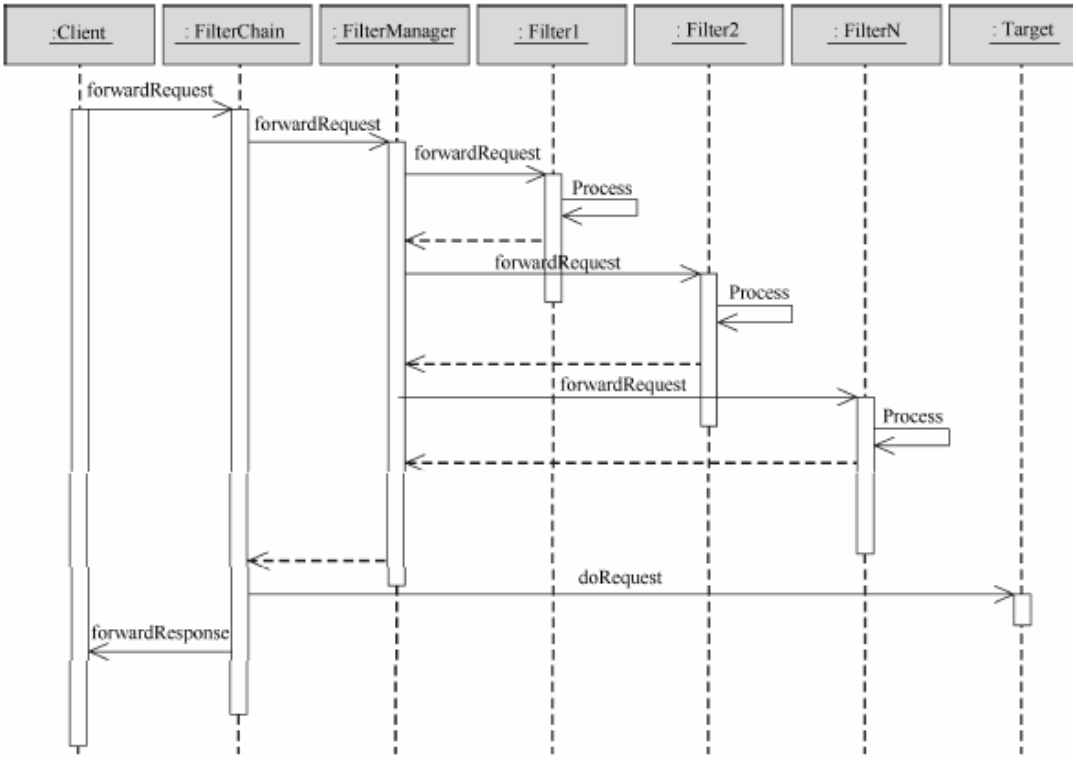


图 10-14　Intercepting Filter 模式序列图

4.效果

使用 Intercepting Filter 模式使得预处理的逻辑和真正的处理逻辑分离，进行实际处理的 Target 只需要关心具体的逻辑，而同请求相关的预处理都放在 FilterManager 中进行。同时解除了这两类处理的耦合性，扩展、修改预处理过程变得容易，系统具有更好的维护性和扩展性。

5．相关讨论

虽然 Intercepting Filter 作为一种 J2EE 模式出现在有关文献中，但实际上有很广泛的应用。不仅应用 J2EE 的开发可以使用该模式分离预处理过程，使用.Net 框架等其他的 Web 应用时也可以使用 Intercepting Filter。

仔细观察 Intercepting Filter 模式就可以发现，它与 Decorator 模式有些类似，都是在真正的处理对象前增加一层扩展对象的操作。只不过实现起来有很大的差别。

希赛教育专家提示：完全可以让 Intercepting Filter 变得更通用，更容易配置。如果对于不同的请求需要使用不同的 FilterChain——例如根据客户不同的编码方式使用不同的解码程序——可以使用工厂模式来动态创建 Filter 对象进行处理。类似的灵活使用设计模式的方法还有很多，也可以用于各种模式，需要架构设计师根据实际情况进行深入的思考。

10.3 设计模式总结

10.2 节详细讨论了 7 种设计模式：Abstract Factory、Singleton、Decorator、Facade/Session Facade、Mediator、Observer 和 Intercepting Filter。目前总结出的设计模式远远不止这些，除了 23 种 GoF 模式外，很多学者都在这方面进行了有益的尝试，总结出了大量良好设计的范例。

学习设计模式最重要的是理解，而不是生搬硬套。每种设计模式中都包含着良好的设计架构的思想，如隐藏内部细节、降低耦合度等，越是复杂的系统越需要这些思想的支撑。观察人类社会就可以发现很多与设计模式相同的地方。每一个人都是一个封装良好的对象，通过感官和行动提供了与外界沟通的媒介。人类对外接收消息的方式不过是视觉、听觉、嗅觉、触觉等有限的几种，但这些感觉器官封装了内部复杂的结构；可以使用望远镜、电话或其他的工具来扩展能力；对于人类的复杂组织，有间接沟通的渠道，中介所提供了交流的平台；新闻机构则可以让人们随时了解其他人和事的最新状态……类似的例子还可以举出来很多，如经纪人与 Proxy 模式非常相近，而绝大多数的组织看起来都像 Composite。软件系统的最终目的是辅助人类的活动，以良好的方式抽象现实世界才可以获得良好的设计，单纯的死记硬背这些模式不会有任何价值，与实际情况相联系才能得出易维、护易复用的系统来。

在使用中，经常会把几种模式综合起来解决复杂的问题，例如 10.2.7 中提到的 Intercepting Filter 和工厂模式的结合。这需要系统架构设计师具体问题具体分析，综合利用设计模式消除系统中的混乱与耦合，提高系统复用性。

希赛教育专家提示：要切记不能滥用设计模式，尤其在一些简单系统中。如果系统中的对象都用工厂模式创建、系统中的工具类都设计成 Singleton、两个对象间的通信还要硬加上一层 Mediator 等都是不可取的，只能毫无价值地提高系统复杂度，反而不利于系统的理解与维护。除最初的设计外，重构也是一个很好的时机，系统架构设计师可以在重构的时候根据需要逐步应用设计模式改良系统，提高系统的维护性和复用性。

第 11 章：测试评审方法

软件测试与评审是软件质量保证的主要手段之一，也是在将软件交付给客户之前所必须完成的步骤。目前，软件的正确性证明尚未得到根本的解决，软件测试与评审仍是发现软件错误（缺陷）的主要手段。

本章重点要求读者掌握测试方法、评审方法、验证与确认、测试自动化、面向对象的测试等 5 个方面的知识。

11.1 测试方法

在介绍软件测试之前，首先应该明确“错误”（error）和“缺陷”（fault）的概念。根据 IEEE 的定义，“错误”主要针对软件开发过程，“缺陷”主要针对软件产品。软件开发人员在软件开发过程（主要是分析、设计和编码过程）中所出现的“错误”是导致软件产品“缺陷”的原因，反过来说，“缺陷”是“错误”的结果和表现形式。

软件测试的目的就是在软件投入生产性运行之前，尽可能多地发现软件产品（主要是指程序）中的错误（缺陷）。

为了发现软件中的错误（缺陷），应竭力设计能暴露错误（缺陷）的测试用例。测试用例是由测试数据和预期结果构成的。一个好的测试用例是极有可能发现至今为止尚未发现的错误（缺陷）的测试用例。一次成功的测试是发现了至今为止尚未发现的错误（缺陷）的测试。

高效的测试是指用少量的测试用例，发现被测软件尽可能多的错误（缺陷）。

软件测试所追求的目标就是以尽可能少的时间和人力发现软件产品中尽可能多的错误（缺陷）。

11.1.1 软件测试阶段

从测试阶段上分，软件测试通常可分为单元测试、集成测试和系统测试。

1．单元测试

单元测试（unit testing），也称模块测试，通常可放在编程阶段，由程序员对自己编写的模块自行测试，检查模块是否实现了详细设计说明书中规定的功能和算法。单元测试主要发现编程和详细设计中产生的错误，单元测试计划应该在详细设计阶段制定。

单元测试期间着重从以下几个方面对模块进行测试：模块接口、局部数据结构、重要的执行通路、出错处理通路和边界条件等。

测试一个模块时需要为该模块编写一个驱动模块和若干个桩（stub）模块。驱动模块用来调用被测模块，它接收测试者提供的测试数据，并把这些数据传送给被测模块，然后从被测模块接收测试结果，并以某种可以看见的方式（例如显示或打印）将测试结果返回给测试

者。桩模块用来模拟被测模块所调用的子模块，它接受被测模块的调用，检验调用参数，并以尽可能简单的操作模拟被调用的子程序模块功能，把结果送回被测模块。顶层模块测试时不需要驱动模块，底层模块测试时不需要桩模块。

模块的内聚程度高可以简化单元测试过程。如果每个模块只完成一种功能，则需要的测试方案数目将明显减少，模块中的错误也更容易预测和发现。

2．集成测试

集成测试（integration testing），也称组装测试，它是对由各模块组装而成的程序进行测试，主要目标是发现模块间的接口和通信问题。例如，数据穿过接口可能丢失，一个模块对另一个模块可能由于疏忽而造成有害影响，把子功能组合起来可能不产生预期的主功能，个别看来是可以接受的误差可能积累到不能接受的程度，全程数据结构可能有问题等。集成测试主要发现设计阶段产生的错误，集成测试计划应该在概要设计阶段制订。

集成的方式可分为非渐增式和渐增式。

非渐增式集成是先测试所有的模块，然后一下子把所有这些模块集成到一起，并把庞大的程序作为一个整体来测试。这种测试方法的出发点是可以“一步到位”，但测试者面对众多的错误现象，往往难以分清哪些是“真正的”错误，哪些是由其他错误引起的“假性错误”，诊断定位和改正错误也十分困难。非渐增式集成只适合一些非常小的软件。

渐增式集成是将单元测试和集成测试合并到一起，它根据模块结构图，按某种次序选一个尚未测试的模块，把它同已经测试好的模块组合在一起进行测试，每次增加一个模块，直到所有模块被集成在程序中。这种测试方法比较容易定位和改正错误，目前在进行集成测试时已普遍采用渐增式集成。

渐增式集成又可分为自顶向下集成和自底向上集成。自顶向下集成先测试上层模块，再测试下层模块。由于测试下层模块时它的上层模块已测试过，所以不必另外编写驱动模块。自底向上集成先测试下层模块，再测试上层模块。同样，由于测试上层模块时它的下层模块已测试过，所以不必另外编写桩模块。这两种集成方法各有利弊，一种方法的优点恰好对应于另一种方法的缺点，实际测试时可根据软件特点及进度安排灵活选用最适当的方法，也可将两种方法混合使用。

3.系统测试

系统测试是软件测试中的最后的、最完整的测试，它是在单元测试和集成测试的基础上进行的，它从全局来考察软件系统的功能和性能要求。系统测试计划应该在需求分析阶段制订。

通常，系统测试包括确认测试和验收测试。

确认测试，主要依据软件需求说明书检查软件的功能、性能及其他特征是否与用户的需求一致。

软件配置复查是确认测试的另一项重要内容。复查的目的是保证软件配置的所有成分都已齐全，质量符合要求，文档与程序完全一致，具有完成软件维护所必需的细节。

如果一个软件是为某个客户定制的，最后还要由该客户来实施验收测试，以便确认其所有需求是否都已得到满足。由于软件系统的复杂性，在实际工作中，验收测试可能会持续到用户实际使用该软件之后的相当长的一段时间。

如果一个软件是作为产品被许多客户使用的，不可能也没必要由每个客户进行验收测试。绝大多数软件开发商都使用被称为 (Alpha)测试和 (Beta)测试的过程，来发现那些看起来只有最终用户才能发现的错误。

a 测试由用户在开发者的场所进行，并且在开发者的指导下进行测试。开发者负责记录发现的错误和使用中遇到的问题。也就是说， 测试是在“受控的”环境中进行的。

b 测试是在一个或多个用户的现场由该软件的最终用户实施的，开发者通常不在现场，用户负责记录发现的错误和使用中遇到的问题并把这些问题报告给开发者。也就是说，测试是在“不受控的”环境中进行的。

经过系统测试之后的软件通常就可以交付使用了。

11.1.2 白盒测试和黑盒测试

从测试方法上分，软件测试可分为白盒测试和黑盒测试。

1．白盒测试

白盒测试，又称结构测试，主要用于单元测试阶段。它的前提是可以把程序看成装在一个透明的白箱子里，测试者完全知道程序的结构和处理算法。这种方法按照程序内部逻辑设计测试用例，检测程序中的主要执行通路是否都能按预定要求正常工作。

白盒测试根据软件的内部逻辑设计测试用例，常用的技术是逻辑覆盖，即考察用测试数据运行被测程序时对程序逻辑的覆盖程度。主要的覆盖标准有 6 种：语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、组合条件覆盖和路径覆盖。

（1）语句覆盖。语句覆盖是指选择足够多的测试用例，使得运行这些测试用例时，被测程序的每个语句至少执行一次。

很显然，语句覆盖是一种很弱的覆盖标准。

（2）判定覆盖。判定覆盖又称分支覆盖，它的含义是，不仅每个语句至少执行一次，而且每个判定的每种可能的结果（分支）都至少执行一次。判定覆盖比语句覆盖强，但对程序逻辑的覆盖程度仍然不高。

（3）条件覆盖。条件覆盖的含义是，不仅每个语句至少执行一次，而且使判定表达式中的每个条件都取得各种可能的结果。

条件覆盖不一定包含判定覆盖，判定覆盖也不一定包含条件覆盖。

（4）判定/条件覆盖。同时满足判定覆盖和条件覆盖的逻辑覆盖称为判定/条件覆盖。它的含义是，选取足够的测试用例，使得判定表达式中每个条件的所有可能结果至少出现一次，而且每个判定本身的所有可能结果也至少出现一次。

（5）条件组合覆盖。条件组合覆盖的含义是，选取足够的测试用例，使得每个判定表达式中条件结果的所有可能组合至少出现一次。

显然，满足条件组合覆盖的测试用例，也一定满足判定/条件覆盖。因此，条件组合覆盖是上述 5 种覆盖标准中最强的一种。然而，条件组合覆盖还不能保证程序中所有可能的路径都至少经过一次。

（6）路径覆盖。路径覆盖的含义是，选取足够的测试用例，使得程序的每条可能执行到的路径都至少经过一次（如果程序中有环路，则要求每条环路路径至少经过一次）。

路径覆盖实际上考虑了程序中各种判定结果的所有可能组合，因此是一种较强的覆盖标准。但路径覆盖并未考虑判定中的条件结果的组合，并不能代替条件覆盖和条件组合覆盖。

2．黑盒测试

黑盒测试，又称功能测试，主要用于集成测试和确认测试阶段。它把软件看作一个不透明的黑箱子，完全不考虑（或不了解）软件的内部结构和处理算法，它只检查软件功能是否能按照软件需求说明书的要求正常使用，软件是否能适当地接收输入数据并产生正确的输出信息，软件运行过程中能否保持外部信息（例如文件和数据库）的完整性等。

黑盒测试根据软件需求说明书所规定的功能来设计测试用例，它不考虑软件的内部结构和处理算法。

常用的黑盒测试技术包括等价类划分、边值分析、错误推测和因果图等。

（1）等价类划分。在设计测试用例时，等价类划分是用得最多的一种黑盒测试方法。所谓等价类就是某个输入域的集合，对于一个等价类中的输入值来说，它们揭示程序中错误的作用是等效的。也就是说，如果等价类中的一个输入数据能检测出一个错误，那么等价类

中的其他输入数据也能检测出同一个错误；反之，如果等价类中的一个输入数据不能检测出某个错误，那么等价类中的其他输入数据也不能检测出这一错误（除非这个等价类的某个子集还属于另一等价类）。

如果一个等价类内的数据是符合（软件需求说明书）要求的、合理的数据，则称这个等价类为有效等价类。有效等价类主要用来检验软件是否实现了软件需求说明书中规定的功能。

如果一个等价类内的数据是不符合（软件需求说明书）要求的、不合理或非法的数据，则称这个等价类为无效等价类。无效等价类主要用来检验软件的容错性。

黑盒测试中，利用等价类划分方法设计测试用例的步骤是：

① 根据软件的功能说明，对每一个输入条件确定若干个有效等价类和若干个无效等价类，并为每个有效等价类和无效等价类编号。

② 设计一个测试用例，使其覆盖尽可能多的尚未被覆盖的有效等价类。重复这一步，直至所有的有效等价类均被覆盖。

③ 设计一个测试用例，使其覆盖一个尚未被覆盖的无效等价类。重复这一步，直至所有的无效等价类均被覆盖。

希赛教育专家提示：无效等价类是用来测试非正常的输入数据的，因此每个无效等价类都有可能查出软件中的错误，所以要为每个无效等价类设计一个测试用例。

（2）边值分析。经验表明，软件在处理边界情况时最容易出错。设计一些测试用例，使软件恰好运行在边界附近，暴露出软件错误的可能性会更大一些。

通常，每一个等价类的边界，都应该着重测试，选取的测试数据应该恰好等于、稍小于或稍大于边界值。

将等价类划分法和边值分析法结合使用，更有可能发现软件中的错误。

（3）错误推测。使用等价类划分和边值分析技术，有助于设计出具有代表性的、容易暴露软件错误的测试方案。但是，不同类型不同特定的软件通常又有一些特殊的容易出错的地方。错误推测法主要依靠测试人员的经验和直觉，从各种可能的测试方案中选出一些最可能引起程序出错的方案。

（4）因果图。因果图法是根据输入条件与输出结果之间的因果关系来设计测试用例的，它首先检查输入条件的各种组合情况，并找出输出结果对输入条件的依赖关系，然后为每种输出条件的组合设计测试用例。

11.1.3 缺陷的分类和级别

根据 IEEE 标准和 Paul C.Jorgensen 的教科书，软件测试中所发现的错误（缺陷）主要包括以下几类：

（1）输入/输出错误。包括不接收正确的输入、接收不正确的输入、描述有错或遗漏、参数有错或遗漏、输出结果有误、输出格式有误、输出时间有误、结果不一致、遗漏结果、不合逻辑的结果、拼写/语法错误、修饰词错误。

（2）逻辑错误。包括遗漏情况、重复情况、极端条件出错、解释有误、遗漏条件、外部条件有错、错误变量的测试、不正确的循环迭代、错误的操作符。

（3）计算错误。包括不正确的算法、遗漏计算、不正确的操作数、不正确的操作、括号错误、精度不够错误的内置函数。

（4）接口错误。包括不正确的中断处理、I/O 时序有错、调用了错误的过程、调用了不存在的过程、参数不匹配、不兼容的类型、过量的包含。

（5）数据错误。包括不正确的初始化、不正确的存储/访问、错误的标识/索引值、不正确的打包/拆包、使用了错误的变量、错误的数据引用、缩放数据范围或单位错误、不正确的数据维数、不正确的下标、不正确的类型、不正确的数据范围、数据超出限制、数据溢出、不一致的数据。

根据错误（缺陷）后果的严重程度，Beizer 将错误（缺陷）分为 10 级：

（1）轻微（例如，界面文字有个别的错别字，但不影响理解）。

（2）中等（例如，界面文字错误可能误导操作者）。

（3）使人不悦（例如，数字串被断开）。

（4）影响使用（例如，有些交易没有处理）。

（5）严重（例如，丢失交易）。

（6）非常严重（例如，不正确的交易处理）。

（7）极为严重（例如，经常出现不正确的交易处理）。

（8）无法容忍（例如，数据库遭到破坏）。

（9）灾难性（例如，系统无法工作）。

（10）传染性（例如，可导致其他系统无法工作）。

11.1.4 调试

调试又称为排错，调试与成功的测试形影相随。测试成功的标志是发现了错误。根据错误迹象确定错误的原因和准确位置并加以改正，主要依靠排错技术。

调试是一个相当艰苦的过程，究其原因除了开发人员心理方面的障碍外，还因为隐藏在程序中的错误具有下列特殊的性质：

（1）错误的外部征兆远离引起错误的内部原因，对于高度耦合的程序结构此类现象更为严重。

（2）纠正一个错误造成了另一错误现象（暂时）的消失。

（3）某些错误征兆只是假象。

（4）因操作人员一时疏忽造成的某些错误征兆不易追踪。

（5）错误是由于分时而不是程序引起的。

（6）输入条件难以精确地再构造（例如，某些实时应用的输入次序不确定）。

（7）错误征兆时有时无，此现象对嵌入式系统尤其普遍。

（8）错误是由于把任务分布在若干台不同处理机上运行而造成的。

在软件排错过程中，可能遇到大大小小、形形色色的问题，随着问题的增多，排错人员的压力也随之增大，过分的紧张致使开发人员在排除一个问题的同时又引入更多的新问题。

尽管排错不是一门好学的技术，但还是有若干行之有效的方法和策略的，常用的排错策略分为三类：

（1）原始类。原始类排错方法是最常用也是最低效的方法，只有在万般无奈的情况下才使用它，主要思想是“通过计算机找错”。例如输出存储器、寄存器的内容，在程序安排若干输出语句等，凭借大量的现场信息，从中找到出错误的线索。虽然最终也能成功，但难免要耗费大量的时间和精力。

（2）回溯类。回溯法能成功地用于程序的排错。方法是从出现错误征兆处开始，人工地沿控制流程往回追踪，直至发现出错的根源，不幸的是程序变大后，可能的回溯路线显著增加，以致人工进行完全回溯可望而不可即。

（3）排除类。排除法基于归纳和演绎原理，采用“分治”的概念，首先分析与错误出现有关的所有数据，假想一个错误原因，用这些数据证明或反驳它；或者一次列出所有可能的原因，通过测试一一排除。只要某次测试结果说明某种假设已呈现端倪，则立即精化数据，乘胜追击。

11.2 评审方法

根据 IEEE 1028 的定义，评审是对软件元素或者项目状态的一种评估手段，以确定其是否与计划的结果保持一致，并使其得到改进。

狭义的“软件评审”通常指软件文档和源程序的评审。广义的“软件评审”还包括与软件测试相结合的评审及管理评审。软件评审包括软件需求评审、概要设计评审、详细设计评审、软件验证和确认评审、功能检查、物理检查、综合检查和管理评审。

（1）软件需求评审。在软件需求分析结束后必须进行软件需求评审（ software requirements review），以确保在软件需求说明书中所规定的各项需求的合适性。

（2）概要设计评审。在软件概要设计结束后必须进行概要设计评审（preliminary design review），以评价软件设计说明书中所描述的软件概要设计在总体结构、外部接口、主要部件功能分配、全局数据结构以各主要部件之间的接口等方面的合适性。

（3）详细设计评审。在软件详细设计结束后必须进行详细设计评审（detailed design review），以评价软件设计说明书中所描述的软件详细设计在每一个基本部件的功能、算法和过程描述等方面的合适性。

（4）软件验证和确认评审。在软件验证与确认计划完成后必须进行软件验证与确认评审（software verification and validation review），以评价软件验证与确认计划中所规定的验证与确认方法的合适性与完整性。

（5）功能检查。在软件释放前，要对软件进行功能检查（functional audit），以验证所开发的软件已经满足在软件需求说明书中规定的所有需求。

（6）物理检查。在软件验收前，要对软件进行物理检查（physical audit），以验证程序和文档已经一致并已做好了交付的准备。

（7）综合检查。在软件验收时，要允许用户或用户所委托的专家对所要验收的软件进行设计抽样的综合检查（comprehensive audit），以验证代码和设计文档的一致性、接口规格说明的一致性（硬件和软件）、设计实现和功能需求的一致性、功能需求和测试描述的一致性。

（8）管理评审。要对计划的执行情况定期（或按阶段）进行管理评审（management reviews），这些评审必须由独立于被评审单位的机构或授权的第三方主持进行。

在评审过程中，以下几点值得注意：

（1）不应以测试代替评审。许多缺陷是在早期阶段引入的，缺陷发现得越晚，纠正费

用越高。而且每个进入下一步骤的缺陷都可能引起下一步骤中的多个缺陷，导致消除成本的剧增。早期阶段可以进行评审，但是无法进行测试，评审的目的就是减少泄漏到测试阶段的缺陷。评审可能会花很多时间，但在测试中节省了时间。而且，测试也不能发现某些特定类型的缺陷（例如违犯编程规范）。

（2）评审人员应关注产品而不应评论开发人员。评审的主要目的是发现产品中的问题，而不是根据产品来评价开发人员的水平。但是往往会出现把产品质量和开发人员水平联系起来的事情，于是评审变了味，变成了“批斗大会”，极大地打击了开发人员的自尊心，以至于严重地影响了评审的效果。

（3）评审人员应关注于实质性问题。评审中经常会出现这样的现象，评审人员过多地关注于一些非实质性的问题，例如，文档的格式、措词，而不是产品的设计。出现这种情况，可能的原因有：没有选择合适的人参加评审，评审人员对评审对象没有足够的了解，无法发现深层次的问题。

（4）评审会议不应变为问题解决方案讨论会。评审会议主要的目的是发现问题，而不是解决问题，问题的解决是评审会议之后需要做的事情。但是，由于开发人员对技术的追求，评审会议往往变成了问题研讨会，大量地占用了评审会议的时间，导致大量评审内容被忽略，留下无数的隐患。

（5）评审应被安排进入项目计划。参与评审需要投入大量的时间和精力，应该被安排进入项目计划中。但在现实中，评审往往变成了临时安排的工作。如此一来，出现评审人员对评审对象不了解的情况也就不足为奇了。

（6）评审参与者应了解整个评审过程。如果评审参与者不了解整个的评审过程，就会有一种自然的抗拒情绪，因为大家看不到做这件事情的效果，感觉很迷茫，这样会严重地影响大家参与评审的积极性。

（7）评审人员事先应对评审材料充分了解。任何一份评审材料都是他人智慧和心血的结晶，需要花足够的时间去了解、熟悉和思考。只有这样，才能在评审会议上发现有价值的深层次问题。在很多的评审中，评审人员因为各种的原因，在评审会议之前对评审材料没有足够的了解，于是出现了评审会议变成了技术报告的怪现象。

（8）应重视评审的组织工作。在组织评审的过程中，很多人不太注意细节。例如，会议时间的设定，会议的通知，会议场所的选择，会场环境的布置，会议设施的提供，会议上气氛的调节和控制等，而实际上这样的细节会大大影响评审会议的效果。

11.3 验证与确认

验证与确认都是确定软件产品是否满足其预期要求和条件的过程。验证可适用于分析、设计、编码、测试和评审等众多的过程，而确认通常用于验收过程。

1.验证

软件项目的验证一般应包括合同验证、过程验证、需求验证、设计验证、编码验证、集成验证和文档验证。

（1）合同验证。应根据下列准则验证合同：

供方具有满足需求的能力。

需求是一致的并覆盖了用户的需要。

为处理需求变更和升级问题规定了适当的规程。

规定了各方之间的接口及其合作规程与范围，包括所有权、许可权、版权和保密要求。

按照需求规定了验收准则和规程。

（2）过程验证。应根据下列准则验证过程：

项目是适当的、及时的。

为项目选择的过程是适当的并满足合同要求的。

用于项目过程的标准、规程和环境是适当的。

根据合同要求为项目配备了经过培训的人员。

（3）需求验证。应根据下列准则验证需求：

需求是明确的、一致的、无歧义的。

需求是可行的。

需求是可测试的。

（4）设计验证。应根据下列准则验证设计：

设计是正确的，是可以实现需求的。

可以从需求导出设计，可以从设计追踪需求。

（5）编码验证。应根据下列准则验证编码：

编码是正确的，可以实现设计和需求。

可以从设计导出编码，可以从编码追踪设计。

（6）集成验证。应根据下列准则验证集成：

每一个软件项的软件部件和软件单元已完整、正确地集成到软件项中。

系统的硬件项、软件项和人工操作已完整、正确地集成到系统中。

（7）文档验证。应根据下列准则验证文档：

文档是充分的、完备的、一致的。

文档制订是及时的。

文档配置管理遵循了规定的规程。

2．确认

如果项目需要开展确认工作，应建立一个确认过程，以确认软件产品满足其预期用途。确认可以是组织内部的，也可以由独立的第三方实施。

一般来讲，确认过程应包括下列任务：

（1）编写测试需求、测试用例和测试规程。

（2）确保这些测试需求、测试用例和测试规程可以反映软件产品的预期用途。

（3）执行测试。

（4）确认软件产品满足其预期用途。

11.4 测试自动化

软件测试的工作量很大，但测试却极有可能应用计算机进行相当一部分自动化的工作，原因是测试的许多操作是重复性的、非智力创造性的、需要细致注意力的工作，而计算机就最适合于代替人类去完成这些任务。测试自动化会对整个开发工作的质量、成本和周期带来非常明显的效果。

一些适于考虑进行自动化的测试工作为：

（1）测试用例的生成（包括测试输入、标准输出、测试操作指令等）。

（2）测试的执行控制（包括单机与网络多机分布运行、夜间及假日运行、测试用例调用控制、测试对象、范围、版本控制等）。

（3）测试结果与标准输出的对比。

（4）不吻合的测试结果的分析、记录、分类和通报。

（5）总测试状况的统计，报表的产生。测试自动化与软件配置管理是密不可分的，与测试有关的资源都应在配置管理中统一考虑。

11.5 面向对象的测试

传统的软件测试策略是从“小型测试”开始，逐步走向“大型测试”。即从单元测试开始，然后进入集成测试，最后是系统测试。

面向对象程序的结构不再是传统的功能模块结构，作为一个整体，原有集成测试所要求的逐步地将开发的模块搭建在一起进行测试的方法已成为不可能。而且，面向对象软件抛弃了传统的开发模式，对每个开发阶段都有不同以往的要求和结果，已经不可能用功能细化的观点来检测面向对象分析和设计的结果。因此，传统的测试模型对面向对象软件已经不再适用。

1．面向对象测试模型

面向对象的开发模型突破了传统的瀑布模型，将开发分为 OOA、OOD 和 OOP 三个阶段。针对这种开发模型，结合传统的测试步骤的划分，可以把面向对象的软件测试分为：面向对象分析的测试、面向对象设计的测试、面向对象编程的测试、面向对象的单元测试、面向对象的集成测试和面向对象的系统测试。

2.面向对象分析的测试

传统的面向过程分析是一个功能分解的过程，是把一个系统看成可以分解的功能的集合。功能分解分析法的着眼点在于一个系统需要什么样的信息处理方法和过程，以过程的抽象来对待系统的需要。而面向对象的分析直接映射问题空间，将问题空间中的实例抽象为对象，用对象的结构反映问题空间的复杂实例和复杂关系，用属性和操作表示实例的特性和行为。OOA 的结果是为后面阶段类的选定和实现、类层次结构的组织和实现提供平台。因此，对 OOA 的测试，应从以下方面考虑：

（1）对认定的对象的测试。

（2）对认定的结构的测试。

（3）对认定的主题的测试。

（4）对定义的属性和实例关联的测试。

（5）对定义的服务和消息关联的测试。

3．面向对象设计的测试

传统的结构化设计方法，采用面向作业的思路，它把系统分解以后，提出一组作业，这些作业是以过程实现系统的基础构造，把问题域的分析转化为求解域的设计，分析的结果是设计阶段的输入。而 OOD 以 OOA 为基础归纳出类，并建立类结构或进一步构造成类库，

实现分析结果对问题空间的抽象。由此可见，OOD 不是 OOA 的另一思维方式，而是 OOA 的进一步细化和更高层的抽象，OOD 与 OOA 的界限通常是难以严格区分的。OOD 确定类和类结构不仅是满足当前需求分析的要求，更重要的是通过重新组合或加以适当的补充，能够方便地实现功能的重用和扩充，以不断适应用户的要求。因此，对 OOD 的测试，应从如下三方面考虑：

（1）对认定的类的测试。

（2）对构造的类层次结构的测试。

（3）对类库的支持的测试。

4．面向对象编程的测试

典型的面向对象程序具有继承、封装和多态的新特性，这使得传统的测试策略必须有所改变。封装是对数据的隐藏，外界只能通过被提供的操作来访问或修改数据，这样降低了数据被任意修改和读写的可能性，降低了传统程序中对数据非法操作的测试。继承是面向对象程序的重要特点，继承使得代码的重用性提高，同时也使错误传播的概率提高。多态使得面向对象程序对外呈现出强大的处理能力，但同时却使得程序内 “同一”函数的行为复杂化，测试时不得不考虑对于不同类型参数具体执行的代码和产生的行为。

面向对象程序是把功能的实现分布在类中。能正确实现功能的类，通过消息传递来协同实现设计要求的功能。因此，在 OOP 阶段，忽略类功能实现的细则，将测试的焦点集中在类功能的实现和相应的面向对象程序风格，主要体现为以下两个方面。

（1）数据成员是否满足数据封装的要求。

（2）类是否实现了要求的功能。

5．面向对象的单元测试

传统的单元测试的对象是软件设计的最小单位——模块，测试依据是详细设计说明书。单元测试应对模块内所有重要的控制路径设计测试用例，以便发现模块内部的错误。单元测试多采用白盒测试技术，系统内多个模块可以并行地进行测试。

对于面向对象的软件，单元的概念发生了变化。每个类和类的实例（对象）封装了属性（数据）和操纵这些数据的操作，而不是个体的模块，最小的可测试单位变成了封装的类或对象。因此，单元测试的意义发生了较大变化。不再孤立地测试单个操作，而是将操作作为类的一部分。

6．面向对象的集成测试

传统的集成测试，有两种典型的集成策略：

（1）自顶向下集成，它从主控模块开始，按照软件的控制层次结构，以深度优先或广度优先的策略，逐步把各个模块集成在一起。

（2）自底向上集成，从“原子”模块（即软件结构最低层的模块）开始组装测试。

面向对象的软件没有层次控制结构，传统的自顶向下和自底向上集成策略已无太大意义。此外，一次集成一个操作到类中（传统的增量集成方法）通常是不可能的。对 OO 软件的集成测试也有两种不同策略：第一种称为基于线程的测试，集成系统的一个输入或事件所需的一组类，每个线程被集成并分别测试，并使用回归测试以保证没有产生副作用。第二种称为基于使用的测试，首先测试那些几乎不使用其他类的类（称为独立类）并开始构造系统，在独立类测试完成后，下一层的使用独立类的类（称为依赖类）被测试。这个依赖类层次的测试序列一直持续到构造完整个系统。

7．面向对象的系统测试

通过单元测试和集成测试，仅能保证软件开发的功能得以实现。但不能确认在实际运行时，它是否满足用户的需要。为此，对完成开发的软件必须经过规范的系统测试。系统测试应该尽量搭建与用户实际使用环境相同的测试平台，应该保证被测系统的完整性，对临时没有的系统设备部件，也应有相应的模拟手段。系统测试时，应该参考 OOA 分析的结果，对应描述的对象、属性和各种服务，检测软件是否能够完全“再现”问题空间。系统测试不仅是检测软件的整体行为表现，从另一个侧面看，也是对软件开发设计的再确认。

希赛教育专家提示：面向对象测试的整体目标是以最小的工作量发现最多的错误，与传统软件测试的目标是一致的，但 OO 测试的策略与传统测试有很大不同。这种不同主要体现在两个方面，第一，测试的焦点从过程构件（模块）移向了类；第二，测试的视角扩大到了分析和设计模型。

第 12 章：嵌入式系统设计

随着计算机技术、微电子技术、通信技术以及集成电路技术的发展，嵌入式技术逐渐发展和成熟起来。嵌入式系统的应用日益广泛，并 在数量上远远超越了通用计算机系统，成为了计算机技术和计算机应用领域的一个重要组成部分。

本章主要讨论嵌入式系统的基本知识与嵌入式系统的开发设计两部分内容，主要包括嵌入式系统的概念、软/硬件组成与基本架构、嵌入式操作系统和嵌入式数据库系统、网络系

统以及窗口系统等核心支撑软件系统的基本原理和技术，最后介绍嵌入式系统的开发设计。

12.1 嵌入式系统概论

嵌入式系统是一种以应用为中心，以计算机技术为基础，可以适应不同应用对功能、可靠性、成本、体积、功耗等方面的要求，集可配置，可裁减的软、硬件于一体的专用计算机系统。它具有很强的灵活性，主要由嵌入式硬件平台、相关支撑硬件、嵌入式操作系统、支撑软件以及应用软件组成。其中，“嵌入性”、“专用性”与“计算机系统”是嵌入式系统的三个基本的核心要素，具体来讲：

嵌入性：指计算机计算机嵌入到对象系统中，且满足对象系统的环境要求，如物理环境（小型）、电气/气氛环境（可靠）、成本（价廉）等要求。

专用性：指软、硬件的裁剪性，满足对象要求的最小软、硬件配置等。

计算机系统：指嵌入式系统必须是一个能满足对象系统控制要求的计算机系统。

归纳起来，典型的嵌入式系统具有以下特点：

（1）系统专用性强。嵌入式系统是针对具体应用的专门系统。它的个性化很强，软件和硬件结合紧密。一般要针对硬件进行软件的开发和移植，根据硬件的变化和增减对软件进行修改。由于嵌入式系统总是用来完成某一特定任务，整个系统与具体应用有机地结合在一起，升级换代也以更换整个产品的方式进行，因此，一个嵌入式产品一旦进入市场，一般具有较长的生命周期。

（2）系统实时性强。嵌入式系统中有相当一部分系统要求对外来事件在限定的时间内及时做出响应，具有实时性。

（3）软、硬件依赖性强。嵌入式系统的专用性决定了其软、硬件之间具有很强的互相依赖性，两者必须协同设计，以达到共同实现预定功能的目的，并满足性能、成本和可靠性等方面的严格要求。

（4）处理器专用。嵌入式系统的处理器与通用计算机的处理器之间最大的不同之处在于，嵌入式系统的处理器一般是为某一特定目的和应用而专门设计的。通常具有功耗低、体积小、集成度高等优点，能够把许多在通用计算机上需要由板卡完成的任务和功能集成到芯片内部，从而有利于嵌入式系统的小型化和移动能力的增强。

（5）多种技术紧密结合。嵌入式系统通常是计算机技术、半导体技术、电力电子技术及机械技术与各行业的具体应用相结合的产物。通用计算机技术也离不开这些技术，但它们

相互结合的紧密程度不及嵌入式系统。

（6）系统透明性。嵌入式系统在形态上与通用计算机系统差异甚大。它的输入设备往往不是常见的鼠标和键盘之类的设备，甚至没有输出装置，用户可能根本感觉不到它所使用的设备中有嵌入式计算机系统的存在，即使知道也不必关心这个嵌入式计算机系统的相关情况。

（7）系统资源受限。嵌入式系统为了达到结构紧凑、可靠性高及降低系统成本的目的，其存储容量、输入/输出设备的数量和处理器的处理能力都比较有限。

12.2 嵌入式系统的组成

嵌入式系统一般都由软件和硬件两个部分组成，其中嵌入式处理器、存储器和外部设备构成整个系统的硬件基础。嵌入式系统的软件部分可以分为 3 个层次：系统软件、应用支撑软件和应用软件。其中，系统软件和支撑软件是基础，应用软件则是最能体现整个嵌入式系统的特点和功能的部分。

12.2.1 硬件架构

图 12-1 是一个嵌入式系统的基本硬件架构。微处理器是整个嵌入式系统的核心，负责控制系统的执行。外部设备是嵌入式系统同外界交互的通道，常见的外部设备有 Flash 存储器、键盘、输入笔、触摸屏、液晶显示器等输入/输出设备，在很多嵌入式系统中还有与系统用途紧密相关的各种专用外设。嵌入式系统中经常使用的存储器有 3 种类型：RAM、ROM（Read-Only Memory，只读内存）和混合存储器。系统的存储器用于存放系统的程序代码、数据和系统运行的结果。

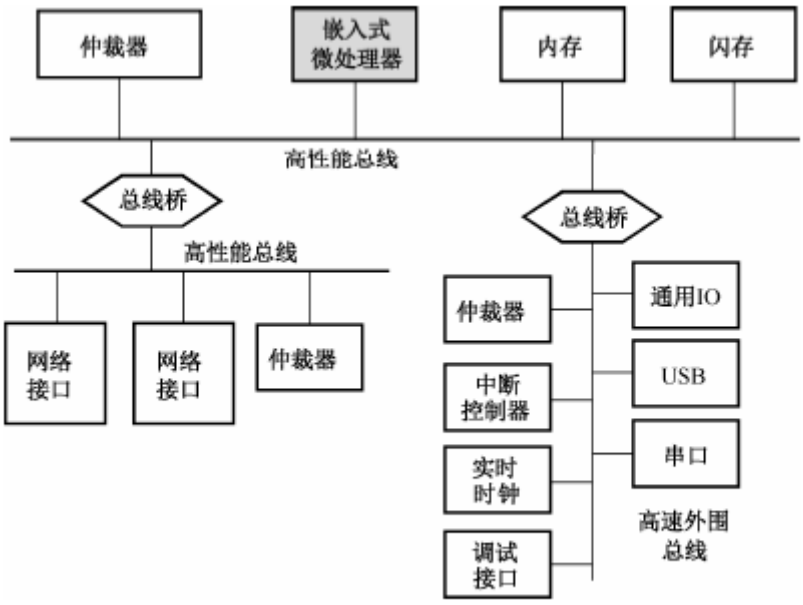


图 12-1　嵌入式硬件平台的系统架构

1．嵌入式处理器

嵌入式系统的核心部件是各种类型的嵌入式处理器。据不完全统计，目前世界上嵌入式处理器的种类已经超过了 1000 种，比较流行的也有 30 几个系列。根据目前的使用情况，嵌入式处理器可以分为如下几类：

（1）嵌入式微处理器。嵌入式微处理器（Embedded Micro Processing Unit，EMPU）是由通用计算机中的 CPU 演变而来。嵌入式微处理器在功能上跟普通的微处理器基本一致，但是它具有体积小、功耗低、质量轻、成本低及可靠性高的优点。通常，嵌入式微处理器和 ROM、RAM、总线接口及外设接口等部件安装在一块电路板上，称为单板计算机。目前，主要的嵌入式微处理器有 AM186/88、386EX、SC-400、POWER PC、MIPS 及 ARM 等系列。

（2）嵌入式微控制器。嵌入式微控制器（Embedded Micro Controlling Unit，EMCU）又称为单片机，就是整个计算机系统都集成到一块芯片中。嵌入式微控制器一般以某一种微处理器内核为核心，芯片内部集成有 ROM/EPROM/E2PROM、RAM、总线、总线逻辑、定时器/计数器、WatchDog（监督定时器）、并口/串口、数模/模数转换器、闪存等必要外设。与嵌入式微处理器相比，嵌入式微控制器的最大特点是单片化，因而体积更小，功耗和成本更低，可靠性更高。

目前，嵌入式微控制器的品种和数量最多，约占嵌入式系统市场份额的 70%。比较有代表性的通用系列有：8051 系列、MCS-96/196/296、C166/167、MC68HC05/11/12/16 等。还有许多半通用系列，如支持 UBS 接口的 MCU 8XC930/931、C540、C541 以及用于支持 I2C、

现场总线等各种微控制器。

（3 ）嵌入式数字信号处理器。嵌入式数字信号处理器（Embedded Digital Signal Processor，EDSP）是一种专门用于信号处理的处理器，DSP 芯片内部采用程序和数据分开的哈佛结构，具有专门的硬件乘法器，广泛采用流水线操作，提供特殊的 DSP 指令，可以用来快速实现各种数字信号的处理算法。目前，数字信号处理器在嵌入式系统中使用非常广泛，如数字滤波、快速傅立叶变换及频谱分析等。同时，嵌入式系统的智能化也是推动嵌入式 DSP 发展的一个动力，如各种带有智能逻辑的消费类产品、生物信息识别终端、带有加密/解密算法的设备、实时语音压缩和解压系统以及虚拟现实显示装置等，这类系统上的智能化算法一般运算量都比较大，这恰好可以充分发挥数字信号处理器的长处。

（4）嵌入式片上系统。嵌入式片上系统（Embedded System On Chip）是一种在一块芯片上集成很多功能模块的复杂系统，如微处理器内核、RAM、USB、IEEE 1394、Bluetooth 等。以往这些单元按照各自的功能做成一个个独立的芯片，并通过电路板与其他单元组成一个系统。现在将这些本来在电路板上的单元都集成到一个芯片中，构成一个嵌入式片上系统，从而大幅度缩小了系统的体积，降低了系统的复杂度，增强了系统的可靠性。在大量生产时，生产成本也远远低于单元部件组成的电路板系统。嵌入式片上系统可以分为通用片上系统和专用片上系统两类。通用类的主要产品有 Siemens 的 Trocore、Motorola 的 M-Core、某些 ARM 系列的器件等。专用类的嵌入式片上系统一般是针对某一个或某些系统而设计的。具有代表性的产品有 Philips 的 Smart XA，它将 XA 单片机的内核和支持超过 2048 位复杂 RSA 算法的 CCU 单元制作在一个芯片上，形成一个可加载 Java 或 C 的专用嵌入式片上系统，可用于网络安全等方面。

2．总线

总线是连接计算机系统内部各个部件的共享高速通路，自 20 世纪 70 年代以来，工业界相继出现了多种总线标准，很多总线技术在嵌入式系统领域得到了广泛的应用。

嵌入式系统的总线一般分为片内总线和片外总线。片内总线是指嵌入式微处理器内的 CPU 与片内其他部件连接的总线；片外总线是指总线控制器集成在微处理器内部或外部芯片上的用于连接外部设备的总线。

（1）AMBA 总线。AMBA（Advanced Microcontroller Bus Archiecture，先进微控制器总线架构）是 ARM 公司研发的一种总线规范，该总线规范独立于处理器和制造工艺技术，增强了各种应用中外设和系统单元的可重用性，它提供将 RISC 处理器与 IP 核集成的机制。该规范定义了三种总线：

先进性能总线（Advanced High-performance Bus，AHB）。AHB 由主模块、从模块和基础结构三部分组成，整个 AHB 总线上的传输都由主模块发起，从模块响应。基础结构包括：仲裁器、主从模块多路选择器、译码器、名义主模块、名义从模块等。AHB 系统具有时钟边沿触发、无三态、分帧传输等特性。AHB 也支持复杂的事务处理，如突发传送、主单元重试、流水线操作以及分批事务处理等。

先进系统总线（Advanced System Bus，ASB）。ASB 用于高性能模块的互连，支持突发数据传输模式，较老的总线格式，逐步由 AHB 总线所替代。

先进外设总线（Advanced Peripheral Bus，APB）。APB 主要用于连接低带宽外围设备，其总线结构只有唯一的主模块，即 APB 桥，它不需要仲裁器以及响应/确认信号，以最低功耗为原则进行设计，具有总是两周期传输、无等待周期和响应信号的特点。

（2）PCI 总线。外围构件互连总线（Peripheral Component Interconnect，PCI）规范先后经历了 1.0 版本、2.0 版本和 2.1 版本等一系列规范。PCI 总线是地址、数据复用的高性能 32 位与 64 位总线，是微处理器与外围设备互连的机构，它规定了互连协议、电气、机械以及配置空间的标准。PCI 是不依赖于具体处理器的局部总线，从结构上看，PCI 是在微处理器和原来的系统总线之间加入的一级总线，由一个桥接电路负责管理，实现上下接口和协调数据传送，管理器提供了信号缓冲，使多种外设能够在高时钟频率下保持高性能。PCI 总线支持主控技术，允许智能设备在需要时获得总线控制权，以加速数据传输。

为了将 PCI 总线规范应用到工业控制计算机中，1995 年，推出了 Compact PCI 规范，并相继推出了 PCI-PCI Bridge 规范、Computer TelephonyTDM 规范和用户定义 I/O 引脚分配规范等。CPCI 总线规范有机地结合了 PCI 总线电气规范的高性能和欧洲卡结构的高可靠。目前，CPCI 总线已经在嵌入式系统、工业控制计算机等高端系统中得到了广泛的应用，并逐步替代了 VME 和 MultiBUS 总线。

（3）Avalon 总线。Avalon 总线是 Altera 公司设计的用于可编程片上系统（System on Programmable Chip，SOPC）中，连接片上处理器和其他 IP 模块的一种简单总线协议，规定了主部件和从部件之间进行连接的端口和通信时序。

作为总结，表 12-1 对比了几种嵌入式总线技术的主要特点。

表 12-1　几种嵌入式总线技术的主要特点

总线类型	主要特点
AMBA 总线	带宽高：采用地址与数据分离的流水线操作；支持固定长与不定长突发传送；兼容性好；支持多个总线主设备
PCI 总线	速度快：支持线性突发传送；支持即插即用；兼容性好；可靠性高；可扩展性好
Avalon 总线	支持字节、半字和字传输；同步接口；独立的地址线、数据线和控制线；设备内嵌译码部件；支持多个总线主设备；自动生成仲裁机制；多个主设备可同时操作使用一条总线；可自动调整总线宽度，以适应尺寸不匹配的数据

3．存储器

嵌入式系统的存储器主要包括主存和外存，图 12-2 所示为嵌入式系统的存储结构，嵌入式系统的存储器主要分为三种：高速缓存（Cache）、片内主存和片外主存以及外存。

（1）高速缓存。高速缓存是存放当前使用最多的程序代码和数据的，即主存中部分内容的副本，在嵌入式系统系统中，Cache 全部集成在嵌入式微处理器内部，可以分为：数据 Cache、指令 Cache 和混合 Cache。

（2）主存。主存是处理器能够直接访问的存储器，用来存放系统和用户的程序和数据，系统上电后，主存中的代码直接运行，主存的主要特点是速度快，一般采用 ROM、EPROM、NOR flash、SRAM 和 DRAM 等存储器件。

（3）外存。外部存储器是不与运算器直接联系的后备存储器，用来存放不常用的或暂不使用的信息，外存一般以非易失性存储器构成，数据能够持久保存，即使掉电，也不消失。Flash 存储器是在 EPROM 和 EEPROM 的基础上发展起来的非易失性存储器，具有结构简单、可靠性高、体积小、质量轻、功耗低、成本低等优点，是最常用的一种外存类型。

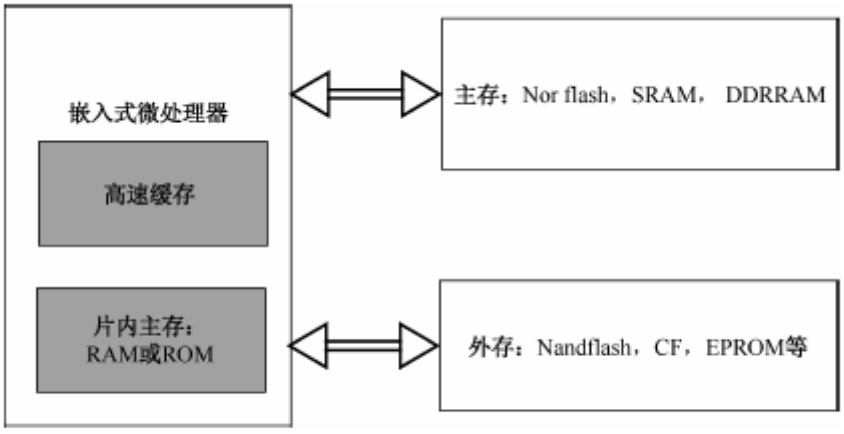


图 12-2　嵌入式系统的存储结构

4．I/O 设备与接口

因其应用领域的不同，嵌入式系统的输入设备多种多样，比较常见的有键盘、鼠标、触摸屏、手柄、声控开关等。通常，根据输入设备实现机理的不同，嵌入式系统的设备可以分

为：机械式、触控式以及声光式三类。嵌入式系统的输出设备除了通用计算机常用的显示器、打印机、绘图仪等外，还包括 LED 指示灯、LCD 屏幕、扬声器等媒体。嵌入式系统与外部设备或其他的计算机系统进行通信时，需经接口适配电路，进行工作速度、数据格式、电平等匹配与转换，嵌入式系统应用的接口形式是多种多样的。

嵌入式系统中接口电路的设计需要首先考虑的是电平匹配问题，嵌入式系统微处理器所提供与接收信号的电平，必须与所连接的设备的电平相匹配，否则将导致电路损坏或逻辑判定错误。其次，还要考虑驱动能力和干扰问题等因素。

当前，在嵌入式系统中广泛应用的接口主要有：RS232-串行接口、并行接口、USB 接口、IEEE-1394 接口以及 RJ-45 接口等，此外，以蓝牙为代表的无线接口在嵌入式系统中的应用也日趋广泛。

（1）RS-232 接口。RS-232 接口是美国电子工业协会推广的一种串行通信总线标准，是数据通信设备和数据终端设备间传输数据的接口总线，RS-232-C 标准规定其最高速率为 20kbps，在低码元畸变的情况下，最大传输距离是 15 米，通过使用增强器，其传输距离已经延长到 1000 米左右。

（2）USB 接口。USB（Universal Serial Bus，通用串行总线）是 1995 年由康柏等几大厂商共同制定的一种支持即插即用的外设接口标准，它支持 USB 外部设备到主机外部总线的连接。在 USB 系统中，必须有一个 USB 主控制器，USB 设备通过四根电缆与 USB 主控制器直接或间接相连，USB 的规范由最初的 1.0 版本发展到了 1.1 版本，以至当前主流的高速 2.0 版本，最高速率可到 480Mbps。

（3）1394 接口。IEEE1394 即火线（FireWire）最初是由 Apple 公司研制的，1995 年 IEEE 协会以 FireWire 为蓝本制定了这个串行接口标准，其电缆接口为 6 根电缆组成，包括一堆电源线和两对双绞信号线。IEEE1394 协议定义了三种传输速率：98.304Mbps、196.608Mbps 和 392.216Mbps，分别称之为 S100、S200 和 S400。为了保证数据传输率，线缆的长度一般不超过 4.5m。

IEEE1394 标准通过所有连接设备建立起一种对等网络，不需要主控节点来控制数据流，即跟 USB 技术相比，最大的区别是 IEEE1394 不需要主控制器，不同的外设之间可以直接传递信息，此外，采用该技术，两台计算机可以共享同一个外部设备。

IEEE1394 同时支持同步和异步传输两种模式。在异步传输模式下，信息的传递可以被中断，在同步模式下，数据将不受任何中断和干扰下实现连续传输。采用异步传输模式时，IEEE1394 会根据不同的设备实际需要分配相应的带宽。同时，IEEE1394 设备也支持热插拔

和即插即用。

12.2.2 软件架构

随着嵌入式技术的发展，特别是在后 PC 时代，嵌入式软件系统得到了极大的丰富和发展，形成了一个完整的软件体系，如图 12-3 所示。

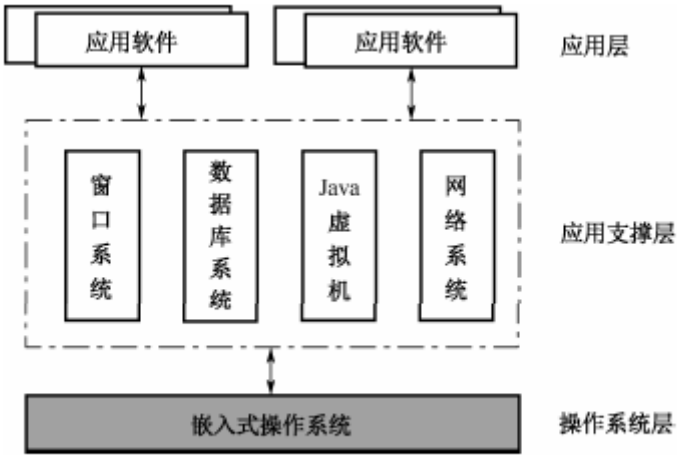


图 12-3　嵌入式系统的软件架构

这个体系自底向上由三部分组成：嵌入式操作系统、应用支撑软件和应用软件。

（1）操作系统。嵌入式操作系统由操作系统内核、应用程序接口、设备驱动程序接口等几部分组成。嵌入式操作一般采用微内核结构。操作系统只负责进程的调度、进程间的通信、内存分配及异常与中断管理最基本的任务，其他大部分的功能则由支撑软件完成。

（2）应用支撑软件。嵌入式系统中的应用支撑软件由窗口系统、网络系统、数据库管理系统及 Java 虚拟机等几部分组成。对于嵌入式系统来讲，软件的开发环境大部分在通用台式计算机和工作站上运行，但从逻辑上讲，它仍然被认为是嵌入式系统支撑软件的一部分。应用支撑软件一般用于一些浅度嵌入的系统中，如智能手机、个人数字助理等。

（3）应用软件。嵌入式系统中的应用软件是系统整体功能的集中体现。系统的能力总是通过应用软件表现出来的，一个嵌入式系统可以没有支撑软件，甚至可以没有操作系统，但不可以没有应用软件，否则它就不可能成为一个系统。从范围上讲，嵌入式系统的应用软件涉及工业控制、家电、商业、通信等诸多领域。从跟用户的交互方式上讲，有跟桌面系统类似的软件，也有嵌入程度很深、使用户感觉不到其存在的应用软件。从运行环境上讲，有在操作系统和支撑软件上运行的软件，也有直接在硬件上运行的应用软件。

12.3 嵌入式开发平台与调试环境

嵌入式系统的应用支撑软件近年来发展迅速。通常，应用支撑软件包括窗口系统、数据库管理系统及 Java 虚拟机等几个部分。应用支撑软件的出现大大改变了应用软件的开发条件，同时也使得应用系统的功能不断增强。本节主要介绍嵌入式开发环境与软件调试技术。

12.3.1 嵌入式系统软件开发平台

嵌入式系统的软件开发方法采用的不是通用的开发方法，而是交叉式开发方法。本小节主要介绍嵌入式系统软件开发的交叉编译环境的基本概念和特点，以及软件调试常用的几种方法。

1．交叉平台开发环境

嵌入式系统的软件开发采用交叉平台开发方法（Cross Platform Development，CPD），即软件在一个通用的平台上开发，而在另一个嵌入式目标平台上运行。这个用于开发嵌入式软件的通用平台通常叫作宿主机系统，被开发的嵌入式系统称为目标机系统。而当软件执行环境和开发环境一致时的开发过程则称为本地开发（Native Development，ND）。

图 12-4 是一个典型的交叉平台开发环境，通常包含三个高度集成的部分：

（1）运行在宿主机和目标机上的强有力的交叉开发工具和实用程序。

（2）运行在目标机上的高性能、可裁剪的实时操作系统。

（3）连接宿主机和目标机的多种通信方式，例如，以太网，串口线，ICE（In Circuit Emulator，在线仿真器）或 ROM 仿真器等。

图 12-4 典型交叉平台开发环境

2.交叉编译环境

宿主机提供的基本开发工具是交叉编译器、交叉链接器和源代码调试器。作为目标机的嵌入式系统则可能提供一个动态装载器、链接装载器、监视器和一个调试代理等。在目标机和宿主机之间有一组连接，通过这组连接程序代码映像从宿主机下载到目标机，这组连接同

时也用来传输宿主机和目标机调试代理之间的信息。

嵌入式系统开发人员需要完全了解目标机系统如何在嵌入式系统上存储程序映像、可执行映像如何下载到内存及执行控制如何传给应用，以及运行期间如何和何时装载程序映像，如何交叉式地开发和调试应用系统。这些方面对代码如何开发、编译、链接等步骤都有影响。

图 12-5 是一个典型的交叉编译环境的示例，显示了如何使用开发工具处理各种输入文件，并产生最终目标文件的过程。其中，交叉编译器将用户编写的 C/C++/Java 源代码文件根据目标机的 CPU 类型生成包含二进制代码和程序数据的目标文件。在此过程中，交叉编译器会建立一个符号表，包含所产生的目标文件中指向映像地址的符号名，当建立重定位输出时，编译器为每个相关的符号产生地址。

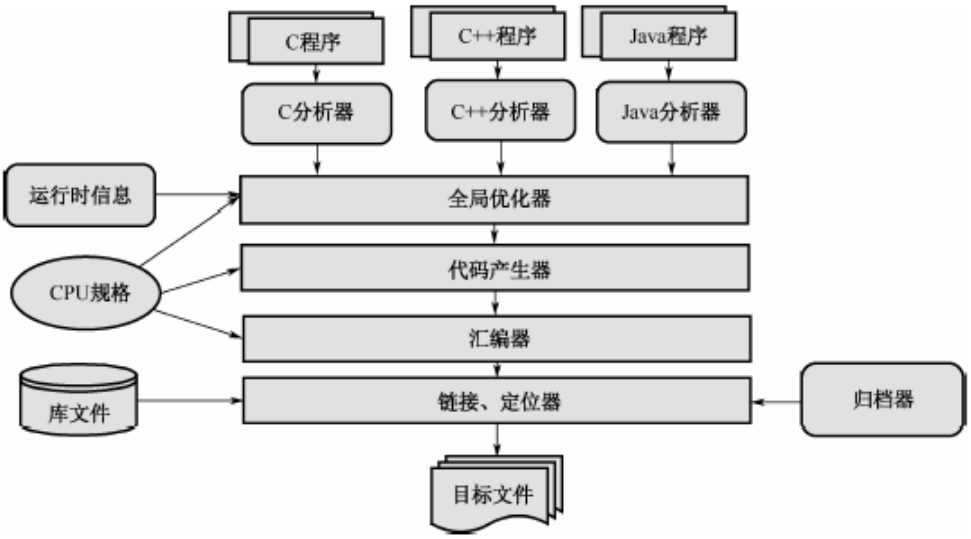


图 12-5　嵌入式系统的交叉编译环境

通常，还可以使用归档工具将这些目标文件收集到一起形成一个库。最后，链接器将这些目标文件作为输入来产生一个可执行的映像。

一般，用户会先编辑一个链接脚本文件，用来指示链接器如何组合和重定位各个代码段以便生成最终文件。此外，链接器还可以将多个目标文件组合成一个更大的重定位目标文件或一个共享目标文件。

目前，嵌入式系统中常用的目标文件格式是 COFF（Common Object File Format，公共对象文件格式）和 ELF（Executable Linking Format，可执行链接格式）。另外，一些系统还需要有一些专门的工具将上述格式转换成二进制代码格式才可使用。

通常，一个目标文件包含：

（1）关于目标文件的通用信息，如文件尺寸、启动地址、代码段和数据段等。

（2）机器架构特定的二进制指令和数据。

（3）符号表和重定位表。

（4）调试信息。

12.3.2 嵌入式开发调试

通用系统与嵌入式系统的软件调试过程存在着明显的差异。对于通用系统，调试工具与被调试的程序位于同一台计算机上，调试工具通过操作系统的调试接口来控制被调试的程序。但是在嵌入式系统中，由于资源的限制，不能直接在其上开发应用程序，调试过程通常也以交叉方式进行的。在实际开发实践中，经常采用的调试方法有直接测试法、调试监控法、在线仿真法、片上调试法及模拟器法等。

1．直接调试法

直接调试法就是将目标代码下载到目标机上，让其执行，通过观察指示灯来判断程序的运行状态。在嵌入式系统发展的早期一般采用这种方式进行调试，其基本步骤是：

（1）在宿主机上编写程序。

（2）在宿主机上编译、链接生成目标机可执行程序代码。

（3）将可执行代码写入目标机的存储器中。

（4）在目标机运行程序代码。

（5）判断程序的运行情况，如有错误则纠正错误，重复以上步骤，直到正确为止。

（6）将可执行代码固化到目标机，开发完成。

这种方法是最原始的调试方法，程序运行时产生的问题，只有通过检查源代码来解决，因而开发效率很低。

2．调试监控法

调试监控法也叫插桩法。目标机和宿主机一般通过串行口、并行口或以太网相连接，采用这种方法还需要在宿主机的调试器内和目标机的操作系统上分别启动一个功能模块，然后通过这两个功能模块的相互通信来实现对应用程序的调试。在目标机上添加的模块叫作桩，也叫调试服务器或调试监控器，主要有两个作用：其一，监视和控制被调试的程序；其二，跟宿主机上调试程序通信，接收控制指令，返回结果等。

在进行调试的时候，宿主机上的调试器通过连接线路向调试监控器发送各种请求，实现目标机内存读/写和寄存器访问、程序下载、单步跟踪和设置断点等操作。来自宿主机的请求和目标机的响应都按照预定的通信协议进行交互。

使用插桩法作为调试手段时，开发应用程序的基本步骤是：

（1）在宿主机上编写程序的源代码。

（2）在宿主机编译、链接生成目标机可执行程序。

（3）将目标机可执行代码下载到目标机的存储器中。

（4）使用调试器进行调试。

（5）在调试器帮助下定位错误。

（6）在宿主机上修改源代码，纠正错误，重复上述步骤直到正确为止。

（7）将可执行代码固化到目标机上。

相对于直接测试法，插桩法明显地提高了开发效率，降低了调试的难度，缩短了产品的开发周期，有效降低了开发成本。但是插桩法仍有明显的缺点，主要体现在以下几个方面：

（1）调试监控器本身的开发是个技术难题。

（2）调试监控器在目标机中要占用一定的系统资源，如 CPU 时间、存储空间及串口或网络接口等外设资源。

（3）调试时，不能响应外部中断，对有时间特性的程序不适用。

（4）在调试过程中，被调试的程序实际上在调试监控器所提供的环境中运行，这个环境可能会与实际目标程序最终的运行环境有一定的差异，这种差异有可能导致调试通过的程序最后仍不能运行。

为了克服插桩法的缺点，出现了一种改良的方法，即 ROM 仿真器法。

ROM 仿真器可以被认为是一种用于替代目标机上 ROM 芯片的硬件设备，ROM 仿真器一端跟宿主机相连，一端通过 ROM 芯片的引脚插座与目标机相连。对于嵌入式处理器来说，ROM 仿真器像是一个只读存储器，而对于宿主机来说，像一个调试监控器。ROM 仿真器的地址可以实时映射到目标机的 ROM 地址空间里，所以它可以仿真目标机的 ROM。ROM 仿真器在目标机和宿主机之间建立了一条高速信息通道，其典型的应用就是跟插桩法相结合，形成一种功能更强的调试方法。该方法具有如下优点：

（1）不必再开发调试监控器。

（2）由于是通过 ROM 仿真器上的串行口、并行口或网络接口与宿主机连接，所以不必占用目标机上的系统资源。

（3）ROM 仿真器代替了目标机上原来的 ROM，所以不必占用目标机上的存储空间来保存调试监控器。

（4）另外，即使目标机本身没有 ROM，调试依然可以进行，并且不需要使用专门工

具向 ROM 写入程序和数据。

3．在线仿真法

ICE 是一种用于替代目标机上 CPU 的设备。对目标机来说，在线仿真器就相当于它的 CPU，在线仿真器本身就是一个嵌入式系统，有自己的 CPU、内存和软件。在线仿真器的 CPU 可以执行目标机的所有指令，但比一般的 CPU 有更多的引脚，能够将内部信号输出到被控制的目标机上，在线仿真器的存储器也被映射到用户的程序空间，因此，即使没有目标机，仅用在线仿真器也可以进行程序的调试。

在线仿真器和宿主机一般通过串行口、并行口或以太网相连接。在连接在线仿真器和目标系统时，用在线仿真器的 CPU 引出端口替代目标机的 CPU。在用在线仿真器调试程序时，在宿主机运行一个调试器界面程序，该程序根据用户的操作指令控制目标机上的程序运行。

在线仿真器能实时地检查运行程序的处理器的状态，设置硬件断点和进行实时跟踪，所以提供了更强的调试功能。在线仿真器，支持多种事件的触发断点，这些事件包括内存读写、I/O 读写及中断等。在线仿真器的一个重要特性就是实时跟踪，在线仿真器上有大容量的存储器用来保存每个指令周期的信息，这个功能使用户可以知道事件发生的精确时序，特别适于调试实时应用、设备驱动程序和对硬件进行功能测试。但是，在线仿真器的价格一般都比较昂贵。

4．片上调试法

片上调试（In Circuit Debugger，ICD）是 CPU 芯片内部的一种用于支持调试的功能模块。按照实现的技术，片上调试可以分为仿调试监控器、后台调试模式（Background Debugging Mode，BDM）、连接测试存取组（Joint Test Access Group，JTAG）和片上仿真（On Chip Emulation，OnCE）等几类。仿调试监控器类的 ICD 芯片产品有 Motorola 的 CPU16、 CPU32 和 ColdFire 等系列，BDM 类的主要有 Motorola 的 MPC5xx 和 MPC8xx 系列等， OnCE 类的主要有 Motorola 的 DSP 芯片系列，JTAG 类的主要有 PPC6xxx、PPC4xx、 ARM7TDMI、ARM9TDMI 及 Intel1960 等。

目前，使用较多的是采用 BDM 技术的 CPU 芯片。这种芯片的外面有跟调试相关的引脚，这些引脚在调试的时候被引出，形成一个与外部相连的调试接口，这种 CPU 具有调试模式和执行模式两种不同的运行模式。当满足了特定的触发条件时，CPU 进入调试模式，在调试模式下，CPU 不再从内存中读取指令，而是通过其调试端口读取指令，通过调试端口还可以控制 CPU 进入和退出调试模式。这样在宿主机上的调试器就可以通过调试端口直接向目标机发送要执行的指令，使调试器可以读/写目标机的内存和寄存器，控制目标程序

的运行及完成各种复杂的调试功能。

该方法的主要优点是：不占用目标机的通信端口等资源；调试环境和最终的程序运行环境基本一致；无须在目标机上增加任何功能模块即可进行；支持软、硬断点；支持跟踪功能，可以精确计量程序的执行时间；支持时序逻辑分析等功能。

该方法的缺点是：实时性不如在线仿真器法强；使用范围受限，如果目标机不支持片上调试功能，则该方法不适用；实现技术多样，标准不完全统一，工具软件的开发和使用均不方便。

5．模拟器法

模拟器是运行于宿主机上的一个纯软件工具，它通过模拟目标机的指令系统或目标机操作系统的系统调用来达到在宿主机上运行和调试嵌入式应用程序的目的。

模拟器适合于调试非实时的应用程序，这类程序一般不与外部设备交互，实时性不强，程序的执行过程是时间封闭的，开发者可以直接在宿主机上验证程序的逻辑正确性。当确认无误后，将程序写入目标机上就可正确运行。

模拟器有两种主要类型：一类是指令级模拟器，在宿主机模拟目标机的指令系统；另一类是系统调用级模拟器，在宿主机上模拟目标操作系统的系统调用。指令级的模拟器相当于宿主机上的一台虚拟目标机，该目标机的处理器种类可以与宿主机不同，如宿主机是英特尔的 x86 系列机，而虚拟机可以是 ARM、PowerPC、MIPS 等。比较高级的指令级模拟器还可以模拟目标机的外部设备，如键盘、串口、网络接口等。系统调用级的模拟器相当于在宿主机上安装了目标机的操作系统，使得基于目标机的操作系统的应用程序可以在宿主机上运行。被模拟的目标机操作系统的类型可以跟宿主机的不同。两种类型的模拟器相比较，指令级模拟器所提供的运行环境与实际目标机更为接近。

使用模拟器的最大好处是在实际的目标机不存在的条件下就可以为其开发应用程序，并且在调试时利用宿主机的资源提供更详细的错误诊断信息，但模拟器有许多不足之处：

（1）模拟器环境和实际运行环境差别很大，无法保证在模拟条件下通过的应用程序也能在真实环境中正确运行。

（2）模拟器不能模拟所有的外部设备，嵌入式系统通常包含诸多外设，但模拟器只能模拟少数部分。

（3）模拟器的实时性差，对于实时类应用程序的调试结果可能不可靠。

（4）运行模拟器需要较高的宿主机配置。

尽管模拟器有很多的不足之处，但在项目开发的早期阶段，其价值是不可估量的，尤其

对那些实时性不强的应用，模拟器调试不需要特殊的硬件资源，是一种非常经济的方法。

12.4 嵌入式网络系统

嵌入式网络是用于连接各种嵌入式系统，使之可以互相传递信息、共享资源的网络系统。嵌入式系统在不同的场合采用不同的连接技术，如在家庭居室采用家庭信息网，在工业自动化领域采用现场总线，在移动信息设备等嵌入式系统则采用移动通信网，此外，还有一些专用连接技术用于连接嵌入式系统。

12.4.1 现场总线网

现场总线（FieldBus）是 20 世纪 80 年代中期继模拟仪表控制系统、集中式数字控制系统及集散控制系统之后，发展起来的一项计算机控制技术，它是当今自动化控制领域技术发展的热点之一，通常也被称作工业自动化领域的计算机局域网。

现场总线是一种将数字传感器、变换器、工业仪表及控制执行机构等现场设备与工业过程控制单元、现场操作站等互相连接而成的网络。它具有全数字化、分散、双向传输和多分支的特点，是工业控制网络向现场级发展的产物。

现场总线是一种低带宽的底层控制网络，位于生产控制和网络结构的底层，因此也被称为底层网（Infranet）。它主要应用于生产现场，在测量控制设备之间实现双向的、串行的、多节点的数字通信。

现场总线控制系统（Field Control System，FCS）是运用现场总线连接各控制器及仪表设备而构成的控制系统，该控制系统将控制功能彻底下放到现场，降低了安装成本和维护费用。实际上 FCS 是一种开放的、具有互操作性的、彻底分散的分布式控制系统。

嵌入式现场控制系统将专用微处理器置入传统的测量控制仪表，使其具备数字计算和数字通信能力。它采用双绞线、电力线或光纤等作为总线，把多个测量控制仪表连接成网络，并按照规范标准的通信协议，在位于现场的多个微机化测量控制设备之间及现场仪表与远程监控计算机之间，实现数据传输与信息交换，形成了各种适用于实际需要的自动控制系统。简言之，现场总线控制系统把单个分散的测量控制设备变成网络节点，以现场总线为纽带，使这些分散的设备成为可以互相沟通信息共同完成自动控制任务的网络系统。借助于现场总线技术，传统上的单个分散控制设备变成了互相沟通、协同工作的整体。

12.4.2 家庭信息网

家庭信息网是一种把家庭范围内的个人计算机、家用电器、水、电、气仪表、照明设备和网络设备、安全设备连接在一起的局域网。其主要功能是集中控制上述设备并将其接入Internet，以共享网络资源和服务。此外，家庭信息网还可以扩展至整幢住宅甚至整个社区，成为智能住宅小区和智能社会的基础。

在家庭信息网络系统中，所有的家庭设备都是智能化的，包括家用电器、水、电、气仪表，以及照明设备等。它们能够互相通信，并通过家庭网关接入 Internet。家庭信息网络的实现为人们提供了更加安全、便捷、舒适的家庭环境。如主人外出时，大门自动关闭、上锁，监视系统自动开启，家中出现异常情况能够自动通知主人，对家中各种设备能够随时随地进行控制，仪表数据能够自动上传等。

家庭信息网需要解决两个基本问题：

（1）如何将家用电器，水、电、气仪表，照明设备等互相连接起来。

（2）如何实现这些连在一起的设备间的互操作，即家庭信息网上的设备可以在需要的时候自动请求服务，相关设备可以提供服务或接受请求并对其进行处理。

家庭信息网可以采用不同的拓扑结构，如总线型、星型结构等。家庭信息网内部还可以进一步划分出若干控制子网和数据子网，其中控制子网类似于现场总线，是一种带宽不高、主要用于发送和接收控制信息的网络。而数据子网对带宽的要求则较高，连接在其上的设备需要传送大量的数据信息。

11.4.3 无线数据通信网

近年来，随着移动电话通信的迅速发展，个人计算机的迅速普及，多种便携式计算机，例如膝上型计算机、笔记本计算机、手持式计算机等迅速增多，固定计算机之间的数据通信已不能满足需要。人们希望能随时随地进行数据信息的传送和交换，于是数据通信传输媒体开始从有线扩展到无线，出现了无线移动数据通信。

无线数据通信网是一种通过无线电波传送数据的网络系统。它是在有线数据通信的基础上发展起来的，能实现移动状态下的数据通信。通过无线数据通信网，智能手机、PDA 及笔记本计算机可以互相传递数据信息，并接入 Internet。

无线数据通信网分为短程无线网和无线 Internet。短程无线网主要包括 802.11、蓝牙、IrDA 及 HomeRF 等。无线 Internet 或移动 Internet 主要采用两种无线连接技术：一种是

移动无线接入技术，例如，GSM、GPRS、CDPD（Cellular Digital Packet Data）等；另一种是固定无线接入技术，包括微波、扩频通信、卫星及无线光传输等。

12.4.4 嵌入式 Internet

随着 Internet 和嵌入式技术的飞速发展，越来越多的信息电器，如 Web 可视电话、机顶盒以及信息家电等嵌入式系统产品都要求与 Internet 连接，来共享 Internet 所提供的方便、快捷、无处不在的信息资源和服务，即嵌入式 Internet 技术。嵌入式 Internet 技术在智能交通、家政系统、家庭自动化、工业自动化、POS 及电子商务等领域具有广阔的应用前景。

1．嵌入式 Internet 的接入方式

嵌入式设备上集成了 TCP/IP 协议栈及相关软件，这类设备可以作为 Internet 的一个节点，分配有 IP 地址，与 Internet 直接互联。这种接入方式的特点是：

设备可以直接连接到 Internet，对 Internet 进行透明访问；

不需要专门的接入设备；

设备的协议标准化；

需要的处理器性能和资源相对较高；

需要占用 IP 资源，由于目前 IPv4 资源紧张，这种方案在 IPv6 网中可能更现实。

通过网关接入 Internet，即采用瘦设备方案，设备不直接接入 Internet，不需要复杂的 TCP/IP 协议全集，而是通过接入设备接入 Internet。如嵌入式微型网互联技术（Embedded Micro Internet-working Technology，EMIT）便是一种将嵌入式设备接入 Internet 的技术。这种接入方式的特点是：

对接入设备的性能和资源要求较低；

接入设备的协议栈开销较小；

不需要分配合法的 IP 地址；

可以降低系统的整体成本；

设备可以实现多样化、小型化。

2．嵌入式 TCP/IP 协议栈

嵌入式 TCP/IP 协议栈完成的功能与完整的 TCP/IP 协议栈是相同的，但是由于嵌入式系统的资源限制，嵌入式协议栈的一些指标和接口等与普通的协议栈可能有所不同。

（1）嵌入式协议栈的调用接口与普通的协议栈不同。普通协议栈的套接字接口是标准的，应用软件的兼容性好，但是，实现标准化接口的代码开销、处理和存储开销都是巨大的。因此，多数厂商在将标准的协议栈接口移植到嵌入式系统上的时候，都做了不同程度的修改简化，建立了高效率的专用协议栈，它们所提供的 API 与通用协议栈的 API 不一定完全一致。

（2）嵌入式协议栈的可裁剪性。嵌入式协议栈多数是模块化的，如果存储器的空间有限，可以在需要时进行动态安装，并且都省去了接口转发、全套的 Internet 服务工具等几个针对嵌入式系统非必需的部分。

（3）嵌入式协议栈的平台兼容性。一般协议栈与操作系统的结合紧密，大多数协议栈是在操作系统内核中实现的。协议栈的实现依赖于操作系统提供的服务，移植性较差。嵌入式协议栈的实现一般对操作系统的依赖性不大，便于移植。许多商业化的嵌入式协议栈支持多种操作系统平台。

（4）嵌入式协议栈的高效率。嵌入式协议栈的实现通常占用更少的空间，需要的数据存储器更小，代码效率高，从而降低了对处理器性能的要求。

12.5 嵌入式数据库管理系统

随着嵌入式技术的发展，嵌入式数据库逐步走向应用。本质上，嵌入式数据库是由通用数据库发展而来的，在各种嵌入式设备上或移动设备上运行，在嵌入式系统中更显示出其优越性，由于受到嵌入式系统本身应用环境的制约，嵌入式数据库有着与通用数据库不同的特点。

通常，嵌入式数据库管理系统就是在嵌入式设备上使用的数据库管理系统。由于用到嵌入式数据库管理系统的多是移动信息设备，诸如掌上电脑、PDA、车载设备等移动通信设备，位置固定的嵌入式设备很少用到，所以，嵌入式数据库也称为移动数据库或嵌入式移动数据库。其作用主要是解决移动计算环境下数据的管理问题，移动数据库是移动计算环境中的分布式数据库。

在嵌入式系统中引入数据库技术，主要是因为直接在嵌入式操作系统或裸机之上开发信息管理应用程序存在如下缺点：

（1）所有的应用都要重复进行数据的管理工作，增加了开发难度和代价。

（2）各应用之间的数据共享性差。

（3）应用软件的独立性、可移植性差，可重用度低。

在嵌入式系统中引入数据库管理系统可以在很大程度上解决上述问题，提高应用系统的开发效率和可移植性。

12.5.1 使用环境的特点

嵌入式数据库系统是一个包含嵌入式数据库管理系统在内的跨越移动通信设备、工作站或台式机及数据服务器的综合系统，系统所具有的这个特点及该系统的使用环境对嵌入式数据库管理系统有着较大的影响，直接影响到嵌入式数据库管理系统的结构。其使用环境的特点可以简单地归纳如下：

（1）设备随时移动性，嵌入式数据库主要用于移动信息设备上，设备的位置经常随使用者一起移动。

（2）网络频繁断接，移动设备或移动终端在使用的过程中，位置经常发生变化，同时也受到使用方式、电源、无线通信及网络条件等因素的影响。所以，一般并不持续保持网络连接，而是经常主动或被动地间歇性断接和连接。

（3）网络条件多样化，由于移动信息设备位置的经常变化，所以移动信息设备同数据服务器在不同的时间可能通过不同的网络系统连接。这些网络在网络带宽、通信代价、网络延迟、服务质量等方面可能有所差异。

（4）通信能力不对称，由于受到移动设备的资源限制，移动设备与服务器之间的网络通信能力是非对称的。移动设备的发送能力都非常有限，使得数据服务器到移动设备的下行通信带宽和移动设备到数据服务器之间的上行带宽相差很大。

12.5.2 系统组成与关键技术

一个完整的嵌入式数据库管理系统由若干子系统组成，包括主数据库管理系统、同步服务器、嵌入式数据库管理系统、连接网络等几个子系统，如图 12-6 所示。

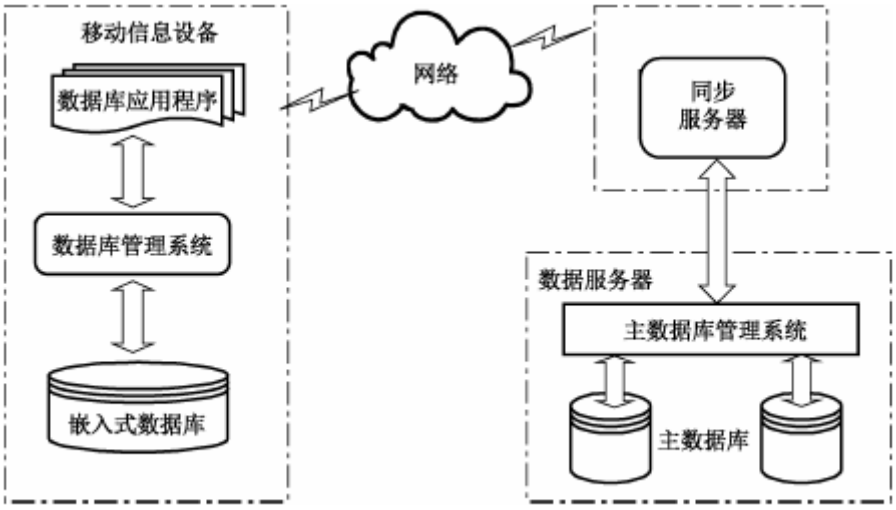


图 12-6　嵌入式数据库系统组成

（1）嵌入式数据库管理系统。嵌入式数据库管理系统是一个功能独立的单用户数据库管理系统。它可以独立于同步服务器和主数据库管理系统运行，对嵌入式系统中的数据进行管理，也可以通过同步服务器连接到主服务器上，对主数据库中的数据进行操作，还可以通过多种方式进行数据同步。

（2）同步服务器。同步服务器是嵌入式数据库和主数据库之间的连接枢纽，保证嵌入式数据库和主数据库中数据的一致性。

（3）数据服务器。数据服务器的主数据库及数据库管理系统可以采用 Oracle 或 Sybase 等大型通用数据库系统。

（4）连接网络。主数据库服务器和同步服务器之间一般通过高带宽、低延迟的固定网络进行连接。移动设备和同步服务器之间的连接根据设备的具体情况可以是无线局域网、红外连接、通用串行线或公众网等。

1．嵌入式移动数据库在应用中的关键

嵌入式移动数据库在实际应用中必须解决好数据的一致性（复制性）、高效的事务处理和数据的安全性等问题。

（1）数据的一致性。嵌入式移动数据库的一个显著特点是，移动数据终端之间及与同步服务器之间的连接是一种弱连接，即低带宽、长延迟、不稳定和经常性断接。为了支持用户在弱环境下对数据库的操作，现在普遍采用乐观复制方法（Optimistic Replication 或 Lazy Replication）允许用户对本地缓存上的数据副本进行操作。待网络重新连接后再与数据库服务器或其他移动数据终端交换数据修改信息，并通过冲突检测和协调来恢复数据的一致性。

（2）高效的事务处理。移动事务处理要在移动环境中频繁的、可预见的断接情况下进

行。为了保证活动事务的顺利完成，必须设计和实现新的事务管理策略和算法。

根据网络连接情况来确定事务处理的优先级，网络连接速度高的事务请求优先处理；

根据操作时间来确定事务是否迁移，即长时间的事务操作将全部迁移到服务器上执行，无须保证网络的一直畅通；

根据数据量的大小来确定事务是上载执行还是下载数据副本执行后上载；

完善的日志记录策略；

事务处理过程中，网络断接处理时采用服务器发现机制还是采用客户端声明机制；

事务移动（如：位置相关查询）过程中的用户位置属性的实时更新。

（3）数据的安全性。许多应用领域的嵌入式设备是系统中数据管理或处理的关键设备，因此嵌入式设备上的数据库系统对存取权限的控制较严格。同时，许多嵌入式设备具有较高的移动性、便携性和非固定的工作环境，也带来潜在的不安全因素。此外，某些数据的个人隐私性又很高，因此在防止碰撞、磁场干扰、遗失、盗窃等方面对个人数据的安全性需要提供充分的保证。保证数据安全的主要措施是：

对移动终端进行认证，防止非法终端的欺骗性接入；

对无线通信进行加密，防止数据信息泄漏；

对下载的数据副本加密存储，以防移动终端物理丢失后的数据泄密。

2．移动数据库管理系统的特性

移动 DBMS 的计算环境是传统分布式 DBMS 的扩展，它可以看做客户端与固定服务器结点动态连接的分布式系统。因此移动计算环境中的数据库管理系统是一种动态分布式数据库管理系统。由于嵌入式移动数据库管理系统在移动计算的环境下应用在嵌入式操作系统之上，所以它有自己的特点和功能需求：

（1）微核结构，便于实现嵌入式功能。考虑到嵌入式设备的资源有限，嵌入式移动 DBMS 应采用微型化技术实现，在满足应用的前提下紧缩其系统结构以满足嵌入式应用的需求。

（2）对标准 SQL 的支持。嵌入式移动 DBMS 应能提供对标准 SQL 的支持。支持 SQL92 标准的子集，支持数据查询（连接查询、子查询、排序、分组等）、插入、更新、删除多种标准的 SQL 语句，充分满足嵌入式应用开发的需求。

（3）事务管理功能。嵌入式移动 DBMS 应具有事务处理功能，自动维护事务的完整性、原子性等特性；支持实体完整性和引用完整性。

（4）完善的数据同步机制。数据同步是嵌入式数据库最重要的特点。通过数据复制，可以将嵌入式数据库或主数据库的变化情况应用到对方，保证数据的一致性。

嵌入式移动数据库管理系统的数据同步机制应具有以下几个特点：

提供多种数据同步方式，具有上载同步、下载同步和完全同步 3 种同步方式；

具有完善的冲突检测机制和灵活的冲突解决方案，具有冲突日志记录功能；

支持快速同步，系统同步时，只传递变化的数据，节省了大量的同步时间；

支持表的水平分割和垂直分割复制，最大限度地降低了嵌入式数据库的大小；

支持异构数据源连接同步，可以用支持 ODBC 的异构数据源作为主数据库和嵌入式设备上的数据库进行数据同步；

具有主动同步的功能，允许用户对系统提供的同步事件自定义过程实现，提供了最大灵活度的同步过程。

（5）支持多种连接协议。嵌入式移动 DBMS 应支持多种通信连接协议。可以通过串行通信、TCP/IP、红外传输、蓝牙等多种连接方式实现与嵌入式设备和数据库服务器的连接。

（6）完备的嵌入式数据库管理功能。嵌入式移动 DBMS 应具有自动恢复功能，基本无须人工干预进行嵌入式数据库管理并能够提供数据的备份和恢复，保证用户数据的安全可靠。

（7）平台无关性与支持多种嵌入式操作系统。嵌入式移动 DBMS 应能支持 Windows CE、Palm OS 等多种目前流行的嵌入式操作系统，这样才能使嵌入式移动数据库管理系统不受移动终端的限制。

（8）零管理特性。嵌入式数据库具有自动恢复功能，不需要人工干预就可以进行嵌入式数据库管理，并提供数据的备份与同步。

另外，一种理想的状态是用户只用一台移动终端（如手机）就能对与它相关的所有移动数据库进行数据操作和管理。这就要求前端系统具有通用性，而且要求移动数据库的接口有统一、规范的标准。前端管理系统在进行数据处理时自动生成统一的事务处理命令，提交当前所连接的数据服务器执行。这样就有效地增强了嵌入式移动数据库管理系统的通用性，扩大了嵌入式移动数据库的应用前景。

总之，在嵌入式移动数据库管理系统中还需要考虑诸多传统计算环境下不需要考虑的问题，如对断接操作的支持、对跨区长事务的支持、对位置相关查询的支持、对查询优化的特殊考虑及对提高有限资源的利用率和对系统效率的考虑等。为了有效地解决上述问题，诸如复制与缓存技术、移动事务处理、数据广播技术、移动查询处理与查询优化、位置相关的数据处理及查询技术、移动信息发布技术、移动 Agent 等技术仍在不断地发展和完善，会进一步促进嵌入式移动数据库管理系统的发展。

12.6 实时系统与嵌入式操作系统

简单地说，实时系统可以看成对外部事件能够及时响应的系统。这种系统最重要的特征是时间性，也就是实时性，实时系统的正确性不仅依赖于系统计算的逻辑结果，还依赖于产生这些结果的时间。

目前，大多数实时系统都是嵌入式的，并且实际运行中的嵌入式系统也都有实时性的需求，因此，在诸多类型的嵌入式操作系统中，实时嵌入式操作系统是最具代表性的一类，它融合了几乎所有类型的嵌入式操作系统的特点，所以本节主要以实时嵌入式操作系统的特性和概念为主线，对嵌入式操作系统的基本概念与特点、基本架构、内核服务、内核对象与内核服务等核心内容进行全面的介绍。

12.6.1 嵌入式系统的实时概念

现实世界中，并非所有的嵌入式系统都具有实时特性，所有的实时系统也不一定都是嵌入式的。但这两种系统并不互相排斥，兼有这两种系统特性的系统称为实时嵌入式系统。

它们之间的关系如图 12-7 所示。

图 12-7　实时嵌入式系统

（1）逻辑（或功能）正确，是指系统对外部事件的处理能够产生正确的结果。

（2）时间正确，是指系统对外部事件的处理必须在预定的周期内完成。

（3）死线（Deadline）或时限、死限、截止时间，是指系统必须对外部事件进行处理的最迟时间界限，错过此界限可能产生严重的后果。通常，计算必须在到达时限前完成。

（4）实时系统，是指功能正确和时间正确同时满足的系统，二者同等重要。换言之，实时系统有时间约束并且是时限驱动的。但是在某些系统中，为了保证功能正确性，有可能牺牲时间正确性。

对于实时系统的划分，通常还可以根据实时性的强弱，即系统必须对外部事件做出响应的时间长短，将实时系统分为：

（1）强实时系统，其系统的响应时间非常短，通常在毫秒或微秒级。

（2）一般实时系统，其系统响应时间比强实时系统要求要低，通常在秒级。

（3）弱实时系统，其系统响应时间可以更长，也可以随系统负载的轻重而变化。

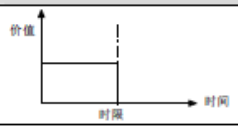
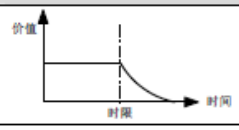
根据对错失时限的容忍程度或后果的严重性，可以将实时系统分为软实时系统和硬实时系统。

（1）硬实时系统，指系统必须满足其灵活性接近零时限要求的实时系统。时限必须满足否则就会产生灾难性后果，并且时限之后得到的处理结果或是零级无用，或是高度贬值。

（2）软实时系统，指必须满足时限的要求，但是有一定灵活性的实时系统。时限可以包含可变的容忍等级、平均的截止时限，甚至是带有不同程度的、可接受性的响应时间的统计分布。在软实时系统中，时限错失通常不会导致系统失败等严重后果。

表 12-2 是对软实时和硬实时系统的对比。

表 12-2 软实时和硬实时系统对比

	硬实时系统	软实时系统
计算结果的价值曲线		
错失时限的后果	在硬实时系统中，错失时限后的处理结果价值为零，错失时限的惩罚是灾难性的	在软实时系统中，错失时限后处理结果的价值根据应用的性质随时间按某种关系下降
实例解析	导弹导航系统：导弹导航系统是硬实时系统。导航系统在时限前如果不能对将要到达的山地区域进行新坐标的计算，就没有足够的距离让导弹改变姿态，来防止与山相撞。该系统错失时限是零容忍的，时限后的坐标不再有用了	DVD 播放器：DVD 播放器是软实时系统。播放器根据用户的命令实时解码视频音频流，用户可以快速发送一系列指令使解码器错失一个或多个时限，除了看到视频弯曲和音频变调外，播放器可以继续工作，人们对此可以容忍，损失是暂时的，时限后的解码数据仍然有用

通过比较，可知由于错过时限对软实时系统的运行没有决定性的影响，一个软实时系统不必预测是否可能有悬而未决的时限错失。相反，软实时系统在探知到错失一个时限后可以启动一个恢复进程。

在实时系统中，任务的开始时间跟时限或完成时间同样重要，由于任务缺少需要的资源，如 CPU 和内存，就有可能阻碍任务执行的开始并直接导致错失任务的完成时限，因此时限问题演变成了资源的调度问题。

这一点对调度算法和任务设计都有至关重要的影响。

12.6.2 嵌入式操作系统概述

所谓嵌入式操作系统就是指运行在嵌入式计算机系统上支持嵌入式应用程序的操作系统，是用于控制和管理嵌入式系统中的硬件和软件资源、提供系统服务的软件集合。嵌入式

操作系统是嵌入式软件的一个重要组成部分。它的出现提高了嵌入式软件开发的效率，提高了应用软件的可移植性，有力地推动了嵌入式系统的发展。

1．嵌入式操作系统的特点

与通用操作系统相比，嵌入式操作系统主要有以下特点：

（1）微型化：嵌入式操作系统的运行平台不是通用计算机，而是嵌入式计算机系统。这类系统一般没有大容量的内存，几乎没有外存，因此，嵌入式操作系统必须做得小巧，以尽量少占用系统资源。为了提高系统的执行速度和可靠性，嵌入式系统中的软件一般都固化在存储器芯片中，而不是存放在磁盘等载体中。

（2）代码质量高：在大多数应用中，存储空间依然是宝贵的资源，这就要求程序代码的质量要高，代码要尽量精简。

（3）专业化：嵌入式系统的硬件平台多种多样，处理器更新速度快，每种都是针对不同的应用领域而专门设计。因此，嵌入式操作系统要有很好的适应性和移植性，还要支持多种开发平台。

（4）实时性强：嵌入式系统广泛应用于过程控制、数据采集、通信、多媒体信息处理等要求实时响应的场合，因此实时性成为嵌入式操作系统的又一特点。

（5）可裁减、可配置：应用的多样性要求嵌入式操作系统具有较强的适应能力，能够根据应用的特点和具体要求进行灵活配置和合理裁减，以适应微型化和专业化的要求。

2．嵌入式操作系统的分类

嵌入式操作系统的种类繁多，可以从不同角度对其进行分类。

从嵌入式操作系统的获得形式上，可以分为商业型和免费型两类：

（1）商业型。商业型嵌入式操作系统一般功能稳定、可靠，有完善的技术支持、齐全的开发工具和售后服务。如 WindRiver 公司的 VxWorks 、pSOS 和 Palm 公司的 Palm OS 等。但是，价格昂贵，用户通常得不到系统的源代码。

（2）免费型。免费型嵌入式操作系统的优势在于价格方面，另外，应用系统开发者可以获得系统源代码，给开发带来了方便。但免费型的操作系统功能简单、技术支持差、系统的稳定性也不够好。典型代表系统有嵌入式 Linux、uC/OS 等。

从嵌入式操作系统的实时性上，可以分为实时嵌入式操作系统和非实时嵌入式操作系统两类。

（1）实时嵌入式操作系统（Real-Time Embedded OS，RTEOS）。实时嵌入式操作系统支持实时系统工作，其首要任务是调度一切可利用资源，以满足对外部事件响应的实时时限，

其次着眼于提高系统的使用效率。实时嵌入式操作系统主要用在控制、通信等领域。目前，大多数商业嵌入式操作系统都是实时操作系统。

（2）非实时嵌入式操作系统。这类操作系统不特别关注单个任务响应时限，其平均性能、系统效率和资源利用率一般较高，适合于实时性要求不严格的消费类电子产品，如个人数字助理、机顶盒等。

12.6.3 实时嵌入式操作系统

整体上看，一个嵌入式系统的实时性能是由硬件、实时操作系统及应用程序共同决定的，其中，嵌入式实时操作系统内核的性能起着关键的作用。通常，有两种类型的实时嵌入式操作系统：实时内核型的 RTEOS 与通用型的 RTEOS。

实时内核型的 RTEOS：这类操作系统，驱动程序传统嵌在内核之中，应用程序和中间件实现在标准的应用程序接口（APIs，Application Programming Interfaces）之上。

实时通用型的 RTEOS：这类操作系统，驱动程序并非深度嵌入到内核中，而是在内核之上实现，并且仅包含少数必要的驱动程序，应用程序和中间件可以直接在驱动程序之上实现，而不必在标准的 APIs 实现。它们的区别如图 12-8 所示。

实时嵌入式操作系统和通用操作系统之间的功能有很多相似之处，例如，它们都支持多任务，支持软件和硬件的资源管理，以及都为应用提供基本的操作系统服务。

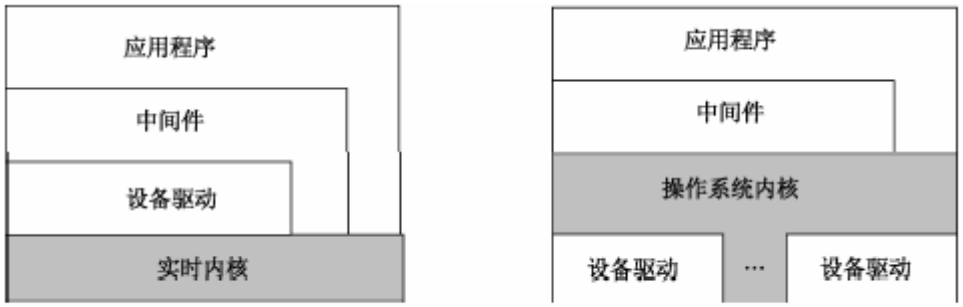


图 12-8　实时内核型的 RTEOS 与通用型的 RTEOS 的比较

1．嵌入式实时操作系统的关键特性

与通用操作系统相比，实时嵌入式操作系统在功能上具有很多特性。实时嵌入式操作系统特有的不同于通用操作系统的关键特性主要有：

满足嵌入式应用的高可靠性；

满足应用需要的可裁减能力；

内存需求少；

运行的可预测性；

采用实时调度策略；

系统的规模紧凑；

支持从 ROM 或 RAM 上引导和运行；

对不同的硬件平台均有更好的可移植性。

2．嵌入式实时操作系统的实时性能指标在评估实时操作系统设计性能时，时间性能指标是最重要的一个性能指标，常用的时

间性能指标主要有如下几个：

（1）任务切换时间：是指 CPU 控制权由运行态的任务转移给另外一个就绪任务所需要的时间，包括在进行任务切换时，保存和恢复任务上下文所花费的时间及选择下一个待运行任务的调度时间，该指标跟微处理器的寄存器数目和系统结构有关。相同的操作系统在不同微处理器上运行时所花费的时间可能不同。

任务切换时间所对应的时序图如图 12-9 所示。

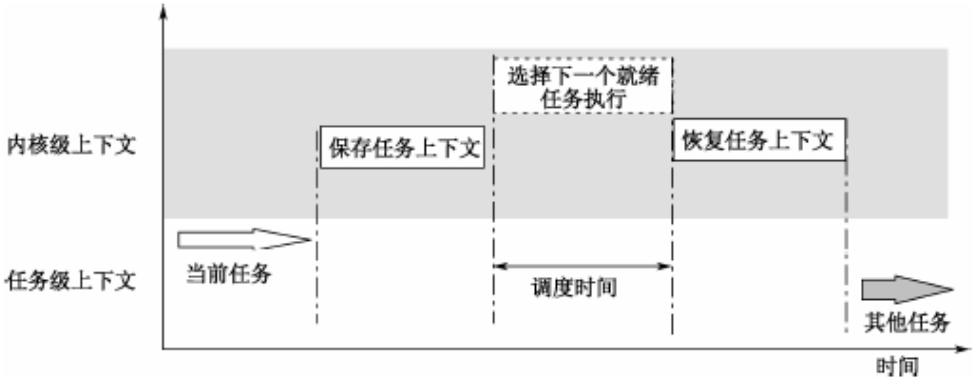


图 12-9　任务切换的时序

（2）中断处理相关的时间指标，对应的中断时序图如图 12-10 所示。

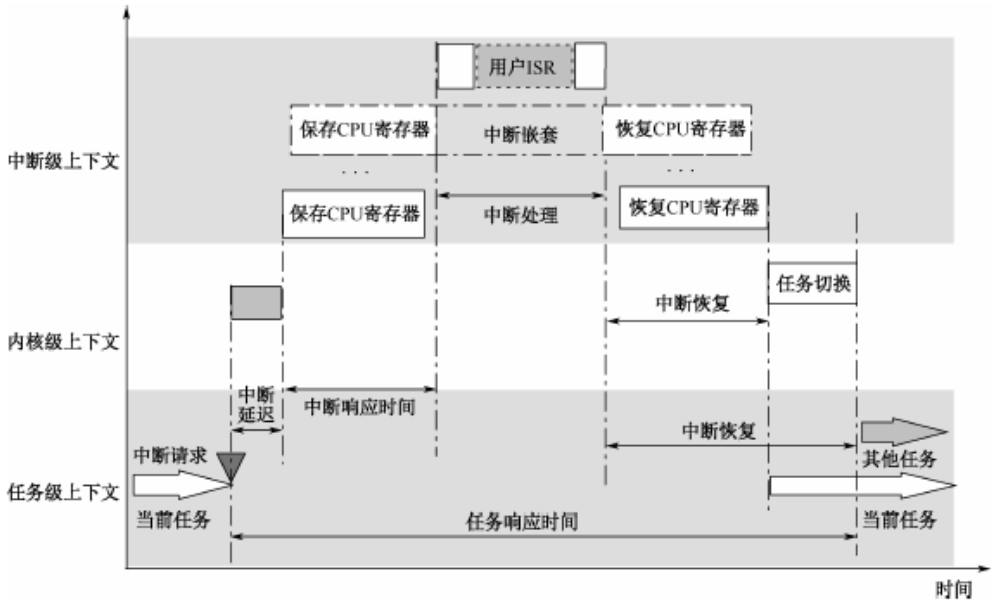


图 12-10　实时内核中断时序

中断延迟时间，是指从中断发生到系统获知中断的时间，主要受系统最大关中断时间的影响，关中断时间越长，中断延迟也就越长；

中断处理执行时间，该时间由具体的应用决定；

中断响应时间，是指从中断发生到开始执行用户中断服务例程的时间；

中断恢复时间，是指用户中断服务例程结束回到被中断的代码之间的时间；

最大关中断时间，包含两个方面：一是内核最大关中断时间，即内核在执行临界区代码时关中断；二是应用关中断时间，关中断最大时间是这两种关中断时间的最大值；

任务响应时间，是指从任务对应的中断产生到该任务真正开始运行的时间；

对于可抢占式调度，中断恢复的时间还要加上进行任务切换和恢复新的任务上下文的时间。

（3）系统响应时间：指系统在发出处理请求到系统做出应答的时间，即调度延迟，这个时间的大小主要由内核任务调度算法所决定。

作为总结，典型的可抢占实时内核的性能指标计算方法如表 12-3 所示。

表 12-3　可抢占实时内核的性能指标计算表

性能指标	抢占式内核
中断延迟时间	MAX（最长指令时间+用户中断禁止时间+内核中断禁用时间） +开始执行中断服务子程序的第一条指令的时间
中断响应时间	中断延迟+保存 CPU 内部寄存器的时间+内核的进入中断服务函数的执行时间
中断恢复时间	判断是否有更高优先级的任务进入了就绪态的时间+恢复那个更高优先级任务的 CPU 内部寄存器的时间+执行中断返回指令的时间
任务响应时间	找到最高优先级任务的时间+任务切换时间
RAM 大小	应用程序代码+内核 RAM+SUM（任务堆栈）+MAX（中断服务程序堆栈）

12.6.4 主流嵌入式操作系统介绍

迄今为止，据不完全统计，世界上现有的嵌入式操作系统的总数达几百个之多。其中最常用的有十几种，这些操作系统在各自的应用领域都有很高的知名度和广大的用户群。

表 12-4 选取了一些业界常见的嵌入式操作系统加以比较。

表 12-4　几种主流嵌入式操作系统介绍

名　称	简　介
ECOS	ECOS 是美国 Cygnus Solutions 公司开发的源代码开放的嵌入式操作系统，适用于深度嵌入式应用，主要用在信息电器上，如数字电视、冰箱、空调等
EPOC	EPOC 是 Psion Software 公司推出的一个 16/32 位多任务嵌入式操作系统，在移动计算设备中应用广泛，在 PDA 手机市场上占有相当的份额。目前支持 EPOC 的主要有 Ericsson、Motorola、Panasonic、Nokia、Psion PLC 等公司
IOS	IOS（Internet Operation System）是 Cisco 公司推出的一个专用嵌入式操作系统，主要用在网络交换机、路由器等网络设备上
LynxOS	Lynx Real-time Systems 开发的一个分布式、可扩展的嵌入式实时操作系统，有很高的市场占有率
Nucleus	Nucleus 是 Accelerated Technology 公司开发的一个嵌入式实时操作系统，主要用在消费电子、网络设备、无线、导航、办公设备、医疗设备和控制等领域。可以向用户开放源代码，在美国具有很高的市场占有率
OS-9	OS-9 是 Microware Systems 公司开发的一个嵌入式实时操作系统，其市场占有率很高，在国外排在前 10 名，主要用在高科技产品中，包括消费电子产品、工业自动化、无线通信产品、医疗仪器、数字电视，以及多媒体设备中。它提供了很好的安全和容错性能，与其他的嵌入式系统相比，更具灵活性
pSOS	pSOS 是 Integrated Systems 公司研发的一个产品，是世界上最早的实时系统之一，是一个模块化的操作系统，比较适用于深度嵌入式系统中。还配有一系列的基于 pSOS 的支撑软件，这些软件包括 TCP/IP 协议栈 pNA、远程过程调用库 pRPC、文件系统管理 pHILE 、ANSI C 标准库 pREPC、调试功能模块 pROBE 及信息系统实时分析工具 pMONT 等
QNX	QNX 是加拿大 QNX Software Systems Europe 公司研制的一个实时、可扩展的操作系统，并部分遵循 POSIX 相关标准，采用微内核结构。主要提供 4 种基本服务，所有的操作系统服务都是能互相通信的用户进程。目前，支持 X86、Power PC 、MIPS、ARM 等处理器，主要的应用领域是消费电子、电信、汽车及医疗设备等

续 表

名　称	简　介
VxWorks	VxWorks 操作系统是美国 WindRiver 公司于 1983 年设计开发的一种嵌入式实时操作系统，是 TornadoII 嵌入式开发环境的关键组成部分。良好的持续发展能力、高性能的内核及友好的用户开发环境，在嵌入式实时操作系统领域逐渐占据一席之地。首先，它十分灵活，具有多达 1800 个功能强大的应用程序接口（API）。其次，它适用面广，可以适用于从最简单到最复杂的产品设计。再次，它可靠性高，可以用于从防抱死刹车系统到星际探索的关键任务。最后，适用性强，可以用于所有流行的 CPU 平台
T-kernel	T-kernel 是日本坂村健教授用近 20 年时间研究开发的一款实时嵌入式操作系统，具有标准的开源结构，在嵌入式系统领域应用广泛，60%是基于 T-kernel 技术为基础的 OS

12.7 嵌入式系统开发设计

嵌入式系统设计的主要任务是定义系统的功能、决定系统的架构，并将功能映射到系统实现架构上。这里，系统架构既包括软件系统架构也包括硬件系统架构。一种架构可以映射到各种不同的物理实现，每种实现表示不同的取舍，同时还要满足某些设计指标，并使其他的设计指标也同时达到最佳化。

嵌入式系统的设计方法跟一般的硬件设计、软件开发的方法不同，是采用硬件和软件协同设计的方法，开发过程不仅涉及软件领域的知识，还涉及硬件领域的综合知识，甚至还涉及机械等方面的知识。要求设计者必须熟悉并能自如地运用这些领域的各种技术，才能使所设计的系统达到最优。

虽然嵌入式系统应用软件的设计方案随应用领域的不同而不同，但是嵌入式系统的分析与设计方法也遵循软件工程的一般原则，许多成 熟的分析和设计方法都可以在嵌入式领域得到应用。嵌入式系统的开发过程同样也包括需求分析、系统设计、实现和测试几个基本阶段，并且每个阶段都有其独有的特征和重点。

本节主要介绍嵌入式系统开发设计的技术与方法，并从嵌入式系统应用和计算模型的角度分析应用软件设计的方法及设计过程中面临的主要问题。最后，讨论嵌入式领域软件移植的相关问题。

12.7.1 嵌入式系统设计概述

进行嵌入式系统设计前，应明确嵌入式系统设计本身的特点及衡量嵌入式系统设计的一些主要的技术指标。

1．嵌入式系统设计的特点

与通常的系统设计相比，嵌入式系统设计具有以下特点：

软、硬件协同并行开发；

微处理器的类型多种多样；

实时嵌入式操作系统具有多样性；

与通用系统开发相比，可利用系统资源很少；

应用支持少；

要求特殊的开发工具；

软、硬件都要很健壮；

调试很困难。

2．嵌入式系统的技术指标

嵌入式系统设计的常用指标有：

（1）NRE 成本（非重复性工程成本）：设计系统所需要支付的一次性货币成本，即一旦设计完毕，不需要支付额外的设计费用，就可以制造任意数目的产品。

（2）单位成本：生产单个产品所需要支付的货币成本，不包含 NRE 成本。

（3）大小：指系统所占的空间，对软件而言，一般用字节数来衡量；对硬件而言，则用逻辑门或晶体管的数目来衡量。

（4）性能：系统完成规定任务所需要的时间，是设计时最常用的设计指标，主要有两种衡量方式，一是响应时间，即开始执行到任务结束之间的时间。二是完成量，即单位时间内所完成的任务量。

（5）功率：系统所消耗的功率，它决定了电池的寿命或电路的散热需求。

（6）灵活性：在不增加 NRE 成本的前提下，改变系统功能的能力。

（7）样机建立时间：建立系统可运行版本所需的时间，系统样机可能比最终产品更大更昂贵，但可以验证系统的用途和正确性，改进系统的功能。

（8）上市时间：从系统开发到可以上市卖给消费者的时间，最主要的影响因素包括设计时间、制造时间和检测时间。

（9）可维护性：系统推出或上市后进行修改的难易程度，特别是针对非原始开发人员进行的修改。

（10）正确性：正确实现了系统的功能，可以在整个设计过程中检查系统的功能，也可以插入测试电路检验是否正确。

（11）安全性：系统不会造成伤害的概率。各个设计指标之间一般是互相竞争的，改良了某个指标常常会导致其他指标的恶化，

为了最好地满足设计最佳化，设计者必须了解各种软、硬件的实现技术，并且能够从一种技术转移到另一种技术，以便找到特定约束下的最佳方案。

3．嵌入式系统的设计挑战

嵌入式系统设计所面临的挑战有以下几个方面。

（1）需要多少硬件：设计者对用于解决问题的计算能力有较强的控制能力，不仅可以选择使用何种处理器，而且可以选择存储器的数量、所使用的外设等，因为设计不仅要满足性能的需求，还要受到制造费用的约束，硬件的选择十分重要，硬件太少，将达不到功能和性能的要求，硬件过多又会使产品过于昂贵。

（2）如何满足时限：使用提高处理器速度的方法使程序运行速度加快来解决时间约束的方法是不可取的，因为这样会使系统的价格上升。同时，提高了处理器的时钟频率，有时并不能提高执行速度，因为程序的速度有可能受存储系统的限制。

（3）如何减少系统的功耗：对采用电池供电的系统，功耗是一个十分敏感的问题。对于非电池供电的系统，高功率意味着高散热。降低系统功耗的一种方法是降低它的运算速度，但是单纯地降低运算速度显然会导致性能不能满足，因此，必须认真设计在降低功耗的同时满足性能的约束。

（4）如何保证系统的可升级性：系统的硬件平台可能使用几代，或者使用同一代的不同级别的产品，这些仅需要一些简单的改变，设计者必须通过改变软件来改变系统的特性，设计一种机器使它能够提供现在仍未开发的软件的性能。

（5）如何保证系统的可靠性：可靠性是产品销售时一项重要的指标，产品能够很好地工作是消费者的合理要求，可靠性在一些系统中尤为重要，如安全控制系统。

（6）测试的复杂性：测试一个嵌入式系统比仅仅输入一些数据困难得多，所以不得不运行整台机器以产生正确的数据，数据产生的时间是十分重要的，即不能离开嵌入式系统工作的整个环境来测试嵌入式系统。

（7）可视性和可控制性有限：嵌入式系统通常没有显示设备和键盘，这将导致开发者很难了解系统内部发生了什么，也不能响应系统的动作，有时候不得不通过观察微处理器的信号来了解。在实时系统中，一般无法为了观察而让系统停机。

（8）开发环境受限：嵌入式系统的开发环境，如开发软件、硬件工具通常比通用计算机或工作站上的可用环境更为有限，故只能采用交叉式开发，给开发进度带来很大影响。

12.7.2 开发模型与设计流程

与通用系统的开发类似，嵌入式系统的开发也可以采用软件工程中常见的开发模型，主要包括瀑布模型、螺旋模型、逐步求精模型及层次模型。

1．常用开发模型

设计流程是系统设计期间应遵循的一系列步骤，其中一些步骤可以由自动化工具完成，而另外一些只可用手工完成。在嵌入式系统领域，有如下几种常用开发过程模型。

（1）瀑布模型。瀑布模型由五个主要阶段构成：需求分析阶段确定目标系统的基本特点；系统结构设计阶段将系统的功能分解为主要的构架；编码阶段主要进行程序的编写和调试；测试阶段检测错误；最后一个是维护阶段，主要负责修改代码以适应环境的变化，并改正错误、升级。各个阶段的工作和信息总是由高级的抽象到较详细的设计步骤单向流动，是一个理想的自顶向下的设计模型。

（2）螺旋模型。螺旋模型假定要建立系统的多个版本，早期的版本是一个简单的试验模型，用于帮助设计者建立对系统的直觉和积累开发此系统的经验，随着设计的进展，会创建更加复杂的系统。在每一层设计中，设计者都会经过需求分析、结构设计、测试三个阶段。在后期，当构成更复杂的系统版本时，每一个阶段都会有更多的工作，并需要扩大设计的螺旋，这种逐步求精的方法使设计者可以通过一系列的设计循环加深对所开发的系统的理解。螺旋的顶部第一个循环是很小很短的，而螺旋底部的最后的循环加入了对螺旋模型的早期循环的细节补充，螺旋模型比瀑布模型更加符合实际。

（3）逐步求精模型。逐步求精模型是一个系统被建立多次，第一个系统被作为原型，其后逐个将系统进一步求精。当设计者对正在建造的系统的应用领域不是很熟悉时，这个方法很有意义。通过建造几个越来越复杂的系统，从而精炼系统，使设计者能检验架构和设计技术。此外，各种迭代技术也可仅被局部完成，直到系统最终完成。

（4）层次模型。许多嵌入式系统本身是由更多的小设计组成的，完整的系统可能需要各种软件构件、硬件构件。这些部件可能由尚需设计的更小部件组成，因此从最初的完整系统设计到为个别部件的设计，设计的流程随着系统的抽象层次的变化而变化，从最高抽象层次的整体设计到中间抽象层次的详细设计，再到每个具体模块的设计，都是逐层展开的，其中每个流程可能由单个设计人员或设计小组来承担，每个小组依靠其他小组的结果，各个小组从上级小组获得要求，同时上级小组依赖于各个分组设计的质量和性能。而且，流程的每个实现阶段都是一个从规格说明到测试的完整流程。

2．嵌入式系统的设计方法

一个良好的嵌入式系统设计方法是十分重要的，这是因为：

（1）良好的设计方法可以使设计者清楚地了解他们所做工作的进度，这样可以确保不遗漏其中的任何一项工作。

（2）允许使用计算机辅助工具帮助设计者进行工作，将整个过程分成几个可控的步骤进行。

（3）良好的设计方法方便设计团队的成员之间相互交流，通过定义全面的设计过程，使团队里的每个成员可以很好地理解他们所要做的工作及完成分配给他们的任务时所达到的目标。

嵌入式系统软件的开发过程可以分为项目计划、可行性分析、需求分析、概要设计、详细设计、程序建立、下载、调试、固化、测试及运行等几个阶段。

项目计划、可行性分析、需求分析、概要设计及详细设计等几个阶段，与通用软件的开发过程基本一致，都可按照软件工程方法进行，如采用原型化方法、结构化方法等。

希赛教育专家提示：由于嵌入式软件的运行和开发环境不同，开发工作是交叉进行的，所以每一步都要考虑到这一点。

程序建立阶段的工作是根据详细设计阶段产生的文档进行的。这一阶段的工作主要是源代码编写、编译、链接等几个子过程，这些工作都是在宿主机进行的，不需要用到目标机。

产生应用程序的可执行文件后，就要用到交叉开发环境进行调试，根据实际情况可以选用可用的几种调试方法之一或它们的有效组合来进行。

嵌入式系统设计不同于传统的软件设计，如图 12-11 所示。经常包含硬件设计和软件设计，其中前端活动，如规格说明和系统架构，需要同时考虑硬件和软件两个方面。

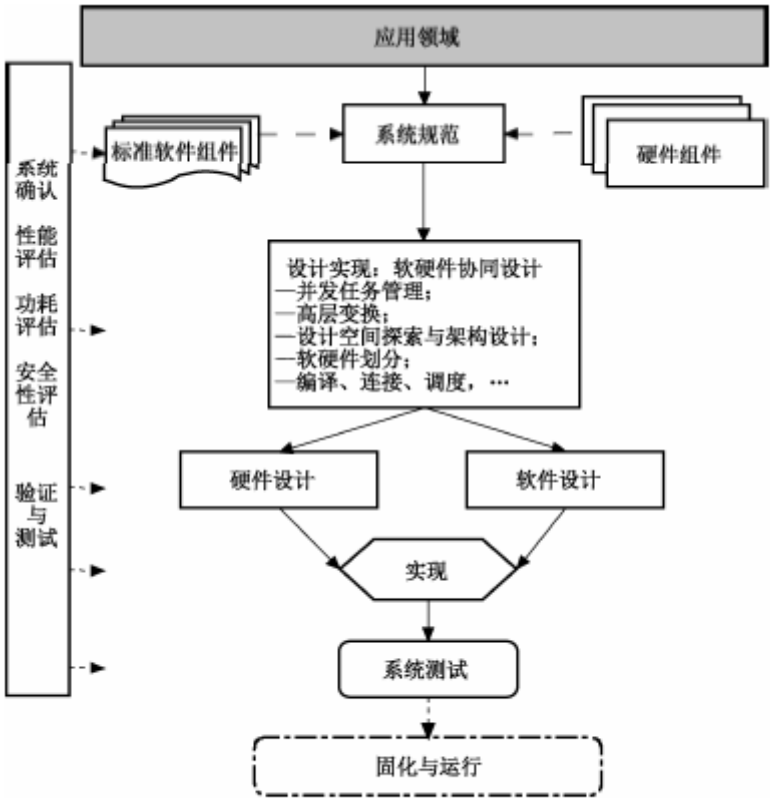


图 12-11　嵌入式系统常用的公共硬件/软件协同设计方法

类似的，后端设计，如系统集成和测试要考虑整个系统。在中间阶段中，软件和硬件构件的开发彼此相互独立，并且大多数的硬件和软件的工作能够相对独立地进行。最后，要将经调试后正确无误的可执行程序固化到目标机上。根据嵌入式系统硬件上配置的不同，固化有几种方式，可以固化在 EPROM 和 FLASH 等存储器中，也可固化在 DOC 和 DOM 等电子盘中。通常还要借助一些专用编程器进行。

由于嵌入式系统对安全性和可靠性的要求比通用计算机系统要高，所以在对嵌入式系统进行白盒测试时，要求有更高的代码覆盖率。

在系统开发流程的各个阶段，分别要进行系统的确认和性能评估、安全性评估及风险性评价，并对系统的实现进行测试验证。

12.7.3 嵌入式系统设计的核心技术

嵌入式系统的开发是软、硬件综合开发，与通用系统的开发存在巨大差异，一方面是因为每个嵌入式系统都是一个软硬件的结合体；另一方面，嵌入式系统一旦研制完成，软件便随着硬件固化到产品中，具有很强的专用性。在这些特点的影响下，必然要有一种不同于通用软件开发过程的工程方法学来支持嵌入式系统的开发过程，同时，这些特点也决定了嵌入

式系统开发所采用的独特的核心技术。

总体来看，在嵌入式开发领域，主要有三种核心技术：处理器技术、IC 技术、设计/ 验证技术。

1.处理器技术

处理器技术与实现系统功能的计算引擎结构有关，很多不可编程的数字系统也可以视为处理器，这些处理器的差别在于其面向特定功能的专用化程度，导致其设计指标与其他处理器不同。

（1）通用处理器。这类处理器可用于不同类型的应用，一个重要的特征就是存储程序，由于设计者不知道处理器将会运行何种运算，所以无法用数字电路建立程序。另一个特征就是通用的数据路径，为了处理各类不同的计算，数据路径是通用的，其数据路径一般有大量的寄存器及一个或多个通用的算术逻辑单元。设计者只需要对处理器的存储器编程来执行所需的功能，即设计相关的软件。

在嵌入式系统中使用通用处理器具有设计指标上的一些优势。上市时间和 NRE 成本较低，因为设计者只需编写程序，而不需做任何数字设计，灵活性高，功能的改变通过修改程序进行即可。与自行设计处理器相比，数量少时单位成本较低。

当然，这种方式也有一些设计指标上的缺陷，数量大时单位成本相对较高，因为数量大时，自行设计的 NRE 成本分摊下来，可降低单位成本。同时，对于某些应用，性能可能很差。由于包含了非必要的处理器硬件，系统的体积和功耗可能变大。

（2）单用途处理器。单用途处理器是设计用于执行特定程序的数字电路，也指协处理器、加速器、外设等。如 JPEG 编码解码器执行单一程序，压缩或解压视频信息。嵌入式系统设计者可通过设计特定的数字电路来建立单用途的处理器。设计者也可以采用预先设计好的商品化的单用途处理器。

在嵌入式系统中使用单用途处理器，在指标上有一些优缺点。这些优缺点与通用处理器基本相反，性能可能更好，体积与功率可能较小，数量大时单位成本可能较低，而设计时间与 NRE 成本可能较高，灵活性较差，数量小时单位成本较高，对于某些应用，性能不如通用处理器。

（3）专用处理器。专用指令集处理器是一个可编程处理器，针对某一特定类型的应用进行最优化。这类特定应用具有相同的特征，如嵌入式控制、数字信号处理等。在嵌入式系统中使用专用处理器可以在保证良好的性能、功率和大小的情况下，提供更大的灵活性，但这类处理器仍需要昂贵的成本建立处理器本身和编译器。单片机和数字信号处理器是两类应

用广泛的专用处理器，数字信号处理器是一种针对数字信号进行常见运算的微处理器，而单片机是一种针对嵌入式控制应用进行最佳化的微处理器。

2．IC 技术

从系统的集成电路设计描述得到实际芯片的物理映射过程的实现技术便是 IC（Integrated Circuits，集成电路）技术，当前在半导体领域的三类实现技术，即全定制、半定制和可编程技术均可应用于嵌入式系统的硬件设计。

（1）全定制/VLSI（Very Large Scale Integrated Circuites，超大规模集成电路）。在全定制 IC 技术中，需要根据特定的嵌入式系统的数字实现来优化各层设计人员从晶体管的版图尺寸、位置、连线开始设计以达到芯片面积利用率高、速度快、功耗低的最优化性能。利用掩膜在制造厂生产实际芯片，全定制的 IC 设计也常称为 VLSI，具有很高的 NRE 成本、很长的制造时间，适用于大量或对性能要求严格的应用。

（2）半定制/ASIC（Application Specific Integrated Circuit，专用集成电路）。半定制 ASIC 是一种约束型设计方法，包括门阵列设计法和标准单元设计法。它是在芯片制作好一些具有通用性的单元元件和元件组的半成品硬件，设计者仅需要考虑电路的逻辑功能和各功能模块之间的合理连接即可。这种设计方法灵活方便、性价比高，缩短了设计周期，提高了成品率。

（3）可编程/ASIC。可编程器件中所有各层都已经存在，设计完成后，在实验室里即可烧制出设计的芯片，不需要 IC 厂家参与，开发周期显著缩短。可编程 ASIC 具有较低的 NRE 成本，单位成本较高，功耗较大，速度较慢。

3．设计/验证技术

嵌入式系统的设计技术主要包括硬件设计技术和软件设计技术两大类。其中，硬件设计领域的技术主要包括芯片级设计技术和电路板级设计技术两个方面。

芯片级设计技术的核心是编译/综合、库/IP（Intellectual Property，知识产权）、测试/ 验证。编译/综合技术使设计者用抽象的方式描述所需的功能，并自动分析和插入实现细节。库/IP 技术将预先设计好的低抽象级实现用于高级抽象。测试/验证技术确保每级功能正确，减少各级之间反复设计的成本。

软件设计技术的核心是软件语言。软件语言经历了从低级语言（机器语言、汇编语言）到高级语言（例如，结构化设计语言、面向对象设计语言）的发展历程，推动其发展的是汇编技术、分析技术、编译/解释技术等诸多相关技术。软件语言的级别也从实现级、设计级、功能级逐渐向需求级语言发展过渡。

早期，随着通用处理器概念的逐渐形成，软件技术迅速发展，软件的复杂度也开始增加，软件设计和硬件设计的技术和领域完全分开。设计技术和工具在这两个领域同步得到发展，也使得行为描述可以在越来越抽象的级别上进行，以适应设计复杂度不断增长的需要。这种同步发展如今又使得这两个领域都使用同样的时序模型来描述行为，因而这两个领域即将可能再度统一为一个领域。

鉴于大多数嵌入式系统都是实时的反应式系统，反应式系统具有多任务并发、时间约束严格与可靠性高的特点，针对反应式系统的设计和描述，人们相继提出了多种描述语言和验证方法学。例如，采用时序逻辑用来刻画反应式系统的性质及推理反应式系统的行为，采用模型检验技术验证反应式系统设计的正确性等，这些技术已逐步在嵌入式开发过程中发挥着重要的作用。

12.7.4 嵌入式开发设计环境

嵌入式系统的开发环境种类很多，大体可以把它们分为如下几类：

（1）与嵌入式操作系统配套的开发环境，属于这一类的开发环境较多，如 PalmOS、THOS、VxWorks、Windows CE 等商业嵌入式操作系统都有与其配套的功能齐全的开发环境。

（2）与处理器芯片配套的开发环境。这类开发环境一般由处理器厂商提供，如 EPSON 公司推出的一个专门为基于 S1C33 系列微控制器芯片的嵌入式系统开发的工具包便是这一类型的开发环境。

（3）与具体应用平台配套的开发环境。这类开发环境针对性较强，如高通公司的 Brew SDK 等。

（4）其他类的开发环境。这类开发环境主要指一些嵌入式系统供应商在 GNU 开源工具的基础上开发或定制的较为通用的开发环境。这类工具可以免费获得，而且支持的处理器类型繁多，功能齐全，但在技术支持方面比专业化商业工具略逊一些。

12.7.5 嵌入式软件设计模型

随着嵌入式系统的功能日益复杂，要描述这些功能复杂的系统的行为也越来越困难，实践证明通过采用计算模型的方法来对系统进行描述和分析是一种具有工程价值的方法。

本节介绍几种嵌入式领域常用的计算模型，并从计算模型的角度分析和阐述嵌入式应用设计和开发的相关问题。计算模型提供一组用简单对象来组合复杂行为的方法，可以帮助设

计者理解和描述系统行为。嵌入式系统常用的计算模型有如下几种：时序计算模型、通信进程模型、状态机模型、数据流模型、面向对象模型、并发进程模型。

这些模型分别在不同的应用领域使用，如状态机模型特别适合描述以控制为主的系统，数据流模型可以很好地描述数据处理和转换问题。目前使用最广泛的是并发进程模型。

1．状态机模型

有限状态机（Finite-State Machine，FSM）是一个基本的状态模型，可以用一组可能的状态来描述系统的行为，系统在任何时刻只能处于其中一个状态，也可以描述由输入确定的状态转移，最后可以描述在某个状态下或状态转移期间可能发生的操作。

有限状态机 FSM 是一个六元组 F<S，I，O，F，H，S0>，其中 S 是一个状态集合{s0，s1，…，sl}，I 是输入集合{I0，I1，…，Im}，O 是输出集合{o0，o1，…，on}，F 是次态函数或转移函数，将状态和输入映射到状态(S×I→S)，H 是输出函数，将状态映射到输出 (S→O)，S0 是初始状态。

图 12-12 是电梯的控制单元的状态机描述。在初始“空闲”态，将 up 和 down 设置为 0， open 设置为 1。在所请求的楼层不同于当前楼层之前，状态机一直停留在“空闲”状态。如果所请求的楼层大于当前楼层，则状态机转移到“上升”状态，并将 up 设置为 1。如果所请求的楼层小于当前楼层，则状态机转移到“下降”状态，并将 down 设置为 1。在当前楼层等于所请求的楼层之前，状态机一直留在“下降”或“上升”状态，然后状态转移到“开门”状态，并将 open 设置为 1。通常，系统有一个计时器 timer，因此，当状态机转移到“开门”状态时，还要将计时器启动，状态机停留在“开门”态，直到计时器超时，最后转移到“空闲”态。

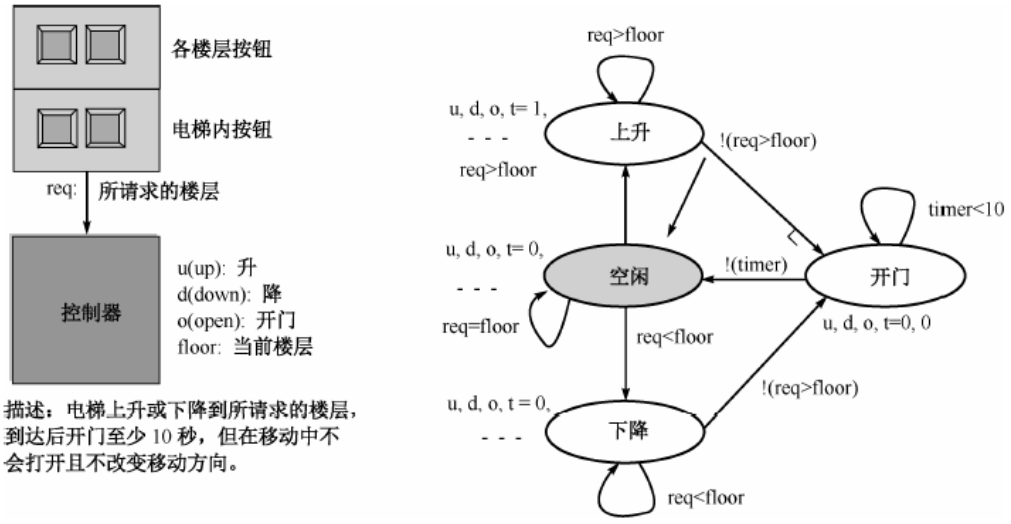


图 12-12　电梯控制器的状态转移描述

当 FSM 被用于嵌入式系统设计时，其输入和输出的数据类型都是布尔类型，而函数表示含有布尔运算的布尔函数，这种模型对于没有数据输入或输出的很多纯控制系统而言已经足够。如果要处理数据，则将 FSM 扩展为带有数据路径的状态机（FSM with Datapath，FSMD）。另外，对状态机模型可以进一步扩展以支持分级和并发，这种模型称为分级/并发 FSM（Hierarchical/Concurrent FSM，HCFSM）模型。

2．数据流模型

数据流模型是并发多任务模型派生出的一种模型，该模型将系统的行为描述为一组结点和边，其中结点表示变换，边表示从一个结点到另一个结点的数据流向。每个结点使用来自其输入边的数据，执行变换并在其输出边上产生数据。

每条边可能有或没有数据，出现在边上的数据称为令牌，当某个结点的所有输入边都至少有一个令牌时，该结点可触发。结点触发后，将使用来自每条输入边的一个令牌，对所有使用的令牌进行数据变换，并在输出边上产生一个令牌，结点的触发仅决定于令牌出现的情况。

图 12-13 所示是计算 z=(a+b)×(c-d)的数据流模型。

目前，已有若干商业化的工具支持用图形化语言表达数据流模型，这些工具可以自动将数据流模型转换为并发多任务模型，以便在微处理器上实现。其转换方法为将每个结点转换为一个任务，每条边转换为一个通道，其中并发多任务模型的实现方法是使用实时操作系统对并发任务进行映射。

图 12-14 是一个同步数据流模型，这个模型中，在结点的每条输入边和输出边上分别标注每次触发所使用和产生的令牌数。该模型的优点是，在实现时不需要将其转换为并发多任务模型，而是用静态方式调度结点，产生时序程序模型。该模型可以使用时序程序语言（如 C 语言）来表达，不需要实时操作系统就可以执行，因此其执行效率更高。

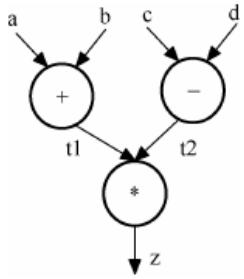


图 12-13　表示算术变换的数据流模型

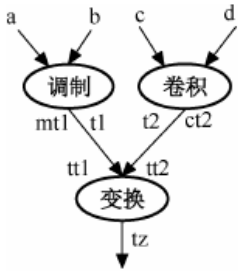


图 12-14　同步数据流模型

3.并发进程模型

并发进程模型是由一组进程构成，每个进程是一个顺序执行的过程，各进程间可以并发

执行。并发进程模型提供创建、终止、暂停、恢复和连接进程的操作。进程在执行中可以相互通信，交换数据。进程间通信可以采用两种方式：共享变量和消息传递。信号量、临界区、管程和路径表达式等用来对并发进程的操作进行同步。

通常，实时系统可以看成是由许多并发执行的进程构成的系统，其中每个进程都有时间要求。这样，很多嵌入式系统更容易用一组并发执行的任务来描述，因为这些系统本身就是多任务系统，并发进程模型便自然地可以由实时操作系统的多任务来实现。

4．面向对象模型

传统的并发进程模型是围绕进程的概念进行设计的，进程是一个实现级的概念，它是对客观世界活动的一种间接模拟，因此，采用进程模型来解决客观世界中的并发问题就显得极不自然，并且也使得并发程序难以设计和理解。

面向对象模型以一种更加直接的方式刻画客观世界中的活动，模型中存在着潜在的并发执行能力。一个对象向另一个对象发送消息后，若不需要或不立即需要消息的处理结果，前者不必等待后者处理消息，消息发送者和消息接受者可以并发执行。对象不都是处于被动的提供服务状态，它们中的一些除了能通过接收消息向外提供服务外，还可以有自己的事务处理。一个对象往往可以同时处理多个消息。

对象是数据和操作的封装体，数据存放在对象的局部变量中，对象的状态由对象所有的局部变量在某一时刻的取值来表示。在并发环境中，还要考虑对象并发状态的描述问题，因为对象的并发控制是根据对象的并发状态来进行的。

把并发与面向对象相结合，归结起来可分为两条途径：

（1）在面向对象模型中引进并发机制，充分利用面向对象技术刻画客观世界的良好模型能力和面向对象的各个重要特性，同时把其潜在的并发能力描述出来，使其适合于描述并发计算。

（2）在传统并发模型中引进面向对象思想。

面向对象的并发模型可以分为两种类型：隐式并发模型和显式并发模型。

（1）隐式并发模型。这种模型的特点是推迟并发设计，将对象建模作为建模基础。在进入运行阶段之前，将对象看成自主单元，各种对象的活动看成理想并发方式完成的特定工作。就像每个对象拥有一个自己的处理器，这个处理器可以为对象提供一个执行线程。进入系统的外部事件被看成一个处理请求，以广播方式传给一些对象，这些对象接着向其他对象进一步提出处理请求。理论上，对应一个请求，可以有任意多个对象执行相应的处理。在实现时，由调度程序最终决定其对象的操作顺序，如图 12-15 所示。

（2）显式并发模型。这种模型的特点是首先考虑并发，应先把并发概念和对象概念分开。在建立对象以后，用实时操作系统支持的进程概念来表示并发，形成对象和进程两个抽象层次，即先将系统分解为准并发进程作为开始，而在每个进程的内部采用面向对象的技术。对象间交互表示成嵌套的函数调用，通过加入锁、监视器、信号量等显式同步机制，来保证对象的完整。该模型将进程置于对象之上，对象中不必考虑并发、对象串行化，如图 12-16 所示。

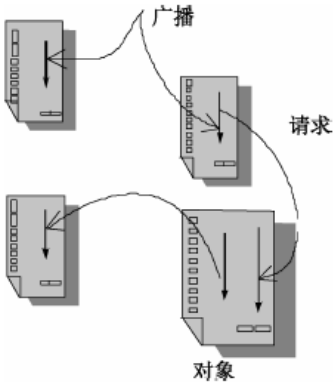


图 12-15　隐式并发模型

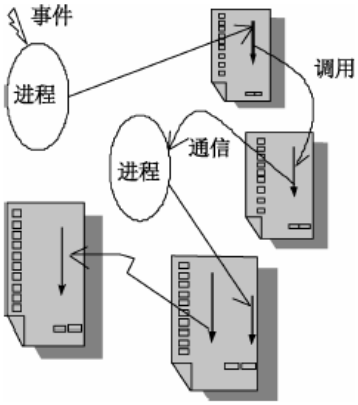


图 12-16　显式并发模型

早期，实时系统的设计方法主要是结构化设计方法，采用结构化方法的系统在复用性、可修改性等方面有很大的局限性。面向对象的实时系统设计方法显然在这些问题上具有明显的优势。较实用的面向对象的设计方法是诺基亚公司的 OCTOPUS 方法，该方法以 OMT 和融合方法（Fusion Method）为基础，提出了对实时系统响应时间、时间域及并发的处理方法，并具体提出了对并发、同步、通信、中断处理、ASIC、硬件界面、端对端响应时间等方面的处理。OCTOPUS 方法将软件开发的主要阶段很好地合并起来，从规格说明到运行模型之间的过渡紧密自然，还支持渐进式开发。OCTOPUS 方法是当前面向对象技术和实时系统相结合的一个典型的设计方法。另外，形式化的面向对象的开发技术和建模语言也逐渐在实时系统建模的初始阶段得到应用。

12.7.6 需求分析

在设计之前，设计者必须知道要设计什么。通常人们用需求和规格说明来描述设计过程的这两个相关而不同的步骤。需求是用户所想要的非形式化的描述，而规格说明是可以用来创建系统架构的更详尽、更精确、更一致的描述。当然，需求和规格说明都是指导系统的外部表示，而非内部表示。需求有两种类型：功能性需求和非功能性需求，功能性需求说明这

个系统必须做什么，而非功能性需求说明系统的其他属性，如物理尺寸、价格、功耗、设计时间、可靠性等。

对一个大系统进行需求分析是一项复杂而费时的工作，但是，获取少量格式清晰、简单明了的信息是理解系统需求的一个良好开端。表 12-5 是在某项工程开始时填写的需求表格，在考虑系统的基本特征时可将该表格作为检查表。

表 12-5　GPS 移动地图系统的需求表格

名　　称	说　　明	示例：GPS 移动地图系统
目的	该项可以简单地列举一些有关将要满足的需求描述或主要特征	GPS 移动地图系统为车辆驾驶人员提供用户级移动地图
输入与输出	这两项内容复杂，对系统输入/输出包含了大量的细节： 数据类型：模拟信号、数字信号、机械输入； 数据特征：周期性到达的数据，每个数据元素的位数； 输入/输出设备的类型：按键、模数转换器、视频显示	输入：一个电源按钮，两个控制按钮 输出：逆光液晶显示屏，分辨率 400×600
功能	该项对系统所做的工作详细地描述，从输入到输出进行分析是提出功能的一种好方法：当系统接收到输入时，执行哪些动作，输入的数据如何对该功能产生影响；不同的功能之间如何相互作用	使用 5 种接收器的 GPS 系统，三种用户可选的分辨率，总是显示当前的经纬度
性能	许多嵌入式系统都要花费一定的时间来控制物理设备，或从外界输入数据。多数情况下，这些计算必须在一定的时间内完成，因此对性能的要求必须尽早明确	每 0.25s 更新一次屏幕显示
生产成本	主要指硬件的费用	100 美元
功耗	该项对功耗做一个粗略的估计，靠电池供电的系统需认真考虑	100mW
物理尺寸和重量	对物理尺寸和质量有一定的了解有助于对系统架构的设计	不大于 2 英寸×16 英寸，12 盎司

这份需求表格内容是以 GPS（Global Position System，移动地图系统）为例编写的。移动地图系统是一种手持设备，针对在高速公路开车的用户或类似的用户而设计，该设备可从 GPS 上得到位置信息，为用户显示当前所在的位置及周围的地形图，地图的内容随着用户及设备所在位置的改变而改变。

需求分析阶段最重要的文档输出就是系统的规格说明。

规格说明是精确反映客户需求并且作为设计时必须遵循的要求的一种技术文档。在软件开发的过程中，规格说明非常重要。系统分析人员接受用户需求产生目标软件系统的规格说明，设计与编码人员根据规格说明，进行模块设计并最终产生程序代码，测试和验收人员验证最终软件是否符合规格说明。规格说明应该是清晰的、无歧义的，否则由该规格说明建造系统可能不符合实际要求。

目前，业界较为流行的方法是采用 UML 进行规格说明的描述。UML 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行建模。UML 适用于系统开发过程中从需求规格描述到系统完成后测试的不同阶段。

图 12-17 是一个显示操作的状态机规格说明示例，开始和结束是特殊的状态，状态机

中的状态代表了不同的概念性操作。

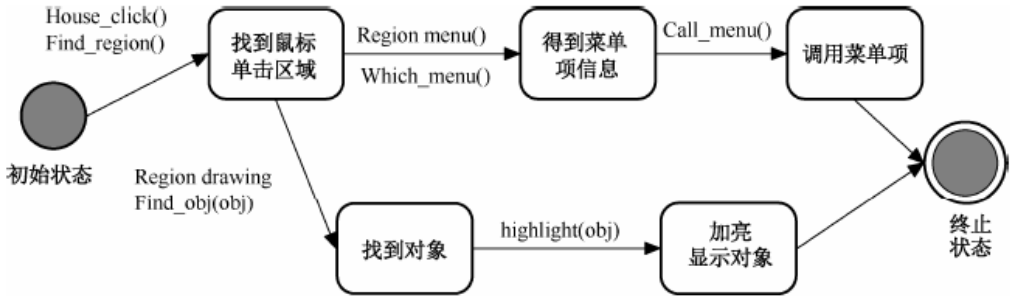


图 12-17　显示操作的状态机规格说明

在需求分析阶段，通过用例来捕获用户需求。通过用例建模，描述对系统感兴趣的外部角色及其对系统（用例）的功能要求。分析阶段主要关心问题域中的主要概念（如抽象、类和对象等）和机制，需要识别这些类及它们相互间的关系，并用 UML 类图来描述。在分析阶段，只对问题域的对象（现实世界的概念）建模，而不考虑定义软件系统中技术细节的类（如处理用户接口、数据库、通信和并行性等问题的类）。

12.7.7 系统设计

目前，嵌入式系统的设计工具可以分为两类：协同合成工具和协同模拟工具。

（1）协同合成工具。当前，用于嵌入式开发的主要的协同合成工具有 POLIS、COSYMA 和 Chinook 等。

POLIS：POLIS 是 UC-Berkeley 开发的交互式嵌入式系统的软、硬件协同设计框架，它适用于小型控制系统的设计，系统描述支持基于 FSM（Finite State Machine）的语言。由于软、硬件均可透明地从同一 CFSM 描述中取得，设计空间的灵活性也相应增加，支持使用 PTOLEMY 的协同模拟，在描述及实现层均支持正式的验证，架构的支持受限，即硬件 CFSMs 所包围的只有一个处理器，而且不支持共享内存。

COSYMA：COSYMA 是由德国 IDA 公司开发的一种探索硬件与软件协同设计合成进程的平台，它面向软件系统的描述较简单，支持自动分割和协同处理器合成，在合成时期可以对设计空间进行探索，系统合成取决于硬件限制，不支持并发模块，即一次只能有一个线程执行，架构同样受限，不支持正式验证，设计的成功与否取决于分割及开销估计技术。

Chinook：Chinook 是为控制系统而设计的，整个系统的描述作为一个输入提供给 Chinook，它的内部模式基于类似等级状态的模式，它不对代码进行分割，它为整个设计提供单一的模拟环境，Chinook 支持多种系统架构，尤其是多处理器结构。同样支持定时限制的描述，它

能合成多种接口，包括系统之间的软、硬件接口，能直接从定时图表中合成设备驱动器，可以控制处理器之间的通信。

（2）协同模拟工具。协同模拟是嵌入式系统设计中至关重要的一个方面，在整个系统设计完成后，在统一框架下模拟不同种类的成分是必要的，协同模拟不仅提供检验，而且为用户提供各系统的性能信息，这有助于在系统的早期提出变更方案，不至于造成重大损失。目前，主要的协同模拟工具有如下两种。

PTOLEMY：PTOLEMY 的关键思想是混合使用面向对象内核的计算模型，可用于模拟多种的系统，在各种应用中被广泛地使用，但不适合于系统合成，硬件模拟也是它的一项功能。

TSS：TSS（Tool for System Simulation）是模拟复杂硬件的工具，采用 C 语言编写，单个模块的提取可由用户控制，可以方便地进行添加与删除模块。但不支持分级模块，没有用于同步各处理器存取共享数据结构的机制，模块间的通信通过端口和总线进行。并且，TSS 支持多核系统的模拟。

1．系统架构设计

描述系统如何实现规格说明中定义的功能是系统架构设计的主要目的。但是在设计嵌入式系统的系统结构时，很难将软件和硬件完全分开。通常的处理是先考虑系统的软件架构，然后再考虑其硬件实现。系统结构的描述必须符合功能上和非功能上的需求。不仅所要求的功能要体现，而且成本、速度、功耗等非功能约束也要满足。从系统原始框图中的功能元素开始逐个考虑和细化，把原始框图转化为软件和硬件系统结构的同时考虑非功能约束，是一个切实可行的方法。下面以 GPS 移动地图系统的架构设计为例进行说明。

（1）原始框图。如图 12-18 所示，这个原始框图是移动地图系统的主要操作和数据流。

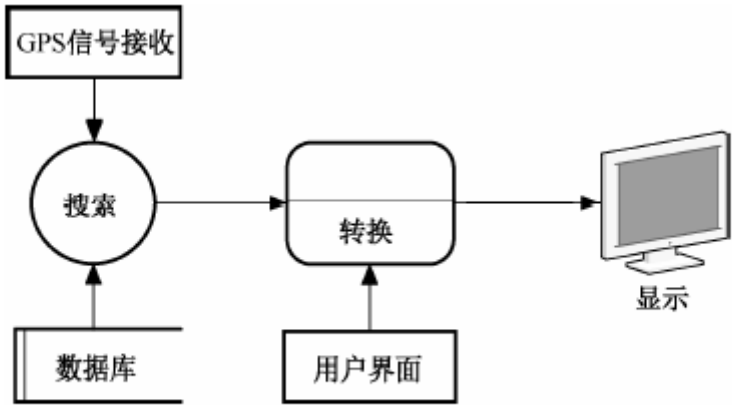


图 12-18　移动地图系统原始框图

（2）软件系统架构。如图 12-19 所示，软件系统主要由用户界面、数据库搜索引擎和数据转换器组成。

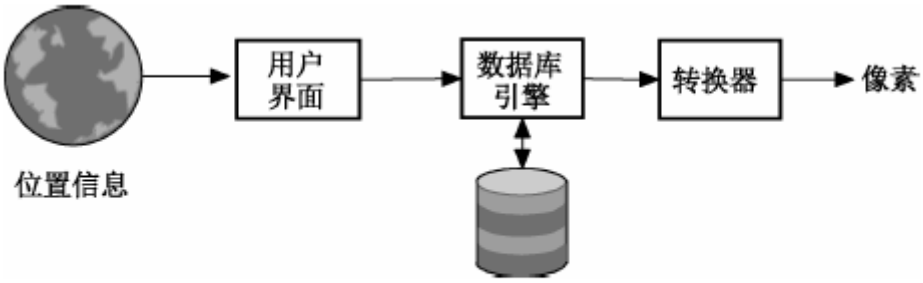


图 12-19　GPS 软件系统架构

（3）硬件系统架构。如图 12-20 所示，硬件系统采用通用微处理器、存储器和 I/O 设备组成。本系统选用两种存储器：通用数据、程序存储器和针对像素显示的帧缓冲存储器。

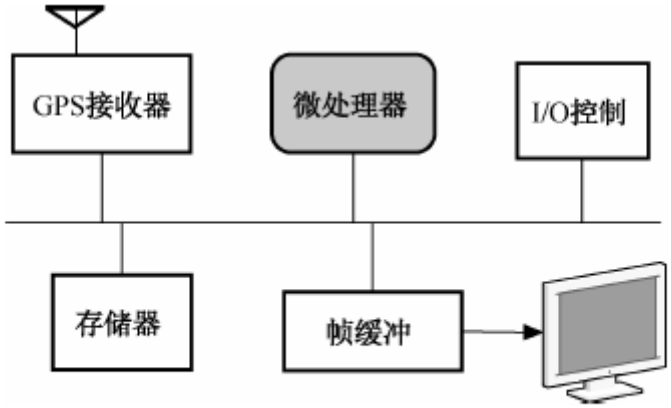


图 12-20　GPS 硬件系统架构

2．硬件子系统设计

嵌入式系统的开发环境由 4 部分组成：目标硬件平台、嵌入式操作系统、编程语言和开发工具，其中处理器和操作系统的选择应当考虑更多的因素，避免错误的决策影响项目的进度。

（1）选择处理器技术。嵌入式系统设计的主要挑战是如何使互相竞争的设计指标同时达到最佳化。设计者必须对各种处理器技术和 IC 技术的优缺点加以取舍。一般而言，处理器技术与 IC 技术无关，也就是说，任何处理器技术都可以使用任何 IC 技术来实现，但是最终器件的性能、NRE 成本、功耗、大小等指标会有很大的差异，如图 12-21 所示。

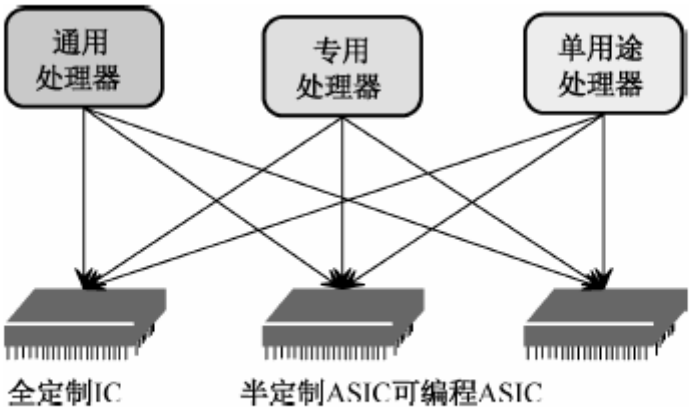


图 12-21　处理器技术和 IC 技术的关系

更通用的可编程技术提供了较大的灵活性，降低了 NRE 成本，建立产品样机与上市的时间较快。定制的技术能够提供较低的功耗、较好的性能、更小的体积和大批量生产时的低成本。

通常，一个公司要推出一种产品，如机顶盒、家庭路由器或通用处理器等，可以先推出半定制产品，以尽快占领市场，然后再推出全定制的产品。也可先用较可靠的老技术实现处理器，再用新制程的技术实现下一代。同样，嵌入式系统的设计者可以使用可编程的器件来建立样机，以加速上市时间，批量时再采用定制器件。

根据这些原则，设计者便可以对采用的处理器技术和处理器做出合理选择。一般，全定制商品化的“通用处理器　软件”是大多数情况下都适用的一个选择。

（2）通用嵌入式处理器的选择。根据用户的需求和项目的需要选择合适的通用嵌入式处理器，选择时需要考虑如下指标。

处理器的速度。一个处理器的性能取决于多个方面的因素：时钟频率，内部寄存器的大小，指令是否对等处理所有的寄存器等。对于许多需用处理器的嵌入式系统设计来说，目标不是在于挑选速度最快的处理器，而是在于选取能够完成作业的处理器和 I/O 子系统。处理器的性能满足系统的需求，并有一定的余量，但也不必选得太高。

技术指标。当前，许多嵌入式处理器都集成了外围设备的功能，从而减少了芯片的数量，进而降低了整个系统的开发费用。开发人员首先考虑的是，系统所要求的一些硬件能否无须过多的组合逻辑就可以连接到处理器上。其次是考虑该处理器的一些支持芯片，如 DMA 控制器、内存管理器、中断控制器、串行设备、时钟等的配套。

开发人员对处理器的熟悉程度，即项目的开发人员需要在处理器本身的成本和开发成本之间做一个权衡。

处理器的 I/O 功能是否满足系统的需求，即许多处理器提供内置的外部设备，以减少芯片数量、降低成本，应尽量考虑这种方案。

处理器的相关软件支持工具，即该款处理器是否具有完善的嵌入式操作系统、编程语言和开发工具的支持等。

处理器的调试，即处理器是否集成了调试功能，如是否支持 JTAG、BDM 等调试方式。

处理器制造商的支持可信度。在产品的生命周期里选择某种处理器时，设计者必须确认它有足够的供货量、技术支持等处理器的低功耗。

嵌入式微处理器最大并且增长最快的市场是手持设备、电子记事本、PDA、手机、GPS 导航器、智能家电等消费类电子产品，这些产品中选购的微处理器的典型特点是要求高性能、低功耗。许多 CPU 生产厂家已经进入了这个领域。

（3）硬件设计的注意事项。首先，将硬件划分为部件或模块，并绘制部件或模块连接框图。其次，对每个模块进行细化，把系统分成更多个可管理的小块，可以被单独实现。通常，系统的某些功能既可用软件实现也可用硬件实现，没有一个统一的方法指导设计者决定功能的软硬件分配，但是可以根据约束清单，在性能和成本之间进行权衡。

设计软、硬件之间的接口时，需要硬件设计者和软件设计者协同工作才能完成，良好的接口设计可以保证硬件简洁、易于编程。

设计时需要注意以下几点。

I/O 端口：列出硬件的所有端口、端口地址、端口属性、使用的命令和序列的意义、端口的状态及意义。

硬件寄存器：对每个寄存器设计寄存器的地址、寄存器的位地址和每个位表示的意义，以及对寄存器读写的说明、使用该寄存器的要求和时序说明。

内存映射：共享内存和内存映射 I/O 的地址，对每个内存映射，说明每个 I/O 操作的读/写序列、地址分配。

硬件中断：如何使用硬件中断，列出所使用的硬件中断号和分配的硬件事件。

存储器空间分配：列出系统中程序和数据占用的空间大小、位置，以及存储器类型和访问方式等。

总之，硬件设计者应该给软件设计者更多、更详细的信息，以便于进行软件设计和开发。

3．软件子系统设计

根据需求分析阶段的规格说明文档，确定系统计算模型，对软件部分进行合理的设计即可。

（1）操作系统的选择。在选择嵌入式操作系统时，需要做多方面的考虑：

操作系统的功能。根据项目需要的操作系统功能来选择操作系统产品，要考虑系统是否支持操作系统的全部功能或部分功能，是否支持文件系统、人机界面，是实时系统还是分时系统及系统是否可裁减等因素。

配套开发工具的选择。有些实时操作系统（rtos）只支持该系统供应商的开发工具。也就是说，还必须向操作系统供应商获取编译器、调试器等。有些操作系统使用广泛且有第三方工具可用，因此，选择的余地比较大。

操作系统的移植难易程度。操作系统到硬件的移植是一个重要的问题。它是关系到整个系统能否按期完工的一个关键因素，因此要选择那些可移植性程度高的操作系统，从而避免操作系统难以向硬件移植而带来的种种困难，加速系统的开发进度。

操作系统的内存需求如何。均衡考虑是否需要额外 ram 或 eeprom 来迎合操作系统对内存的较大要求。有些操作系统对内存的要求是与目标相关的。如 tornado/vxworks，开发人员能按照应用需求分配所需的资源，而不是为操作系统分配资源。从需要几 k 字节存储区的嵌入设计到需求更多的操作系统功能的复杂的高端实时应用，开发人员可任意选择多达 80 种不同的配置。

操作系统附加软件包。是否包含所需的软件部件，如网络协议栈、文件系统、各种常用外设的驱动等。

操作系统的实时性如何。实时性分为软实时和硬实时。有些嵌入式操作系统只能提供软实时性能，如 microsoft windows ce 2.0 是 32 位，windows 兼容，微内核，可伸缩实时操作系统，可以满足大部分嵌入式和非嵌入式应用的需要。但实时性不够强，属于软实时嵌入式操作系统。

操作系统的灵活性如何。操作系统是否具有可剪裁性，即能否根据实际需要进行系统功能的剪裁。有些操作系统具有较强的可剪裁性，如嵌入式 linux 、 tornado/vxworks 等。

（2）编程语言的选择。在选择编程语言时，也需要做多方面的考虑：

通用性。随着微处理器技术的不断发展，其功能越来越专用，种类越来越多，但不同种类的微处理器都有自己专用的汇编语言。这就为系统开发者设置了一个巨大的障碍，使得系统编程更加困难，软件重用无法实现，而高级语言一般和具体机器的硬件结构联系较少，比较流行的高级语言对多数微处理器都有良好的支持，通用性较好。

可移植性。由于汇编语言和具体的微处理器密切相关，为某个微处理器设计的程序不能直接移植到另一个不同种类的微处理器上使用，因此，移植性差。高级语言对所有微处理器都是

通用的，因此，程序可以在不同的微处理器上运行，可移植性较好。这是实现软件重用的基础。

执行效率。一般来说，越是高级的语言，其编译器和开销就越大，应用程序也就越大、越慢。但单纯依靠低级语言，如汇编语言来进行应用程序的开发，带来的问题是编程复杂、开发周期长。因此，存在一个开发时间和运行性能之间的权衡。

可维护性。低级语言如汇编语言，可维护性不高。高级语言程序往往是模块化设计，各个模块之间的接口是固定的。因此，当系统出现问题时，可以很快地将问题定位到某个模块内，并尽快得到解决。另外，模块化设计也便于系统功能的扩充和升级。

基本性能。在嵌入式系统开发过程中使用的语言种类很多，比较广泛应用的高级语言有 Ada、C/C++、Modula-2 和 Java 等。Ada 语言定义严格，易读易懂，有较丰富的库程序支持，目前，在国防、航空、航天等相关领域应用比较广泛，未来仍将在这些领域占有重要地位。C 语言具有广泛的库程序支持，是嵌入式系统中应用最广泛的编程语言，在将来很长一段时间内仍将在嵌入式系统应用领域中占重要地位。 C++是一种面向对象的编程语言，在嵌入式系统设计中也得到了广泛的应用，如 GNU C++。Visual C++是一种集成开发环境，支持可视化编程，广泛应用于 GUI 程序开发。但 C 与 C++相比，C++的目标代码往往比较庞大和复杂，在嵌入式系统应用中应充分考虑这一因素。

（3）软件开发过程。嵌入式软件的开发过程不同于一般通用软件的开发过程，主要有如下步骤：

选择开发语言，建立交叉开发环境；

根据详细设计说明编写源代码，进行交叉编译、链接；

目标代码的重定位和下载；

在宿主机或目标机调试、验证软件功能；

进行代码的优化。

（4）软件开发文档。在嵌入式产品的开发设计过程中，开发阶段完成系统产品的实现，这一阶段同时需要完成一系列的文档，这些文档对完成产品设计、维护相当重要，这些文档分别为技术文件目录、技术任务书、技术方案报告、产品规格、技术条件、设计说明书、试验报告、总结报告等。

12.7.8 系统集成与测试

通常嵌入式系统测试主要包括软件测试、硬件测试、单元测试三个部分。

一般系统的硬件测试包括可靠性测试和电磁兼容性测试，关于电磁兼容性目前已经有了强制性国内和国际标准。

嵌入式系统软件测试方法和原理跟通用软件的测试基本一致，软件测试时，一般需要测试实例或测试序列，序列有两种来源：一种是需要用户进行设计，另一种是标准的测试序列。无论哪种测试实例，都要求实例能够高概率发现更多的错误，但在测试的内容上有些差别：

（1）嵌入式软件必须长时间稳定运行。

（2）嵌入式软件一般不会频繁地版本升级。

（3）嵌入式软件通常使用在关键性的应用中。

（4）嵌入式软件必须和嵌入式硬件一起对产品的故障和可靠性负责。

（5）现实世界的条件是异步和不可预测的，使得模拟测试非常困难。

由于这些差别，使得嵌入式系统软件测试主要集中在以下 4 个不同的方面：

（1）因为实时性和同时性很难同时满足，所以大多数测试集中于实时测试。

（2）大多数实时系统都有资源约束，因此需要更多的性能和可用性测试。

（3）可以使用专用实时跟踪工具对代码覆盖率进行测试。

（4）对可靠性的测试级别比通用软件要高得多。

另外，性能测试也是设计嵌入式系统中需要完成的最主要的测试活动之一，对嵌入式系统有决定性的影响。

由于嵌入式系统的专用性特点，系统的硬件平台和软件平台多种多样，每种都针对不同的应用而专门设计，因此，应用软件在各个平台之间很少具有通用性，并且嵌入式系统的更新换代速度相对较快。为了保护已有的投资、充分利用现有的软件资源和加快产品研制速度，软件的移植在嵌入式领域变得非常频繁。

第 13 章：开发管理

美国国防部曾于 20 世纪 70 年代中期专门针对软件项目失败的原因做了调查。调查结果显示 70%的失败软件项目都是因为管理不善引起的，而并不是事先以为的技术实力不够。到了 20 世纪 90 年代，据对美国软件工程实施现状的调查显示，大约只有 10%的项目，

尤其是商用软件，能够按预先计划的费用和进度交付。因此，业界认为影响软件研发项目全局的因素是管理水平，而技术只影响局部，这就有必要从项目管理的角度去管理软件的开发和运行。加强项目管理的好处是明显的，它可以控制财务成本、提高资源利用率；改进客户关系；缩短开发时间；降低成本；提高利润、生产率、产品质量和可靠性；完善公司内部协调等。

根据美国项目管理协会的项目管理知识体系可知，项目管理是指“在项目活动中运用专门的知识、技能、工具和方法，使项目能够实现或超过项目干系人的需要和期望。”一 般的项目管理可以分为范围管理、时间管理、费用管理、质量管理、人力资源管理、沟通 管理、风险管理、采购管理和整体管理 9 个知识领域。对于软件的开发管理来讲，软件范围管理、软件进度管理、软件成本管理、软件配置管理（属于整体管理）、软件质量管理、软件风险管理、开发人员管理（属于人力资源管理）7 个方面的管理尤为重要，软件开发的每个阶段、每个过程都要重视这几个方面的管理。

13.1 项目的范围、时间与成本

项目管理首先要考虑三个约束条件：项目范围、时间进度、成本预算。在签订软件开发合同时要明确：项目的任务是什么？发起人要通过项目获得什么样的产品或服务？这属于项目范围的范畴；项目需要多长时间？进度如何安排？这属于时间进度的范畴；项目需要花费多少？资金来源如何？这属于项目成本的范畴。

13.1.1 项目范围管理

所谓项目范围管理，包括保证项目顺利完成所需的全部工作过程。其目的是控制项目的全部活动都在需求范围内，以确保项目资源的高效利用。它主要包括项目启动、范围计划编制、范围定义、范围核实和范围变更控制 5 个部分的内容。项目启动指批准项目启动或者允许项目进入下一个阶段；范围计划编制是将生产项目产品所需进行的项目工作渐进明细和形成文件的过程；项目范围定义是把主要的项目可交付成果分解成更小、更易管理的单元，以达到如下目的：

提高对成本、时间及资源估算的准确性。

为绩效测量与控制定义一个基准计划。

便于进行明确的职责分配。

正确的范围定义是项目成功的关键。“当范围定义不明确时，不可避免的变更会使最终项目成本大大超出预算，因为这些不可避免的变更会破坏项目节奏，导致返工、增加项目历时、降低生产率和工作人员的士气”。范围核实是项目干系人（发起人、客户）正式接受项目范围的过程。范围核实需要审查可交付成果和工作结果，以确保它们都已经正确圆满地完成。如果项目被提前终止，范围核实过程应当对项目完成程度建立文档。范围核实与质量控制是不同的，范围核实是有关工作结果的“接收”，而质量控制是有关工作结果的正确性。
项目范围变更控制涉及的是：
对造成范围变更的因素施加影响，以确保这些变更得到一致认可；
确定范围变更是否已经发生；
当范围变更发生时对实际变更进行管理。

范围变更控制必须与其他控制管理过程（进行控制、成本控制和质量控制）结合在一起使用，才能取得良好的效果。

13.1.2 项目成本管理

所谓项目成本管理，是保证在批准预算内完成项目所需要的过程。成本对项目有关各方来说都是非常敏感的问题。因此成本管理在软件项目管理中是一项非常重要的工作。软件项目的成本不仅包括开发成本，也包括开发之前立项阶段及软件在运行中的费用。此外，操作者的培训费用和项目所使用的各种硬件设施费用也都是整个项目成本的一部分，这些成本都需要很好地计划和控制。

项目成本管理包括资源计划编制、成本估算、成本预算、成本控制 4 个主要部分内容。资源计划编制是确定为完成项目各活动需什么资源（人、设备、材料）和这些资源的数量。资源计划与成本估算是紧密相关的。成本估算就是计算出完成一个项目的各活动所需各资源成本的近似值。当一个项目按合同进行时，应区分成本估算和定价这两个不同意义的词。成本估算所涉及的是对可能数量结果的估算——执行组织为提供产品和服务的花费是多少；而定价是一个商业决策——执行组织为提供的产品或服务索取多少费用。成本估算是定价要考虑的因素之一。成本估算包括确认和考虑各种不同的成本估算替代方案。例如软件设计阶段多做些工作可减少编码阶段的成本。而成本估算过程必须考虑增加的设计工作所多花的成本是否被以后的节省所抵消。

成本预算是把估算的总成本分配到单个活动或工作包上去，建立基准计划来度量项目实

际绩效。成本控制的内容有：对造成成本基准计划变化的因素施加影响，以保证这种变化得到一致认可；确定成本基准计划是否已经发生变化；当变化发生和正在发生时，对这种变化执行管理。

成本控制包括以下方面：

监测成本执行情况，以寻找出并掌握计划的偏差及原因。

确保所有变更都准确地记录在成本基准计划中。

防止把不正确、不适宜或未批准的变更纳入成本基准成本。

将批准的变更通知项目干系人。

采取措施，把预计的成本控制在可接受的范围内。

成本控制包括寻找产生正负偏差的原因。成本控制必须和其他控制过程结合。例如，如果成本偏差采取不恰当的应对措施常会引起项目的质量和进度问题或引起项目在后期出现无法接受的风险。

13.1.3 项目时间管理

时间管理包括确保项目按时完成所需的各个过程。它包括活动定义、活动排序、活动历时估算、进度计划编制、进度控制 5 个部分内容。活动定义是对 WBS 中规定的可交付成果或半成品的产生所必须进行的具体活动进行定义，并形成文档。为使项目目标得以实现，在这个过程中对活动做出定义无疑是必要的。活动排序是确定各活动之间的依赖关系，并形成文档。活动必须被正确地加以排序，以便今后制定切实可行的进度计划。排序可由计算机辅助或用手工排序。

项目活动历时估算是根据项目范围和资源的相关信息为进度表设定历时输入的过程。历时估算的输入通常来自项目团队中熟悉该活动特性的个人和团体。估算通常采用渐进明细的方式，同时此过程需考虑输入数据的质量和可获得性。因此，可以假设此估算逐步精确，并且其质量水平是已知的。项目团队中最熟悉具体活动性质的个人或团队应当完成历时估算。制订进度计划要决定项目活动的开始和结束日期。若开始和结束日期是不现实的，项目就不可能按计划完成。进度计划、历时估算、成本估算等过程交织在一起，这些过程反复多次，最后才能确定项目进度计划。进度控制涉及的是：

对造成进度变更的因素施加影响，以确保这些变更得到一致认可；

确定进度变更是否已经发生；

当变更发生时对实际变更进行管理。

13.2 配置管理与文档管理

随着软件规模和复杂性的增大，许多大型开发项目往往都会延迟和超出预算，软件开发不得不直面越来越多的问题，表现为开发的环境日益复杂，代码共享日益困难，需跨越的平台增多；软件的重用性需要提高；软件的维护越来越困难。

为了解决这些问题，作为控制软件系统一系列变化的学科，软件配置管理（Software Configuration Management，SCM）应运而生。其主要作用是通过结构化的、有序化的、产品化的管理软件工程的方法来维护产品的历史，鉴别和定位产品独有的版本，并在产品的开发和发布阶段控制变化；通过有序管理和减少重复性工作，配置管理保证了生产的质量和效率；它涵盖了软件生命周期的所有领域并影响所有数据和过程。作为软件开发中一个重要过程，实现在有限的时间和资金内，满足不断增长的软件产品质量要求，软件配置管理已经逐渐受到各类软件企业的重视。

13.2.1 软件配置管理的概念

对于软件配置管理，IEEE 给出了一个定义：SCM 是指在软件系统中确定和定义构件（源代码、可执行程序、文档等），在整个生命周期中控制发布和变更，记录和报告构件的状态和变更请求，并定义完整的、正确的系统构件的过程。在 IEEE 标准 729-1983 中，软件配置管理包括以下几个方面功能：

配置标识：产品的结构、产品的构件及其类型，为其分配唯一的标识符，并以某种形式提供对它们的存取。

版本控制：通过建立产品基线，控制软件产品的发布和在整个软件生命周期中对软件产品的修改。例如，它将解决哪些修改会在该产品的最新版本中实现的问题。

状态统计：记录并报告构件和修改请求的状态，并收集关于产品构件的重要统计信息。例如，它将解决修改这个错误会影响多少个文件的问题。

审计和审查：确认产品的完整性并维护构件间的一致性，即确保产品是一个严格定义的构件集合。例如，它将解决目前发布的产品所用的文件的版本是否正确的问题。

生产：对产品的生产进行优化管理。它将解决最新发布的产品应由哪些版本的文件和工具来生成的问题。

过程管理：确保软件组织的规程、方针和软件周期得以正确贯彻执行。它将解决要交付给用户的产品是否经过测试和质量检查的问题。

小组协作：控制开发统一产品的多个开发人员之间的协作。例如，它将解决是否所有本地程序员所做的修改都已被加入新版本的产品中的问题。

而在另外一个标准 ISO9000.3 中，对软件配置管理系统做了如下要求：

唯一地标识每个软件项的版本；

标识共同构成一个完整产品的特定版本的每一软件项的版本；

控制由两个或多个独立工作的人员同时对一个给定软件项的更新；

按要求在一个或多个位置对复杂产品的更新进行协调；

标识并跟踪所有的措施和更改；这些措施和更改是在从开始直到放行期间，由于更改请求或问题引起的。

两个文件都强调了配置管理三个核心部分：版本管理、问题跟踪和建立管理，其中版本管理是基础。版本管理应完成以下主要任务：

建立项目；

重构任何修订版的某一项或某一文件；

利用加锁技术防止覆盖；

当增加一个修订版时要求输入变更描述；

提供比较任意两个修订版的使用工具；

采用增量存储方式；

提供对修订版历史和锁定状态的报告功能；

提供归并功能；

允许在任何时候重构任何版本；

权限的设置；

晋升模型的建立；

提供各种报告。

13.2.2 软件配置管理的解决方案

目前，软件配置管理的解决方案有许多厂商提供，如 Rational ClearCase，Merant PVCS，Microsoft VSS。大部分软件具备版本控制、建立管理、构造管理、问题追踪这些基本的功能

模块，有些软件还融合了需求管理、需求变更管理技术，并支持工作流程，以至Internet/Intranet 应用的异地通信和管理功能。可以看到软件配置管理的趋势是涉及面越来越广，将影响软件开发环境、软件过程模型、配置管理系统的使用者、软件产品的质量和用户的组织机构。

常用的软件配置管理工具，主要有如下产品：Rational ClearCase，Merant PVCS，Microsoft VSS，CVS。

1．Rational ClearCase

ClearCase 是 Rational 公司的主要配置管理工具，可以用于 Windows 和 UNIX 开发环境。ClearCase 主要应用于复杂的产品发放、分布式团队合作、并行的开发和维护任务，支持 Client/Server 网络结构。ClearCase 提供了全面的配置管理功能，包括版本控制、工作空间管理、建立管理和过程控制，而且无须软件开发者改变他们现有的环境、工具和工作方式。下面列举其主要功能：

（1）版本控制。ClearCase 的核心功能是版本控制，它是对软件开发进程中一个文件或一个目录发展过程进行追踪的手段，通过分支和归并功能支持并行开发。在软件开发环境中，ClearCase 可以对每一种对象类型（包括源代码、二进制文件、目录内容、可执行文件、文档、测试包、编译器、库文件等）实现版本控制。因而，ClearCase 提供的能力远远超出资源控制，并且可以帮助团队在开发软件时为他们所处理的每一种信息类型建立一个安全可靠的版本历史记录。

（2）工作空间管理。所谓空间管理，即保证开发人员拥有自己独立的工作环境，拥有自己的私人存储区，同时可以访问成员间的共享信息。ClearCase 给每一位开发者提供了一致、灵活的可重用工作空间域。它采用名为 View 的新技术，通过设定不同的视图配置规格，帮助程序员选择特定任务的每一个文件或目录的适当版本，并显示它们。View 可以让开发者在资源代码共享和私有代码独立的不断变更中达到平衡，从而使他们的工作更有效。

（3）建立管理。ClearCase 自动产生软件系统构造文档信息清单，而且可以完全、可靠地重建任何构造环境。ClearCase 也可以通过共享二进制文件和并发执行多个建立脚本的方式支持有效的软件构造。

使用 ClearCase，构造软件的处理过程可以和传统的方法兼容。对 ClearCase 控制的数据，既可以使用自制脚本也可使用本机提供的 make 程序，但 ClearCase 的建立工具 clearmake（支持 UNIX）和 omake（支持 NT）为构造提供了重要的特性：自动完成任务、保证重建的可靠性、存储时间和支持并行的分布式结构的建立。此外，ClearCase 还可以自

动追踪、建立产生永久性的资料清单。

（4）过程控制。ClearCase 有一个灵活、强大的功能，可以明确项目设计的流程。ClearCase 为团队通信、质量保证、变更管理提供了非常有效的过程控制和策略控制机制。ClearCase 可以有效地设置监控开发过程，这体现在：为对象分配属性；超级链接；历史记录；定义事件触发机制；访问控制查询功能等几个方面。自动的常规日志可以监控软件被谁修改、修改了什么内容及执行政策，如可以通过对全体人员的不同授权来阻止某些修改的发生，无论任何时刻某一事件发生应立刻通知团队成员，对开发的进程建立一个永久记录并不断维护它。

综上所述，ClearCase 支持全面的软件配置管理功能，给那些经常跨越复杂环境（如 UNIX、Windows 系统）进行复杂项目开发的团队带来巨大效益。此外，ClearCase 也支持广泛的开发环境，它所拥有的特殊构件已成为当今软件开发人员、工程人员和管理人员必备的工具。

2．Merant PVCS

Merant 的 PVCS 是世界领先的软件开发管理工具，在软件生命周期管理市场占有绝大多数市场份额，是公认的工业标准。全球的著名企业、软件机构、银行等诸多行业及政府机构大多数都应用了 PVCS。它能够实现配置管理中的各项要求，并且能和多种流行开发平台集成，为配置管理提供了很大的方便。PVCS 包含多种工具，几乎覆盖了软件开发管理中的所有问题。

PVCSVersionManager：能完整、详细地记录开发过程中出现的变更和修改，可快速得到系统中任何文件的各个版本，并使修订版本自动升级。

PVCSConfigurationBuilder：为软件系统提供了可靠的自动重建过程。它保证系统在任何时候对某一发布的产品准确地进行重建，避免发生错误，同时自动地对修改过的模块重新编译以节省时间。

PVCSTracker：在整个开发过程中确定和追踪软件的每一变更的要求。

PVCSNotify：将软件状态的变更通过 E-mail 通知组织机构中的其他成员。

PVCSReporter：为 GUI 界面环境提供一个客户报表工具，使用它能很容易地生成和存储多个项目的报表。

PVCSProductionGateway：提供了局域网间与大型机 MVS 系统双向同步互联。

PVCSDeveloper’sToolkit：为 PVCS 客户提供了应用程序开发接口（API），使项目信息通过编程访问。

PVCSRequisitePro：提供了一个独特的 MicrosoftWord 界面和需求数据库，从而可以使开发机构实时、直观地对来自于最终用户的项目需求及需求变更进行追踪和管理，可有效地避免

重复开发，保证开发项目按期、按质、按原有的资金预算交付用户。

上述的 8 个模块既可以单独安装和使用，也可以相互集成，建立工业化软件开发企业所需的完整的软件开发管理环境。PVCS 不仅很好地解决了代码重用、数据丢失等问题，它还从下述的几个主要方面，满足了软件开发机构迅速增长的市场需求，成为全球开发机构首选的软件配置管理工具。

3．Microsoft VSS，CVS

微软的 VSS（Visual SourceSafe）提供了基本的认证安全和版本控制机制，包括 CheckIn（入库）、CheckOut（出库）、Branch（分支）、Label（标定）等功能。版本控制是工作组软件开发中的重要方面，它能防止意外的文件丢失、允许反追踪到早期版本、并能对版本进行分支、合并和管理。在软件开发中比较两种版本的文件或找回早期版本的文件时，源代码的控制是非常有用的。

VSS 是一种源代码控制系统，它提供了完善的版本和配置管理功能，以及安全保护和跟踪检查功能。VSS 带有一个专业的文档、代码管理库，通过将有关项目文档（包括文本文件、图像文件、二进制文件、声音文件、视频文件）存入数据库进行项目研发管理工作。通过 VSS 与 APT 系统的配合，能够对文件进行控制，用户可以根据需要随时快速有效地共享文件。从文档的控制流程（增加、删除、修改、借阅等），到文档的修改信息记录，实现完善的文档管理。VSS 提供了历史版本的提取、提供源码历史版本对比。VSS 文件一旦被添加进 VSS，它的每次改动都会被记录下来，用户可以恢复文件的早期版本，项目组的其他成员也可以看到有关文档的最新版本，并对它们进行修改，VSS 也同样会将新的改动记录下来。用 VSS 来组织管理项目，使得项目组间的沟通与合作更简易而且直观。

VSS 可以同 Visual studio 开发环境及 Microsoft Office 应用程序集成在一起，提供了方便易用、面向项目的版本控制功能。VSS 可以处理由各种开发语言、创作工具或应用程序所创建的任何文件类型，用户可以同时在文件和项目级进行工作。VSS 面向项目的特性能更有效地管理工作组应用程序开发工作中的日常任务。VSS 的客户端既可以连接服务器运行，也可以在本机运行，非常适合于个人程序开发的版本管理。

CVS（并发版本系统，ConcurrentVersions System）是主流的开放源码的版本控制系统，Linux 和 UNIX 下系统自带的版本控制工具。CVS 对于从个人开发者到大型、分布式团队都是有用的。与微软 VSS 在实现功能上属于同一个级别，不过支持的操作平台不一样。

13.2.3 软件文档管理

所谓文档，是指某种数据媒体和其中所记录的数据。它具有永久性，并可以由人或机器阅读，通常仅用于描述人工可读的东西。在软件工程中，文档常常是用来对活动、需求、过程或结果进行描述、定义、规定、报告或认证的任何书面或图示的信息。

在软件生产过程中，总是产生和使用大量的信息。软件文档在产品的开发过程中起着重要的作用。

1．软件文档的作用

（1）管理依据。在软件开发过程中，管理者必须了解开发的进度、存在的问题和预期目标。每一阶段计划安排的定期报告提供了项目的可见性，把开发过程中发生的事件以某种可阅读的形式记录在文档中。定期报告还提醒各级管理者注意该部门对项目承担的责任及该部门效率的重要性。开发文档规定若干个检查点和进度表，使管理者可以评定项目的进度，如果开发文档有遗漏、不完善或内容陈旧，则管理者将失去跟踪和控制项目的重要依据。管理人员可把这些记载下来的材料作为检查软件开发进度和开发质量的依据，分析评估项目、检查调整项目/计划、调配专用资源，实现对软件开发的工程管理。

（2）任务之间联系的凭证。大多数软件开发项目通常被划分成若干个任务，并由不同的小组去完成。学科方面的专家建立项目，分析员阐述系统需求，设计员为程序员制定总体设计，程序员编制详细的程序代码，质量保证专家和审查员评价整个系统性能和功能的完整性，负责维护的程序员改进各种操作或增强某些功能。

这些人员需要的互相联系是通过文档资料的复制、分发和引用而实现的，因而，任务之间的联系是文档的一个重要功能。大多数系统开发方法为任务的联系规定了一些正式文档。分析员向设计员提供正式需求规格说明，设计员向程序员提供正式设计规格说明等

（3）质量保证。软件文档能提高开发效率。软件文档的编制使得开发人员对各个阶段的工作都进行周密思考、全盘权衡、减少返工。并且可在开发早期发现错误和不一致性，便于及时加以纠正。那些负责软件质量保证和评估系统性能的人员需要程序规格说明、测试和评估计划、测试该系统用的各种质量标准，以及关于期望系统完成什么功能和系统怎样实现这些功能的清晰说明；必须制订测试计划和测试规程，并报告测试结果；他们还必须说明和评估完全控制、计算、检验例行程序及其他控制技术。这些文档的提供可满足质量，保证人员和审查人员上述工作的需要。

（4）培训与参考。软件文档作为开发人员在一定阶段的工作成果和结束标志，它的另

一个功能是使系统管理员、操作员、用户、管理者和其他有关人员了解系统如何工作，以及为了达到他们各自的目的，如何使用系统。

（5）软件维护支持。记录开发过程中有关信息，便于协调以后的软件开发、使用和维护。维护人员需要软件系统的详细说明以帮助他们熟悉系统，找出并修正错误，改进系统以适应用户需求的变化或适应系统环境的变化。软件文档提供对软件的运行、维护的有关信息，便于管理人员、开发人员、操作人员、用户之间的协作、交流和了解。

（6）历史档案。良好的文档系统，作为全组织范围内共享所存储的文档信息，对于软件企业而言，也是一个很好的学习资源。通常文档记载系统的开发历史，可使有关系统结构的基本思想为以后的项目利用。系统开发人员通过审阅以前的系统以查明什么部分已试验过了，什么部分运行得很好，什么部分因某种原因难以运行而被排除。良好的系统文档有助于把程序移植和转移到各种新的系统环境中。

（7）销售可能。软件文档便于潜在用户了解软件的功能、性能等各项指标，为他们选购符合自己需要的软件提供依据。

从某种意义上来说，良好的文档管理是优秀项目的重要标志，文档是软件开发规范的体现和指南，也是记录和管理知识的重要形式。文档与知识管理文档是固化的知识，是显性知识的重要载体，按规范要求生成一整套文档的过程，就是按照软件开发规范完成一个软件开发的过程。从历史经验来看，写作文档在项目开发的早期可能会使项目的进度比起不写文档要稍慢，但随着项目的进展，部门间配合越来越多、开发方对用户需求越来越细，开发者越来越需要知道系统设计的开发思路和用户的进一步功能需求，才能使自己的开发朝着正确的方向推进。一个明显的例子就是系统整合，或者某些环节是建立在其他环节完成的基础之上时，就更显现出文档交流的准确性和高效性。文档的管理虽然是一个非常烦琐的工作，但是长远来看，它不仅使项目的开发对单个主要人员的依赖减少，从而减少人员流动给项目带来的风险，更重要的是在项目进行到后 10%的时候起到拉动项目的作用。所以，在使用工程化的原理和方法来指导软件的开发和维护时，应当充分注意软件文档的编制和管理。

2．文档的归类

按照文档产生和使用的范围，软件文档大致可分为 3 类：开发文档；管理文档；产品文档。

（1）开发文档。开发文档是描述软件开发过程，包括软件需求、软件设计、软件测试、保证软件质量的一类文档，开发文档也包括软件的详细技术描述（程序逻辑、程序间相互关系、数据格式和存储等）。开发文档起到如下 5 种作用：

它们是软件开发过程中包含的所有阶段之间的通信工具，它们记录生成软件需求、设计、编码和测试的详细规定和说明；

它们描述开发小组的职责。通过规定软件、主题事项、文档编制、质量保证人员，

以及包含在开发过程中任何其他事项的角色来定义做什么、如何做和何时做；

它们用作检验点而允许管理者评定开发进度。如果开发文档丢失、不完整或过时，管理者将失去跟踪和控制软件项目的一个重要工具；

它们形成了维护人员所要求的基本的软件支持文档。而这些支持文档可作为产品文档的一部分；

它们记录软件开发的历史。基本的开发文档包括：可行性研究和项目任务书；需求规格说明；概要设计说明；详细设计说明，包括程序和数据规格说明；项目开发计划；软件集成和测试计划；质量保证计划、标准、进度；安全和测试信息。

（2）产品文档。产品文档规定关于软件产品的使用、维护、增强、转换和传输的信息。产品的文档起到如下 3 种作用：

为使用和运行软件产品的任何人规定培训和参考信息；

使得那些未参加开发本软件的程序员维护它；

促进软件产品的市场流通或提高可接受性。

产品文档主要应用于下列类型的读者：

用户——他们利用软件输入数据、检索信息和解决问题；

运行者——他们在计算机系统上运行软件；

维护人员——他们维护、增强或变更软件。

产品文档包括如下内容：用于管理者的指南和资料，他们监督软件的使用；宣传资料通告软件产品的可用性并详细说明它的功能、运行环境等；一般信息对任何对其感兴趣的人描述软件产品。基本的产品文档实物包括：培训手册；参考手册和用户指南；软件支持手册；产品手册和信息广告；维护修改建议等。

（3）管理文档。这种文档建立在项目管理信息的基础上，从管理的角度规定涉及软件生存的信息。它包括：项目开发计划、测试计划；开发过程的每个阶段的进度和进度变更的记录；软件变更情况的记录；相对于开发的判定记录；开发人员职责定义；测试报告、开发进度月报；项目开发总结等。

另外，软件文档从用途上还可以分为内部文档和外部文档。其中，内部文档包括项目开发计划、需求分析、架构设计说明、详细设计说明、构件索引、构件成分说明、构件接口及

调用说明、构件索引、构件接口及调用说明、类索引、类属性及方法说明、测试报告、测试统计报告、质量监督报告、源代码、文档分类版本索引和软件安装打包文件等。

外部文档主要包括软件安装手册、软件操作手册、在线帮助、系统性能指标报告和系统操作索引等。

3．文档编制计划

软件开发的管理部门应该根据本单位承担的应用软件的专业领域和本单位的管理能力，制定一个对文档编制要求的实施规定。对于一个具体的应用软件项目，项目负责人应根据上述实施规定，确定一个文档编制计划。

文档计划可以是整个项目计划的一部分或是一个独立的文档。应该编写文档计划并把它分发给全体开发组成员，作为文档重要性的具体依据和管理部门文档工作责任的备忘录。编制计划的工作应及早开始，对计划的评审应贯穿项目的全过程。如同任何别的计划一样，文档计划指出未来的各项活动，当需要修改时必须加以修改。导致对计划作适当修改的常规评审应作为该项目工作的一部分，所有与该计划有关的人员都应得到文档计划。

文档计划一般包括以下几方面内容：

列出应编制文档的目录；

提示编制文档应参考的标准；

指定文档管理员；

提供编制文档所需要的条件，落实文档编写人员、所需经费及编制工具等；

明确保证文档质量的方法，为了确保文档内容的正确性、合理性，应采取一定的措施，如评审、鉴定等；

绘制进度表，以图表形式列出在软件生存期各阶段应产生的文档、编制人员、编制日期、完成日期、评审日期等。

还必须明确：要编制哪几种文档，详细程度如何；各文档的编制负责人和进度要求；审查/批准负责人和时间进度安排；在开发时期内各文档的维护、修改和管理的负责人，以及批准手续。文档计划还应确定该计划和文档的分发，有关的开发人员必须严格执行这个文档编制计划。文档计划还应该规定每个文档要达到的质量等级，以及为了达到期望的结果必须考虑哪些外部因素。

4．对文档质量的要求如果不重视文档编写工作，或是对文档编写工作的安排不当，就不可能得到高质量的

文档。质量差的文档一般会使读者难以理解，给使用者造成许多不便；会削弱对软件的管理

（难以确认和评价开发工作的进展情况），提高软件成本（一些工作可能被迫返工）；造成误操作。一般而言，好的软件文档要求具备如下特征。

（1）针对性。文档编制前应分清读者对象。对不同的类型、不同层次的读者，决定如何满足适应他们的需要。

管理文档主要面向管理人员。

用户文档主要面向用户。

这两类文档不应像开发文档（面向开发人员）那样过多地使用软件的专业术语。

（2）精确性。文档的行文应当十分确切，不能出现多义性的描述。同一课题几个文档的内容应当是协调一致、没有矛盾的。

（3）清晰性。文档编写应力求简明，如有可能，配以适当的图表，以增强其清晰性。

（4）完整性。任何一个文档都应当是完整的、独立的，它应自成体系。例如，前言部分应做一般性介绍，正文给出中心内容，必要时还有附录，列出参考资料等。同一课题的几个文档之间可能有部分内容相同，这种重复是必要的。不要在文档中出现转引其他文档内容的情况。如，一些段落没有具体描述，用“见××文档××节”的方式。

（5）灵活性。各个不同软件项目，其规模和复杂程度有着许多实际差别，不能相同看待。应根据具体的软件开发项目，决定编制的文档种类。

13.3 软件需求管理

在软件开发的整个过程中，随着客观条件的变化和客户对软件或业务理解的加深，会产生很多新的软件需求，项目经理需要经常面对需求变更。需求管理的目的就是控制和维持事先约定，保证项目开发过程的一致性，使用户能够得到他们最终想要得到的软件产品。下面的内容主要涉及需求管理的两个方面：需求变更、需求跟踪。

13.3.1 需求变更

需求变更是指在软件开发过程中，用户确定软件需求之后，由于各种客观和主观条件的变化，用户增加了新的需求或改变了原有需求。

项目经理需要在整个项目生命周期中管理需求变更，将项目变更的影响降到最低。进行需求变更控制的主要依据是项目计划、变更请求和反映项目执行状况的绩效报告。为保证项目变更的规范性和项目的有效实施，通常软件开发机构会采取如下措施。

（1）项目启动阶段的变更预防。对于任何项目，变更都无可避免，也无从逃避，只能积极应对。这个应对应该是从项目启动的需求分析阶段就开始了。对一个需求分析做得很好的项目来说，基准文件定义的范围越详细、清晰，用户跟项目经理的分歧就越少。如果需求做得好，文档清晰且有客户签字，那么后期客户提出的变更超出了合同范围，就需要另外处理。

（2）项目实施阶段的需求变更。成功项目和失败项目的区别就在于项目的整个过程是否是可控的。项目经理应该树立一个理念 “需求变更是必然的、可控的、有益的”。项目实施阶段的变更控制需要做的是分析变更请求，评估变更可能带来的风险和修改基准文件。控制需求变更需要注意以下几点：

需求一定要与投入有联系，如果需求变更的成本由开发方来承担，则项目需求的变更就成为必然了。所以，在项目的开始，无论是开发方还是出资方都要明确这一条：需求变，软件开发的投入也要变。

需求的变更要经过出资者的认可，使需求的变更有成本的概念。这样项目实施涉及各方就能够慎重地对待需求的变更。

小的需求变更也要经过正规的需求管理流程。在实践中，人们往往不愿意为小的需求变更去执行正规的需求管理过程，认为降低了开发效率，浪费了时间。但正是由于这种观念才使需求逐渐变为不可控，最终导致项目的失败。

还要注意沟通的技巧。实际情况是用户、开发者都认识到了上面的几点问题，但是由于需求的变更可能来自客户方，也可能来自开发方，因此，作为需求管理者，项目经理需要采用各种沟通技巧来使项目的各方受益。

13.3.2 需求跟踪

需求跟踪是指在软件需求管理的过程中定义需求变更流程，分析需求变更影响，控制变化的版本，维护需求变更记录,跟踪每项需求状态。

（1）确定需求变更控制过程。制定一个选择、分析和决策需求变更的标准过程，所有的需求变更都需遵循此过程。

（2）进行需求变更影响分析。评估每项需求变更，以确定它对项目计划安排和其他需求的影响，明确与变更相关的任务，并评估完成这些任务需要的工作量。通过这些分析将有助于需求变更控制部门做出更好的决策。

（3）建立需求基准版本和需求控制版本文档。确定需求基准，这是项目各方对需求达成一致认识时刻的一个快照，之后的需求变更遵循变更控制过程即可。每个版本的需求规格说明都必须是独立说明，以避免将底稿和基准或新旧版本相混淆。

（4）维护需求变更的历史记录。将需求变更情况写成文档，记录变更日期、原因、负责人、版本号等内容，及时通知到项目开发所涉及的人员。为了尽量减少困惑、冲突、误传，应指定专人来负责更新需求。

（5）跟踪每项需求的状态。可以把每一项需求的状态属性（如已推荐的，已通过的，已实施的，或已验证的）保存在数据库中，这样可以在任何时候得到每个状态类的需求数量。

13.4 软件开发的质量与风险

随着软件开发的规模越来越大，软件的质量问题越来越引起人们的关注。关于软件质量，IEEE 729—1983 标准有以下定义：

软件产品满足给定需求的特性及特征的总体的能力。

软件拥有所期望的各种属性组合的程度。

顾客或用户认为软件满足他们综合期望的程度。

软件组合特性在使用中，满足用户预期需求的程度。

从上述这个定义可以看到质量不是绝对的，它总是与给定的需求有关。因此，对软件质量的评价总是在将产品的实际情况与从给定的需求中推导出来的软件质量的特征和质量标准进行比较后得出来的。尽管如此，这里给出的软件质量还是一个模糊的概念且难以衡量。所以，软件质量管理的目的是建立对项目的软件产品质量的定量理解和实现特定的质量目标。

13.4.1 软件质量管理

项目质量管理包括保证项目能满足原先规定的各项要求所需要的过程，即“总体管理功能中决定质量方针、目标与责任的所有活动，并通过诸如质量规划、质量保证、质量控制、质量改进等手段在质量体系内加以实施”。软件质量管理着重于确定软件产品的质量目标、制订达到这些目标的计划，并监控及调整软件计划、软件工作产品、活动及质量目标以满足顾客及最终用户对高质量产品的需要及期望。

软件质量管理包括三个部分：质量计划——判断哪些质量标准与本项目相关，并决定应如何达到这些质量标准；质量保证——定期评估项目总体绩效，建立项目能达到相关质量标

准的信心；质量控制——监测项目的总体结果，判断它们是否符合相关质量标准，并找出如何消除不合格绩效的方法。

1．软件质量计划

在正式进行软件开发前，需要制订一个软件质量计划，用于说明项目管理团队将如何实施其质量方针。用 ISO9000 的话来说，它应该说明项目质量体系，即：“用以实施质量管理的组织结构、责任、程序、过程和资源”。目前国际上有许多质量标准，较常用的是 ANSI/IEEE STOL730—1984，983—1986 标准。质量计划可以识别哪些质量标准适用于本项目，并确定如何满足这些标准的要求。在软件质量计划阶段应该完成如下活动：

对项目的软件质量活动做出计划。

对软件产品质量的可测量的目标及其优先级进行定义。

确定软件产品质量目标的实现过程是可量化和可管理的。

为管理软件产品的质量提供适当的资源和资金。

对实施和支持软件质量管理的人员进行实施和支持过程中所要求的培训。

对软件开发项目组和其他与软件项目有关的人员进行软件质量管理方面的培训。

按照已文档化的规程制订和维护项目的软件质量计划。

项目的软件质量管理活动要以项目的软件质量计划为基础。

在整个软件生命周期，要确定、监控和更新软件产品的质量目标。

2．软件质量保证质量保证指为项目符合相关质量标准要求树立信心而在质量系统内部实施的各项有计划的系统活动。质量保证应贯穿于项目的始终，在事件驱动的基础上，对软件产品的质量进行测量、分析，并将分析结果与既定的质量标准相比较，以提供质量改进的依据。如果属于软件外包，还需要对软件产品的定量质量目标进行合理的分工，分派给项目交付软件产品的承包商。

3．软件质量控制质量控制指监视项目的具体结果，确定其是否符合相关的质量标准，并判断如何杜绝造成不合格结果的根源。软件质量的控制不单单是一个软件测试问题，评审、调试和测试是保证软件质量的重要手段。质量控制指监视项目的具体结果，确定其是否符合相关的质量标准，并判断如何杜绝造成不合格结果的根源。质量控制应贯穿于项目的始终。项目结果既包括产品结果（例如可交付成果）、也包括项目管理结果（例如成本与进度绩效）。质量控制通常由机构中的质量控制部或名称相似的部门实施，软件质量控制包括如下活动：

对软件产品进行测试，并将测试结果用于软件质量管理活动的状态。

高级管理者定期参与评审软件质量管理的活动。

软件项目负责人定期参与评审软件质量管理的活动。

软件质量保证评审小组负责评审软件的质量管理活动和工作产品，并填写相关报告。

（1）软件评审。软件评审并不是在软件开发完毕后进行评审，而是在软件开发的各个阶段都要进行评审。因为在软件开发的各个阶段都可能产生错误，如果这些错误不及时发现并纠正，会不断地扩大，最后可能导致开发的失败。

首先，要明确评审目标包括如下部分：

发现任何形式表现的软件功能、逻辑或实现方面的错误；

通过评审验证软件的需求；

保证软件按预先定义的标准表示；

已获得的软件是以统一的形式开发的；

使项目更容易管理。

其次，评审过程应包括：

召开评审会议。

会议结束时必须做出以下决策之一：① 接受该产品，不需作修改；② 由于错误严重，拒绝接受；③ 暂时接受该产品。

评审报告与记录；所提出的问题都要进行记录，在评审会结束前产生一个评审问题表，另外必须完成评审简要报告。

还应该遵循基本的评审准则，如：

对每个正式技术评审分配资源和时间进度表；

评审产品，而不是评审设计者，不能使设计者有任何压力；

会议不能脱离主题，应建立议事日程并维持它；

评审会不是为了解决问题，而是为了发现问题，限制争论与反驳；

对每个被评审的产品建立评审清单，以帮助评审人员思考；

（2）测试。软件测试是软件开发的一个重要环节，同时也是软件质量保证的一个重要环节。所谓测试就是用已知的输入在已知环境中动态地执行系统（或系统的构件）。测试一般包括单元测试、模块测试、集成测试和系统测试。如果测试结果与预期结果不一致，则很可能是发现了系统中的错误，测试过程中将产生下述基本文档。

测试计划：确定测试范围、方法和需要的资源等。

测试过程：详细描述与每个测试方案有关的测试步骤和数据（包括测试数据及预期的结果）。

测试结果：把每次测试运行的结果归入文档，如果运行出错，则应产生问题报告，并且必须

经过调试解决所发现的问题。

13.4.2 项目风险管理

无论是系统集成还是软件开发，IT 公司经常面临着各种项目的实施和管理，面临着如何确定项目的投资价值、评估利益大小、分析不确定因素、决定投资回收时间等众多问题。并且，一个 IT 项目，无论其规模大小，必然会为被实施方（用户）在管理、业务经营等多方面带来变革，这就使 IT 项目必然具有高风险性的特点。尤其是近年来，IT 项目的广泛实施，一方面为众多的企业带来了管理、经营方面的革新，而另一方面，夭折、中断、失败的项目也不在少数。因此，如何在项目实施中有效地管理风险、控制风险，已经成为了项目实施成功的必要条件。

实际上，项目风险的管理不仅贯穿于整个项目过程，而且在项目事件发生之前风险的分析就已经开始。可以根据风险控制与项目事件发生的时间将风险管理划分为三个部分：事前控制——风险管理规划，事中控制——风险管理方法，事后控制——风险管理报告。

1．项目风险管理的概念

根据 PMBOK2000 版的定义，风险管理指对项目风险进行识别、分析、并采取应对措施的系统过程。它包括尽量扩大有利于项目目标事项发生的概率与后果，而尽量减小不利于项目目标事项发生的概率与后果。

项目风险按是否有可确定性划分为：已知风险、可预知风险、不可预知风险。按风险管理的内容又可以划分为如下几种类型。

（1）内部技术风险：技术变化和创新是项目风险的重要来源之一。一般说来，项目中采用新技术或技术创新无疑是提高项目绩效的重要手段，但这样也会带来一些问题，许多新的技术未经证实或并未被充分掌握，则会影响项目的成功。还有，当人们出于竞争的需要，就会提高项目产品性能、质量方面的要求，而不切实际的要求也是项目风险的来源。

（2）内部非技术风险：公司的经营战略发生了变化相关的战略风险、涉及公司管理/ 项目管理人员管理水平等的管理风险，以及与范围变更有关的风险；没有按照要求的技术性能和质量水平完成任务的质量风险；没有在预算的时间范围内完成任务的进度风险；没有在预算的成本范围内完成任务的成本风险。

（3）外部法律风险：包括与项目相关的规章或标准的变化，如许可权、专利、合同失效、诉讼等。

（4）外部非法律风险：主要是指项目的政治、社会影响、经济环境的变化，组织中雇佣关系的变化，如公司并购、政府干预、货币变动、通货膨胀、税收、自然灾害等。这类风险对项目的影响和项目性质的关系较大。

2．风险管理的过程风险管理包括对项目风险识别、分析和应对的过程，从而将正面事件影响扩大到最大化和将负面事件影响减少到最小化。项目风险管理的主要过程包括：

风险管理规划，决定如何指导和规划项目的风险管理活动。

项目风险识别，找到哪些风险可能影响项目，并记录其特征。

定性风险分析，完成风险和环境的定性分析，并按其对项目目标的影响进行排序。定性风险分析是决定具体风险的重要性并指导做出相关反应的一种方法。与风险相关的动作的时间相关性可能使风险的重要性加大。

定量风险分析，度量风险的可能性和后果，估量其对项目目标的潜在影响。

风险应对计划，创建过程和技术来为项目目标增进机会和减小威胁。

风险监督与控制，在项目生命周期中监视现存的风险、识别新的风险、执行缓解风险计划及评估其效果。

上述过程不仅彼此有交互作用，而且也同其他知识领域的过程有交互作用。一般来说，每个过程在项目中至少出现一次。

（1）风险识别。风险的识别就是识别整个项目过程中可能存在的风险事件。在项目开始、每个项目阶段中间、主要范围变更批准之前都要进行风险识别，实际上它在整个项目生命周期内都是一个连续的过程。要识别风险，首先应该了解在软件开发的各个阶段都有可能发生哪些风险（风险事件或风险来源）。表 13-1 列出软件开发各阶段可能发生的风险。

表 13-1　软件开发各阶段的风险事件

阶　段	风险事件
初始阶段	项目目标不清 项目范围不明确 用户参与少或和用户沟通少 对业务了解不够 对需求了解不够 没有进行可行性研究
设计阶段	项目队伍缺乏经验，如缺乏有经验的系统分析员 没有变更控制计划，以至于变更没有依据，该变更的不变，不该变更的改变，这样得来的设计势必会失败或者偏离用户需求 仓促计划，可能带来进度方面的风险 漏项，由于设计人员的疏忽某个功能没有考虑进去
实施阶段	开发环境没有具备好 设计错误带来的实施困难 程序员开发能力差，或程序员对开发工具不熟 项目范围改变（突然要增加或修改一些功能，需要重新考虑设计） 项目进度改变（要求提前完成任务等） 人员离开，在一个项目内软件开发工作有一定的连续性，需要移交和交接，有时人员离开对项目的影响会很大 开发团队内部沟通不够，导致程序员对系统设计的理解上有偏差 没有有效的备份方案 没有切实可行的测试计划 测试人员经验不足
收尾阶段	质量差 客户不满意 设备没有按时到货 资金不能回收

其中，初始阶段主要进行大部分需求分析、小部分设计（大部分业务建模和需求、少部分分析设计）；设计阶段主要进行大部分设计、小部分编码（大部分分析设计，部分实施及测试，开始考虑部署）；实施阶段进行大部分编码和测试，也涉及小部分设计（大部分实施及测试，部分部署）；收尾阶段完成安装及维护（大部分部署）。

除了考虑项目过程之外，软件企业在人力资源管理中也存在风险，如招聘失败、新政策引起员工不满、技术骨干突然离职等，这些事件会影响公司的正常运转，甚至会对公司造成致命的打击。特别是高新技术企业，由于对人的依赖更大，所以更需要识别人力资源方面的风险。

以上只是列举了常见的风险事件，对不同项目可能发生的风险事件不同，应该对具体项目识别出真正有可能发生在该项目的风险事件。一般是根据项目的性质，从潜在的事件及其产生的后果和潜在的后果及其产生的原因来检查风险。收集、整理项目可能的风险并充分征求各方意见就形成项目的风险列表，并对这些风险事件进行描述，如：可能性、可能后果范围、预计发生时间、发生频率等。风险识别的有效方法有很多，如：建立风险项目检查表、

因果分析图、采访各种项目干系人等。

（2）风险分析。确定了项目的风险列表之后，接下来就可以进行风险分析了。风险分析的目的是确定每个风险对项目的影响大小，一般是对已经识别出来的项目风险进行量化估计，这里要注意三个概念。

风险得失值：它是指一旦风险发生可能对项目造成的影响大小，说明可能造成的损失。如果损失的大小不容易直接估计，可以将损失分解为更小部分再评估它们。风险得失值可用相对数值表示，建议将损失大小折算成对计划影响的时间表示。

风险概率：它是风险发生可能性的百分比表示，是一种主观判断。

风险值：风险值又风险曝光度或风险暴露，它是评估风险的重要参数，“风险值” “风险概率” “风险影响”。如：某一风险概率是 25%，一旦发生会导致项目计划延长 4 周，因而，风险值 25% 4 周 1 周。

风险分析就是对以上识别出来的风险事件做风险影响分析。通过对风险及风险的相互作用的估算来评价项目可能结果的范围，从成本、进度及性能三个方面对风险进行评价，确定哪些风险事件或来源可以避免，哪些可以忽略不考虑（包括可以承受），哪些要采取应对措施。

（3）风险应对方法。完成了风险分析后，就已经确定了项目中存在的风险及它们发生的可能性和对项目的风险冲击，并可排出风险的优先级。此后就可以根据风险性质和项目对风险的承受能力制订相应的防范计划，即风险应对。制订风险应对策略主要考虑以下 4 个方面的因素：可规避性、可转移性、可缓解性、可接受性。风险的应对策略在某种程度上决定了采用什么样的项目开发方案。对于应“规避”或“转嫁”的风险在项目策略与计划时必须加以考虑。

项目中的风险永远不能全部消除，PMBOK2012 版提到 4 种应对方法：

规避。规避风险指改变项目计划，以排除风险或条件，或者保护项目目标，使其不受影响。虽然项目永远不可能排除所有的风险事件，但某些具体风险则是可以规避的。出现于项目早期的某些风险事件可以通过澄清要求、取得信息、改善沟通，或获取技术专长而获得解决。通过分析找出发生风险事件的原因，消除这些原因来避免一些特定的风险事件发生。例如，如何避免客户不满意？客户不满意有两种情况，一种情况是没有判断客户满意度的依据，即没有双方互相认可的客户验收标准，还有一种是开发方没有达到验收标准，即没有满足用户需求。不管是哪一种，开发方都有不可推卸的责任，只要做好以下环节就完全可以避免：业务建模阶段要让客户参与；需求阶段要多和客户沟通，了解客户真正的需求；目标系统的模

型或 DEMO 系统要向客户演示，并得到反馈意见，如果反馈的意见和 DEMO 系统出入比较大时，一定要将修改后的 DEMO 系统再次向客户演示，直到双方都达成共识为止；要有双方认可的验收方案和验收标准；做好变更控制和配置管理等。

转移。转移风险指设法将风险的后果连同应对的责任转移到第三方身上。转移风险实际只是把风险管理责任推给另一方，而并非将其排除。该方法基本上需要向承担风险者支付风险费用。它包括利用保险、履约保证书、担保书和保证书。可以利用合同将具体风险的责任转嫁给另一方。如果项目的设计是稳定的，可以用固定总价合同把风险转嫁给卖方。虽然成本报销合同把较多的风险留给了顾客或赞助人，但如果项目中途发生变化时，它有助于降低成本。

减轻。通过降低风险事件发生的概率或得失量来减轻对项目的影响。提前采取行动以减小风险发生的概率或者减小其对项目所造成的影响，这样比在风险发生后亡羊补牢地进行补救要有效得多。减轻风险的成本应估算得当，要与风险发生的概率及其后果相称。项目预算中考虑应急储备金是另一种降低风险影响的方法。例如，经过风险识别发现，项目组的程序员对所需开发技术不熟。可以采用熟悉的技术来减轻项目在成本或进度方面的影响。也可以事先进行培训来减轻对项目的影响。

接受。接受风险造成的后果。采取此项技术表明项目团队已经决定不为处置某项风险而改变项目计划，或者表明他们无法找到任何其他应对良策。主动地接受风险包括制定一套万一发生风险时所准备实施的应变计划。例如，为了避免自然灾害造成的后果，在一个大的软件项目中考虑了异地备份中心。

确定风险的应对策略后，就可编制风险应对计划，它主要包括：已识别的风险及其描述、风险发生的概率、风险应对的责任人、风险对应策略及行动计划、应急计划等。

（4）风险应对计划。针对需要采取应对措施的风险事件，开发应对计划，一旦发生风险事件，就实施应对计划。应对计划常应用于项目进行期间发生的已识别风险，事先制订应变计划可大大降低风险发生时采取行动的成本。风险触发因素，例如缺失的中间里程碑，应确定其定义，并进行跟踪。如果风险的影响甚大，或者所选用的对策不见得有效时，就应制订一套后备权变计划。该项计划可包括留出一笔应急款项、制定其他备用方案、或者改变项目范围。最常见的接受风险的应对措施是预留应急储备，或者简称储备，包括为已知风险留出时间、资金、或者资源。为所接受的风险所预留的储备取决于按可接受风险水平计算所得影响的大小。

例如，在一个软件开发项目中，某开发人员有可能离职，离职后会对项目造成一定的影响，则应该对这个风险事件开发应对计划，过程参照如下：

进行调研，确定流动原因。

在项目开始前，把缓解这些流动原因的工作列入风险管理计划。

项目开始时，作好一旦人员离开时便可执行的计划，以确保人员离开后项目仍能继续进行。

制定文档标准，并建立一种机制，保证文档及时产生。

对所有工作进行细微详审，使更多人能够按计划进度完成自己的工作。

对每个关键性技术人员培养后备人员。

当然该应对计划需要花费一定的成本，在考虑风险成本之后，决定是否采用上述策略。

（5）风险监控。制定了风险应对计划后，风险并非不存在，在项目推进过程中还可能会增大或者衰退。因此，在项目执行过程中，需要时刻监督风险的发展与变化情况，并确定随着某些风险的消失而带来的新的风险。

风险监控包括两个层面的工作：其一是跟踪已识别风险的发展变化情况，包括在整个项目周期内，风险产生的条件和导致的后果变化，衡量风险减缓计划需求。其二是根据风险的变化情况及时调整风险应对计划，并对已发生的风险及其产生的遗留风险和新增风险及时识别、分析，并采取适当的应对措施。对于已发生过和已解决的风险也应及时从风险监控列表调整出去。

最有效的风险监控工具之一就是“前 10 个风险列表”，它是一种简便易行的风险监控活动，是按“风险值”大小将项目的前 10 个风险作为控制对象，密切监控项目的前 10 个风险。每次风险检查后，形成新的“前 10 个风险列表”。

13.5 人力资源管理

软件项目人力资源管理包括为最有效地使用参与项目人员所需的各项过程，一般包括组织规划、人员招募和团队建设三个主要过程。

1．组织规划组织规划用于确定、记录并分派项目角色、职责和请示汇报关系。角色、职责和请示汇报关系可以分派给个人或者集体。这些个人与集体可以是项目实施组织的一部分，也可以来自组织外部，通过人员招聘、借用等方式获得。实施组织往往与某个具体职能部门相关，例如，工程部门、销售部门或者财务部门，通过与职能经理协商、谈判等方式获得。

软件项目组织一般由担当各种角色的人员所组成。每位成员扮演一个或多个角色，常见的一些项目角色包括：策划师、数据库管理员、设计师、操作/支持工程师、程序员、项目

经理、项目赞助者、质量保证工程师、需求分析师、用户、测试人员等。组织规划取决于可供选择的人员、项目的需求及组织的需求，组织的具体形式可以有三种方案：垂直方案、水平方案和混合方案。以垂直方案组织的团队由多面手组成，每个成员都充当多重角色。以水平方案组织的团队由专家组成，每个成员充当一到两个角色。以混合方案组织的团队既包括多面手，又包括专家。

如果可供选择的人员中大多数人员是多面手，则往往需要采用垂直方案；同样，如果大多数人员是专家，则采用水平方案。如果正引入一些新人，即使这些人员都是合同工，则仍然需要优先考虑项目和组织。本节描述了形成团队组织的垂直、水平和混合方案，并指出了它们各自的优缺点。

（1）垂直团队组织。垂直团队由多面手组成。如，功能模块分配给了个人或小组，然后由他们从头至尾地实现该功能模块。其优点在于，以单个功能模块为基础实现平滑的端到端开发；开发人员能够掌握更广泛的技能。而缺点也很明显：

多面手通常是一些要价很高并且很难找到的顾问。

多面手通常不具备快速解决具体问题所需的特定技术专长。

主题专家可能不得不和若干开发人员小组一起工作，从而增加了他们的负担。

所有多面手水平各不相同。

（2）水平团队组织。水平团队由专家组成。此类团队同时处理多个功能模块，每个成员都从事功能模块中有关其自身的方面。其优点在于能高质量地完成项目各个方面（需求、设计等）的工作；一些外部小组，如用户或操作人员，只需要与了解他们确切要求的一小部分专家进行交互。其缺点在于：专家们通常无法意识到其他专业的重要性，导致项目的各方面之间缺乏联系；由于专家们的优先权、看法和需求互不相同，所以项目管理比较困难。

（3）混合团队组织。混合团队由专家和多面手共同组成。多面手继续操作一个功能模块的整个开发过程，支持并处理多个功能模块，使各部分的专家们一起工作。它可能拥有前两种方案的优点：外部小组只需要与一小部分专家进行交互；专家们可集中精力从事他们所擅长的工作；各个功能模块的实现都保持一致。但是它可能拥有前两种方案的缺点：尽管这应该由多面手来调节，专家们仍然不能认识到其他专家的工作并且无法很好地协作；多面手较难找到，故而，项目管理仍然较难。

因而，要综合考虑、确定团队组织方案。在方案确定后，合理配备人员是成功完成软件开发项目的切实保证。所谓合理配备人员应包括按不同阶段适时运用人员，恰当掌握用人标准。一般来说，软件项目不同阶段、不同层次技术人员的参与情况是不一样的。如人员配置

不当，很容易造成人力资源的浪费，并延误工期。特别是采用恒定人员配备方案时，在项目的开始和最后都会出现人力过剩，而在中期又会出现人力不足的情况。

在多数项目中，组织规划大都是作为项目早期阶段的一部分进行的。但在项目的整个过程中都应对其结果定期检查，以保证其继续适用性。如果当初的组织规划已不再适用，就应该及时对其进行修改。

2．人员招募人员招募指获取分派到项目上、并在那里工作所需的人力资源（个人或集体）。在多数环境中，很可能无法到“最佳”资源，因此项目管理团队必须注意保证所物色到的人力资源符合项目要求。

通过前一阶段完成的成果——人员配备管理计划，来进行实际的人员招募。首先必须了解组织可用/备用人员库的情况。按照组织规划确定的人力资源需求情况，充分考虑可调配人员的特点。要考虑的问题主要有：

以往经验——这些个人或集体以前是否从事过类似或者相关的工作？工作表现如何？

个人兴趣——这些个人或集体对本项目的工作感兴趣吗？

能否得到——最理想的个人或集体人选能在规定期限内招募到手吗？

胜任与熟练程度——需要何种能力及何种水平？

而后，通过谈判、事先分派和采购等方式获取项目人员，以保证项目在规定期限内获得足以胜任的工作人员。其中谈判对象可能是实施组织的职能经理，也可能分到其他项目团队，以争取稀缺或特殊人才得到合理分派。在某些情况下，人员可能事先被分派到项目上。这种情况往往发生在：项目是方案竞争的结果，而且事先已许诺具体人员指派是获胜方案的组成部分，或者项目为内部服务项目，人员分派已在项目章程中明确规定。在实施组织缺乏完成项目所需的内部人才时，就需要动用采购手段。

项目经理是团队组织的核心，其综合素质直接影响项目的成败。一般要求项目经理具备如下能力：

（1）领导能力。项目经理必须具备高超的领导才能和强烈的科技意识和较强的业务处理能力。领导能力，简单地说，通过项目团队来达到目标。

首先，项目经理应懂得如何授权和分配职责，采取参与和顾问式的领导方式，发挥导向和教练作用，让成员在职责范围内充分发挥能动性，自主地完成项目工作；

其次，项目经理应善于激励。由于项目经理通常没有太大的权力对成员进行物质方面的激励，因此，非物质激励方式就特别重要。例如，借助项目的唯一性，给项目成员接受挑战的机会往往可以对优秀的项目成员起到极大的激励作用；另外，对项目成员的工作成绩要及

时表示认可。及时是非常重要的，并且最好是当众表扬，例如，在上级领导或客户面前对项目团队或具体成员做出正面的评价。

第三，项目经理应该为成员树立榜样，表现出积极的心态，成为团队的典范和信心的源泉。只有身先士卒，各方面以身作则，才能得到广大开发人员的认可和信任，才能树立较高的威信。

第四，项目经理应该能够果断抉择，负责人的重要任务是决策，特别是有多种选择的情况下，一个正确的选择往往事半功倍。

（2）沟通技巧。有效的沟通是项目顺利进行的保证，沟通及时、集思广益、步调一致，才能取得项目最终的成功。项目过程中，项目经理需要通过多种渠道保持与团队及分包商、客户方、公司上级的定期交流沟通，及时了解项目的进程、存在的问题及获得有益的建议。沟通的方式可以是口头的或书面的，如：面谈、电话、邮件、会议等。在沟通过程中，项目经理应善于提问，并做到有效地聆听，能经常站在对方的角度思考问题。

（3）人际交往能力。良好的人际关系有助于项目的协调，避免生硬的操作方式。项目经理必须积极对外联络，充分利用外部资源，例如其他部门做过类似项目者，可以向他们取经甚至直接获得源码。这对一个项目争取时间，避免重复工作很重要项目协调是随时需要的，主要来自于项目内部及客户，可能是资源的配置问题，也可能是项目范围的调整等。人际交往需要从一点一滴做起，而且往往发生在项目工作之外，项目经理需要采取主动、热情的姿态。

（4）应付压力的能力。项目的特点决定了项目工作过程存在一定的不可预见性，项目经理需要做好随时面对压力甚至是冲突的准备。一旦面临压力或冲突，最重要的是保持冷静，避免使项目陷入困境。项目经理要以乐于解决问题的姿态出现在团队及上级或客户面前。

（5）培养员工的能力。出色的项目经理重视对项目成员的培养，通过项目过程使小组每个成员都能发挥才能并提升员工的能力，促进员工的自我发展。项目经理要帮助成员明晰自己的职业与技能发展方向，分配合适的工作任务，鼓励学习和相互交流，让项目小组成员具有很强的成就感。

（6）时间管理技能。当需要在同一时段处理两项以上的任务时，时间管理就是必要的。而项目经理往往需要同时面对数项甚至十几项任务，可见有效的时间管理是极为重要的。项目经理不仅需要管理好自己的时间，还需要与相关部门及人员订立时间使用协议，尽量减少非预期的时间占用。

合格的项目经理具有敏锐的洞察力，能瞄准目标，实事求是，精心组织，坚决果断，灵

活应变，享有信誉；善于制订计划，解决问题，沟通信息；具有良好的市场意识和交际能力。

当然同时满足这些条件比较困难，但是他应该具有实现这些条件的素质，并注重经验的积累、素质的提高和能力的培养。

3．团队建设项目团队的建设既包括提高项目干系人作为个人做出贡献的能力，也包括提高项目团队作为集体发挥作用的能力。个人的培养（管理能力与技术水平）是团队建设的基础，而团队建设则是项目实现其目标的关键。

因为软件开发是一项长期艰苦的工作，一个团结、协作的团体才能在规定的时间内完成一个质量上乘的软件项目。团队中的每个人必须积极融入整个集体中，不能互相推诿，更不能互相埋怨和指责，正确的态度是大家在充分信任的基础上团结协作、互相帮助、主动承担任务，利用集体的智慧获得成功。整个团队就是一部机器，只有每一个齿轮都能正常运作，才能生产出优质的产品。

人是最宝贵的资源，在软件项目中，应该为软件开发人员和管理人员等各类项目人员营造一个和谐、良好的工作氛围，为开发人员创造出一个人尽其才的环境也是项目成功的重要环节，让他们能得心应手地施展自己的才华，特别在工作安排上要煞费苦心，针对每个人不同的特长，根据项目的具体环境和条件把人员合理地安排在恰当的岗位上。使他们能感到项目成功的把握并有积极的工作心态，将项目作为自己事业的一部分，确保项目队伍的稳定性和连续性。在项目结束之际，项目团队的各个成员是否感到他们从自己的经历中学到了一些知识、是否喜欢为这次项目工作，以及是否希望参与组织的下一个项目都是非常重要的。

对于软件项目团队的成长规律，有人归纳出了以下 4 个阶段：

（1）形成阶段。形成阶段促使个体成员转变为团队成员。每个人在这一阶段都有许多疑问：我们的目的是什么？其他团队成员的技术、人品怎么样？每个人都急于知道他们能否与其他成员合得来，自己能否被接受。

为使项目团队明确方向，项目经理一定要向团队说明项目目标，并设想出项目成功的美好前景及成功所产生的益处；公布项目的工作范围、质量标准、预算及进度计划的标准和限制。项目经理在这一阶段还要进行组织构建工作，包括确立团队工作的初始操作规程，规范沟通渠道、审批及文件记录工作。所以在这一阶段，对于项目成员采取的激励方式主要为预期激励、信息激励和参与激励。

（2）震荡阶段。这一阶段，成员们开始着手执行分配到的任务，缓慢地推进工作。现实也许会与个人当初的设想不一致。例如，任务比预计的更繁重或更困难；成本或进度计划的限制可能比预计的更紧张；成员们越来越不满意项目经理的指导或命令。

震荡阶段的特点是人们有挫折、愤怨或者对立的情绪。这一阶段士气很低，成员可能会抵制形成团队，因为他们要表达与团队联合相对立的个性。

因此在这一阶段，项目经理要做导向工作，致力于解决矛盾，决不能希望通过压制来使其自行消失。这时，对于项目成员采取的激励方式主要是参与激励、责任激励和信息激励。

（3）正规阶段。经受了震荡阶段的考验，项目团队就进入了发展的正规阶段。项目团队逐渐接受了现有的工作环境，团队的凝聚力开始形成。这一阶段，随着成员之间开始相互信任，团队内大量地交流信息、观点和感情，合作意识增强，团队成员互相交换看法，并感觉到他们可以自由地、建设性地表达他们的情绪及意见。

在正规阶段，项目经理采取的激励方式除参与激励外，还有两个重要方式：一是发掘每个成员的自我成就感和责任意识，引导员工进行自我激励；二是尽可能地多创造团队成员之间互相沟通、相互学习的环境，以及从项目外部聘请专家讲解与项目有关的新知识、新技术，给员工充分的知识激励。

（4）表现阶段。团队成长的最后阶段是表现阶段。这时，项目团队积极工作，急于实现项目目标。这一阶段的工作绩效很高，团队有集体感和荣誉感，信心十足。团队能感觉到被高度授权，如果出现技术难题，就由适当的团队成员组成临时攻关小组，解决问题后再将相关知识或技巧在团队内部快速共享。

这一阶段，项目经理需要特别关注预算、进度计划、工作范围及计划方面的项目业绩。如果实际进程落后于计划进程，项目经理就需要协助支持修正行动的制定与执行。这一阶段激励的主要方式是危机激励、目标激励和知识激励。

需要强调的是，对于信息系统建设人才，要更多地引导他们进行自我激励和知识激励。当然，足够的物质激励是不言而喻的，它从始至终都是最有效的激励。

激励的结果是使参与信息系统的所有成员组成一个富有成效的项目团队，这种团队具有如下特点：

能清晰地理解项目的目标；

每位成员的角色和职责有明确的期望；

以项目的目标为行为的导向；

项目成员之间高度信任、高度合作互助。总之，科学地进行团队建设有助于按期、保质、高效、在预算内完成软件项目。

13.6 软件的运行与评价

软件的运行与评价是指软件开发结束后交付用户使用，用户在实际使用中对软件是否符合开发时制定的一系列评价标准进行打分，看是否满足了用户的使用要求。通常，关注如下几点:

（1）软件的稳定性和可靠性评价。软件的稳定性，指软件在一个运行周期内、在一定的压力条件下，软件的出错几率、性能劣化趋势等，并观察其运行环境内的应用服务器、数据库服务器等系统的稳定性。从用户角度看,软件在使用过程中如果出现系统故障、系统反应速度慢等就表明软件本身的可靠性需要提高。通常在软件交付用户使用前都要进行大量的系统功能测试和用户可靠性测试，如果测试工作做得很好，后期运行使用时这方面的问题会减少。

（2）软件是否满足了用户的需求。满足用户的需求是软件开发的基本要求。系统交付使用前，需要用户对系统提供的需求进行评价。用户评价的标准就是归档的需求文档，用户可以根据文档的要求逐个检查系统是否提供了相应的功能。只有软件满足了用户需求，用户才会同意交付使用。

（3）软件实施给用户带来的好处。这是用户需要软件开发的原因。通常体现为价值指标,如节省多少人工成本，节省业务流程时间，减小数据质量出错率等。软件交付用户使用后一段时间，用户可以测量相应的指标是否满足了之前设定的目标，对新系统给予评价。

13.7 软件过程改进

处于激烈市场竞争中的软件开发机构若想在预定的期限内用有限的资金，满足不断增长的软件产品的需求，就必须不断加强软件开发过程的管理，对软件开发的所有环节施加有效的管理和控制，将软件的开发逐渐转变成为一种工业化的生产过程。

软件过程改进（Software Process Improvement，SPI）用于帮助软件企业对其软件生产过程进行计划、过程诊断、改进方案的制订及实施等工作。它的实施对象是软件企业的软件过程，即软件产品的生产过程，也包括配置管理、软件维护等辅助过程。目前，使用最多的软件过程改进模型包括 CMM、CMMI、ISO9000 和 ITIL 等系列标准。

1．CMM

W-CMM（软件能力成熟度模型）为软件企业的过程能力提供了一个阶梯式的进化框架，阶梯共有五级。第一级实际上是一个起点，任何准备按 CMM 体系进化的企业都自然处于

这个起点上，并通过这个起点向第二级迈进。除第一级外，每一级都设定了一组目标，如果达到了这组目标，则表明达到了这个成熟级别，可以向下一个级别迈进。 CMM 体系不主张跨越级别的进化，因为从第二级起，每一个低的级别实现均是高的级别实现的基础。

（1）初始级。初始级的软件过程是未加定义的随意过程，项目的执行是随意甚至是混乱的。也许，有些企业制定了一些软件工程规范，但若这些规范未能覆盖基本的关键过程要求，且执行没有政策、资源等方面的保证时，那么它仍然被视为初始级。

（2）可重复级。根据多年的经验和教训，人们总结出软件开发的首要问题不是技术问题而是管理问题。因此，第二级的焦点集中在软件管理过程上。一个可管理的过程则是一个可重复的过程，一个可重复的过程则能逐渐进化和成熟。第二级的管理过程包括了需求管理、项目管理、质量管理、配置管理和子合同管理五个方面。其中项目管理分为计划过程和跟踪与监控过程两个过程。实施这些过程，从管理角度可以看到一个按计划执行的且阶段可控的软件开发过程。

（3）定义级。在第二级仅定义了管理的基本过程，而没有定义执行的步骤标准。在第三级则要求制定企业范围的工程化标准，而且无论是管理还是工程开发都需要一套文档化的标准，并将这些标准集成到企业软件开发标准过程中去。所有开发的项目需根据这个标准过程，剪裁出与项目适宜的过程，并执行这些过程。过程的剪裁不是随意的，在使用前需经过企业有关人员的批准。

（4）管理级。第四级的管理是量化的管理。所有过程需建立相应的度量方式，所有产品的质量（包括工作产品和提交给用户的产品）需有明确的度量指标。这些度量应是详尽的，且可用于理解和控制软件过程和产品。量化控制将使软件开发真正变成一种标准的工业生产活动。

（5）优化级。第五级的目标是达到一个持续改善的境界。所谓持续改善是指可根据过程执行的反馈信息来改善下一步的执行过程，即优化执行步骤。如果一个企业达到了这一级，那么表明该企业能够根据实际的项目性质、技术等因素，不断调整软件生产过程以求达到最佳。

2．CMMI

CMMI（Capability MaturityModel Integration），即能力成熟度模型集成。CMMI 是 CMM 模型的最新版本。CMMI 与 CMM 最大的不同点在于：CMMISM-SE/SW/IPPD/SS 1.1 版本有四个集成成分，即：系统工程（SE）和软件工程（SW）是基本的科目，对于有些组织还可以应用集成产品和过程开发方面（IPPD）的内容，如果涉及供应商外包管理可以相应地应用

SS（Supplier Sourcing）部分。

CMMI 有两种表示方法，一种是和软件 CMM 一样的阶段式表现方法，另一种是连续式的表现方法。这两种表现方法的区别是：阶段式表现方法仍然把 CMMI 中的若干个过程区域分成了 5 个成熟度级别，帮助实施 CMMI 的组织建议一条比较容易实现的过程改进发展道路。而连续式表现方法则通过将 CMMI 中过程区域分为四大类：过程管理、项目管理、工程及支持。对于每个大类中的过程区域，又进一步分为基本的和高级的。这样，在按照连续式表示方法实施 CMMI 的时候，一个组织可以把项目管理或者其他某类的实践一直做到最好，而其他方面的过程区域可以完全不必考虑。

3．ISO 9000

国内软件公司采用的 ISO 9000 系列质量体系认证通常有 ISO 9001 的 1994 年版和 2000 年版。ISO 9001 和 CMM 非常相似的是，两者都共同着眼于质量和过程管理，而且它们都是基于戴明博士的全面质量管理 TQM 产生的，因此不存在任何矛盾的地方。但是，它们的基础是不同的：ISO9001（ISO9000 标准系列中关于软件开发和维护的部分）确定一个质量体系的最少需求，而 CMM 强调持续过程改进。在 1994 年版的 ISO 9001 中，CMM 2 级的 6 个关键过程区域所涉及的部分，基本上都比较明确地作出了要求；而 CMM 3 级的 7 个关键过程区域中所涉及的内容大多数都提到了，但做出的要求不是非常详细。很多实施了 94 版 ISO9000 的企业在了解了 SW-CMM 以后，普遍反映 CMM 比 ISO 的要求明确、详细得多。如果 94 版 ISO 实施的效果很好的话，实施 CMM 2 级工作量是可以减少很多的。而 2000 版的 ISO 则和 CMM 有更多的直接对应的关系，甚至是大量 CMM 4 级和 5 级的要求。

4．ITIL

ITIL（信息技术基础设施库）是英国政府中央计算机与电信管理中心（CCTA）在 20 世纪 90 年代初期发布的一套 IT 服务管理最佳实践指南。在此之后，CCTA 又在 HP、IBM、BMC、CA 和 Peregrine 等主流 IT 资源管理软件厂商近年来所做出的一系列实践和探索的基础之上，总结了 IT 服务的最佳实践经验，形成了一系列基于流程的方法，用以规范 IT 服务的水平，并在 2000 年至 2003 年期间推出新的 ITIL V2.0 版本，于 2007 年推出 V3.0 版本。

ITIL 为企业的 IT 服务管理实践提供了一个客观、严谨、可量化的标准和规范，企业的 IT 部门和最终用户可以根据自己的能力和需求定义自己所要求的不同服务水平，参考 ITIL 来规划和制定其 IT 基础架构及服务管理，从而确保 IT 服务管理能为企业的业务运作提供

更好的支持。对企业来说，实施 ITIL 的最大意义在于把 IT 与业务紧密地结合起来，从而让企业的 IT 投资回报最大化。

第 14 章：信息系统基础知识

信息系统是一个由人、计算机等组成的能进行信息的收集、传递、存储、加工、维护和使用的系统，它是一门综合了经济管理理论、运筹学、统计学、计算机科学的系统性和边缘性学科，是一门尚处在不断发展完善的多元目的的新兴学科。

信息系统包含三大要素，分别是系统的观点、数学的方法和计算机应用。而它又不同于一般的计算机应用，它能够充分利用数据资源为企业经营管理服务；它能够利用信息和模型辅助企业决策，从而预测和控制企业行为。信息系统是企业提升核心竞争力的重要和有力的武器。

14.1 信息系统概述

信息系统（Information System，IS）一般泛指收集、存储、处理和传播各种信息的具有完整功能的集合体。在这里，信息系统并没有强调收集、存储、处理和传播信息所用的工具。一般意义上的信息系统在任何时代、任何社会都会存在，然而，只有到了今天，信息系统的概念才被创造出来，并得到相当程度的普及，这是因为，在当今社会，信息系统总是与计算机技术和互联网技术的应用联系在一起，因此，现代的信息系统总是指以计算机为信息处理工具、以网络为信息传输手段的信息系统。因此，现如今说到的信息系统，一般来说，就是指这样的信息系统，而不必特意说明是“现代”信息系统。

14.1.1 信息系统的组成

由于其广泛的应用，当今的信息系统已经发展成为一个极为庞大的家族，而且几乎每个信息系统其内部构成都非常复杂。

为了充分认识信息系统，要从多种角度进行分析。

1．信息系统的数据环境目前对于信息系统最为权威的分类方法是世界信息系统大师詹姆斯·马丁的分类。马丁从信息系统的数据环境的角度出发，对信息系统进行分类。

马丁在《信息工程》和《战略数据规划方法学》中将信息系统的数据环境分为 4 种类

型，并认为清楚地了解它们之间的区别是很重要的，因为它们对不同的管理层次，包括高层管理的作用是不同的。

第一类数据环境：数据文件。其特征是：没有使用数据库管理系统，根据大多数的应用需要，由系统分析师和程序员分散地设计各种数据文件。其特点是简单，相对容易实现。但随着应用程序增加，数据文件数目剧增，导致很高的维护费用；一小点应用上的变化都将引起连锁反应，使修改和维护工作缓慢且费用高昂，并很难进行。

第二类数据环境：应用数据库。这类信息系统，虽然使用了数据库管理系统，但没达到第三类数据环境的共享程度。分散的数据库为分散的应用而设计。实现起来比第三类数据环境简单。像第一类数据环境一样，随着应用的扩充，应用数据库的个数，以及每个数据库中的数据量也在急剧增加，随之而导致维护费用的大幅度增高，有时甚至高于第一类数据环境。该类数据环境还没有发挥使用数据库的主要优越性。

第三类数据环境：主题数据库。信息系统所建立的主题数据库与一般具体的应用有很大的区别，它有很强的独立性，数据经过设计，其存储的结构与使用它的处理过程都是独立的。各种面向业务主题的数据，如顾客数据、产品数据或人事数据，通过一些共享数据库被联系和体现出来。这种主题数据库的特点是：经过严格的数据分析，建立应用模型，虽然设计开发需要花费较长的时间，但其后的维护费用很低。最终（但不是立即）会使应用开发加快，并能使用户直接与这些数据库交互使用数据。主题数据库的开发需要改变传统的系统分析方法和数据处理的管理方法。但是，如果管理不善，也会蜕变成第二类甚至是第一类数据环境。

第四类数据环境：信息检索系统。一些数据库经过组织能保证信息检索和快速查询的需要，而不是大量的事务管理。软件设计中要采用转换文件、倒排表或辅关键字查询技术。新的字段可随时动态地加入到数据结构中。有良好的最终用户查询和报告生成软件工具。大多数用户掌握的系统都采用第四类数据库。这种环境的特点是：比传统的数据库有更大的灵活性和动态可变性。一般应该与第三类数据环境共存，支持综合信息服务和决策系统。

在数据库技术逐渐普及，软件工程方法得到推广的一二十年中，不同的企业单位相继开展计算机应用，从而形成了多种多样的数据环境；这些企业的高层领导和数据处理部门或迟或早都会认识到，需要对现存的数据环境进行改造，以保证信息需求的不断提高，克服现行计算机在数据处理方面的问题，提高科学管理水平，这就需要进行战略数据规划。还有一些企业单位，计算机应用刚刚起步，或者准备开展计算机应用，需要吸取别人的经验教训，避免走错路、走弯路。如果有先进的方法论作指导，就会快速、科学地实现目标，这就更需要这种战略性的、奠基性的规划工作——战略数据规划。对于前一类单位，通过战略数据规划，

尽快地将现有数据环境转变到第三类、第四类数据环境，以保证高效率、高质量地利用数据资源。对于后一类单位，战略数据规划是整个计算机应用发展规划的基础与核心，是计算机设备购置规划、人才培训规划和应用项目开发规划的基础。两类单位搞战略数据规划的共同目标是分析、组织、建立企业稳定的数据结构，规划各种主题数据库的实施步骤和分布策略，为企业管理计算机化打下坚实的基础。

2．信息系统的应用层次

一个公司的管理活动可以分成 4 级：战略级、战术级、操作级和事务级，相应的，信息系统就其功能和作用来看，也可以分为 4 种类型，即战略级信息系统、战术级信息系统、操作级信息系统和事务级信息系统。不同级别的信息系统的所有者和使用者都是不同的。一般来说，战略级的信息系统的所有者和使用者都是企业的最高管理层，对于现代公司制企业，就是企业的董事会和经理班子；战术级信息系统的使用者一般是企业的中层经理及其管理的部门；操作级信息系统的使用者一般是服务型企业的业务部门，例如，保险企业的保单处理部门；事务级信息系统的使用者一般是企业的管理业务人员，例如，企业的会计、劳资员等。

希赛教育专家提示：以上不同层级的信息系统，都属于一个大的信息系统的子系统。信息系统是企业信息化的基础性工程，从某种角度说，企业信息化就是信息系统的建设和运行。

14.1.2 信息系统的生命周期

信息系统与其他事物一样，也要经历产生、发展、成熟和消亡的过程。人们把信息系统从产生到消亡的整个过程称为信息系统的生命周期。

一般来说，信息系统的生命周期分为 4 个阶段，即产生阶段、开发阶段、运行阶段和消亡阶段。

1．信息系统的产生阶段

信息系统的产生阶段，也是信息系统的概念阶段或者是信息系统的需求分析阶段。这一阶段又分为两个过程，一是概念的产生过程，即根据企业经营管理的需要，提出建设信息系统的初步想法；二是需求分析过程，即对企业信息系统的需求进行深入的调研和分析，并形成需求分析报告。

2．信息系统的开发阶段信息系统的开发阶段是信息系统生命周期中最重要和最关键的阶段。该阶段又可分为 5 个阶段，即，总体规划、系统分析、系统设计、系统实施和系统验收阶段。

（1）总体规划阶段。信息系统总体规划是系统开发的起始阶段，它的基础是需求分析。以计算机和互联网为工具的信息系统是企业管理系统的重要组成部分，是实现企业总体目标的重要工具。因此，它必须服从和服务于企业的总体目标和企业的管理决策活动。总体规划的作用主要有：

指明信息系统在企业经营战略中的作用和地位。

指导信息系统的开发。

优化配置和利用各种资源，包括内部资源和外部资源。

通过规划过程规范企业的业务流程。

一个比较完整的总体规划，应当包括信息系统的开发目标、信息系统的总体架构、信息系统的组织结构和管理流程、信息系统的实施计划、信息系统的技术规范等。

（2）系统分析阶段。系统分析阶段的目标是为系统设计阶段提供系统的逻辑模型。系统分析阶段以企业的业务流程分析为基础，规划即将建设的信息系统的基本架构，它是企业的管理流程和信息流程的交汇点。系统分析的内容主要应包括，组织结构及功能分析、业务流程分析、数据和数据流程分析、系统初步方案等。

（3）系统设计阶段。系统设计阶段是根据系统分析的结果，设计出信息系统的实施方案。系统设计的主要内容包括，系统架构设计、数据库设计、处理流程设计、功能模块设计、安全控制方案设计、系统组织和队伍设计、系统管理流程设计等。

（4）系统实施阶段。系统实施阶段是将设计阶段的结果在计算机和网络上具体实现，也就是将设计文本变成能在计算机上运行的软件系统。由于系统实施阶段是对以前的全部工作的检验，因此，系统实施阶段用户的参与特别重要。如果说在系统设计阶段以前，用户处于辅助地位的话，而到了系统实施阶段以后，用户就应逐步变为系统的主导地位。

（5）系统验收阶段。信息系统实施阶段结束以后，系统就要进入试运行。通过试运行，系统性能的优劣、是否做到了用户友好等问题都会暴露在用户面前，这时就进入了系统验收阶段。

3．信息系统的运行阶段

当信息系统通过验收，正式移交给用户以后，系统就进入了运行阶段。一般来说，一个性能良好的系统，运行过程中会较少出现故障，即使出现故障，也较容易排除；而那些性能较差的系统，运行过程中会故障不断，而且可能会出现致命性故障，有时故障会导致系统瘫痪。因此，长时间的运行是检验系统质量的试金石。

另外，要保障信息系统正常运行，一项不可缺少的工作就是系统维护。在软件工程中，

把维护分为 4 种类型，即排错性维护、适应性维护、完善性维护和预防性维护。一般在系统运行初期，排错性维护和适应性维护比较多，而到后来，完善性维护和预防性维护就会比较多。

4．信息系统的消亡阶段

通常人们比较重视信息系统的开发阶段，轻视信息系统运行阶段，而几乎完全忽视信息系统的消亡阶段。其实，这样做是片面的。因为，计算机技术和互联网技术的发展十分快速，新的技术、新的产品不断出现；同时，由于企业处在瞬息万变的市场竞争的环境之中，在这种情况下，企业开发好一个信息系统想让它一劳永逸地运行下去，是不现实的。企业的信息系统会经常不可避免地会遇到系统更新改造、功能扩展，甚至报废重建的情况。对此，在信息系统建设的初期企业就应当注意系统的消亡条件和时机，以及由此而花费的成本。

14.1.3 信息系统建设的原则

为了能够适应开发的需要，在信息系统规划设计，以及系统开发的过程中，必须要遵守一系列原则，这是系统成功的必要条件。以下是信息系统开发常用的原则。

1．高层管理人员介入原则

一个信息系统的建设目标总是为企业的总体目标服务，否则，这个系统就不应当建设。而真正能够理解企业总体目标的人必然是那些企业高层管理人员，只有他们才能知道企业究竟需要什么样的信息系统，而不需要什么样的信息系统；也只有他们才知道企业有多大的投入是值得的，而超过了这个界限就是浪费。这点是那些身处某一部门的管理人员，或者是技术人员所无法做到的。因此，信息系统从概念到运行都必须有企业高层管理人员介入。当然，这里的“介入”有着其特定的含义，它可以是直接参加，也可以是决策或指导，还可以是在政治、经济、人事等方面的支持。

这里需要说明的是，高层管理人员介入原则在现阶段已经逐步具体化，那就是企业的“首席信息官”（Chief Information Officer，CIO）的出现。CIO 是企业设置的相当于副总裁的一个高级职位，负责公司信息化的工作，主持制定公司信息规划、政策、标准，并对全公司的信息资源进行管理控制的公司行政官员。在大多数企业里，CIO 是公司最高管理层中的核心成员之一。毫无疑问，深度介入信息系统开发建设，以及运行是 CIO 的职责所在。

2．用户参与开发原则

在我国信息系统开发中流行所谓“用户第一”或“用户至上”的原则。当然，这个原则

并没有错，一个成功的信息系统，必须把用户放在第一位，这应该是毫无疑义的。但是，究竟应当怎么“放”？怎么“放”才算是第一位？没有一个确切的标准。而马丁提出的“用户参与开发原则”就把“用户第一原则”具体化了。

用户参与开发原则主要包括以下几项含义：

一是“用户”有确定的范围。究竟谁是用户？人们通常把“用户”仅仅理解成为用户单位的领导，其实，这是很片面的。当然，用户单位领导应该包括在用户范围之内，但是，更重要的用户，或是核心用户是那些信息系统的使用者，而用户单位的领导只不过是辅助用户或是外围用户。

二是用户，特别是那些核心用户，不应是参与某一阶段的开发，而应当是参与全过程的开发，即用户应当参与从信息系统概念规划和设计阶段，直到系统运行的整个过程。而当信息系统交接以后，他们就成为系统的使用者。

三是用户应当深度参与系统开发。用户以什么身份参与开发是一个很重要的问题。一般说来，参与开发的用户人员，既要以甲方代表身份出现，又应成为真正的系统开发人员，与其他开发人员融为一体。

3．自顶向下规划原则

在信息系统开发的过程中，经常会出现信息不一致的问题，这种现象的存在对于信息系统来说往往是致命的，有时，一个信息系统会因此而造成报废的结果。研究表明，信息的不一致是由计算机应用的历史性演变造成的，它通常发生在没有一个总体规划的指导就来设计实现一个信息系统的情况之下。因此，坚持自顶向下规划原则对于信息系统的开发和建设来说是至关重要的。自顶向下规划的一个主要目标是达到信息的一致性。同时，自顶向下规划原则还有另外一个方面，那就是这种规划绝不能取代信息系统的详细设计。必须鼓励信息系统各子系统的设计者在总体规划的指导下，进行有创造性的设计。

4．工程化原则

在 20 世纪 70 年代，出现了世界范围内的“软件危机”。所谓软件危机是指一个软件编制好以后，谁也无法保证它能够正确地运行，也就是软件的可靠性成了问题。软件危机曾一度引起人们，特别是工业界的恐慌。经过探索，人们认识到，之所以会出现软件危机，是因为，软件产品是一种个体劳动产品，最多也就是作坊式的产品。因此，没有工程化是软件危机发生的根本原因。此后，发展成了“软件工程”这门工程学科，在一定程度上解决了软件危机。

信息系统也经历了与软件开发大致相同的经历。在信息系统发展的初期，人们也像软件

开发初期一样，只要作出来就行，根本不管实现的过程。这时的信息系统，大都成了少数开发者的“专利”，系统可维护性、可扩展性都非常差。后来，信息工程、系统工程等工程化方法被引入到信息系统开发过程之中，才使问题得到了一定程度的解决。

其实，工程化不仅是一种有效的方法，它也应当是信息系统开发的一项重要原则。

5．其他原则

对于信息系统开发，人们还从不同的角度提出了一系列原则，例如：

创新性原则，用来体现信息系统的先进性。

整体性原则，用来体现信息系统的完整性。

发展性原则，用来体现信息系统的超前性。

经济性原则，用来体现信息系统的实用性。

14.1.4 信息系统开发方法

企业信息系统对于企业信息化的重要意义是不言而喻的。从实际运行的效果来看，有些信息系统运行得很成功，取得了巨大的经济效益和社会效益；但也有些信息系统效果并不显著，甚至还有个别信息系统开始时还能正常运行，可时间一长，系统就故障不断，最后走上报废之路。是什么导致这样截然不同的结果呢？当然，这里的原因可能很复杂，但有一个原因是十分重要和关键的，那就是信息系统的开发方法问题。

信息系统是一个极为复杂的人－机系统，它不仅包含计算机技术、通信技术，以及其他的工程技术，而且，它还是一个复杂的管理系统，还需要管理理论和方法的支持。下面简单介绍几种最常用的信息系统开发方法。

1．结构化方法

结构化方法是由结构化系统分析和设计组成的一种信息系统开发方法。

结构化方法是目前最成熟、应用最广泛的信息系统开发方法之一。它假定被开发的系统是一个结构化的系统，因而，其基本思想是将系统的生命周期划分为系统调查、系统分析、系统设计、系统实施、系统维护等阶段。这种方法遵循系统工程原理，按照事先设计好的程序和步骤，使用一定的开发工具，完成规定的文档，在结构化和模块化的基础上进行信息系统的开发工作。结构化方法的开发过程一般是先把系统功能视为一个大的模块，再根据系统分析设计的要求对其进行进一步的模块分解或组合。

结构化生命周期法主要特点如下：

（1）开发目标清晰化。结构化方法的系统开发遵循“用户第一”的原则，开发中要保持与用户的沟通，取得与用户的共识，这使得信息系统的开发建立在可靠的基础之上。

（2）工作阶段程式化。结构化方法的每个阶段的工作内容明确，注重开发过程的控制。每一阶段工作完成后，要根据阶段工作目标和要求进行审查，这使得各阶段工作有条不紊，也避免为以后的工作留下隐患。

（3）开发文档规范化。结构化方法的每一阶段工作完成后，要按照要求完成相应的文档，以保证各个工作阶段的衔接与系统维护工作的便利。

（4）设计方法结构化。结构化方法采用自上而下的结构化、模块化分析与设计方法，使各个子系统间相对独立，便于系统的分析、设计、实现与维护。结构化方法被广泛地应用于不同行业信息系统的开发中，特别适合于那些业务工作比较成熟、定型的系统，如银行、电信、商品零售等行业。

2．原型法

原型法是一种根据用户需求，利用系统开发工具，快速地建立一个系统模型展示给用户，在此基础上与用户交流，最终实现用户需求的信息系统快速开发的方法。在现实生活中，一个大型工程项目建设之前制作的沙盘，以及大型建筑的模型等都与快速原型法有同样的功效。应用快速原型法开发过程包括系统需求分析、系统初步设计、系统调试、系统检测等阶段。用户仅需在系统分析与系统初步设计阶段完成对应用系统的简单描述，开发者在获取一组基本需求定义后，利用开发工具生成应用系统原型，快速建立一个目标应用系统的最初版本，并把它提交给用户试用、评价，根据用户提出的意见和建议进行修改和补充，从而形成新的版本，再返回给用户。通过这样多次反复，使得系统不断地细化和扩充，直到生成一个用户满意的方案为止。

希赛教育专家提示：原型法具有开发周期短、见效快、与业务人员交流方便的优点，特别适用于那些用户需求模糊，结构性比较差的信息系统的开发。

3．面向对象方法面向对象方法是对客观世界的一种看法，它是把客观世界从概念上看成一个由相互配合而协作的对象所组成的系统。信息系统开发的面向对象方法的兴起是信息系统发展的必然趋势。数据处理包括数据与处理两部分。但在信息系统的发展过程的初期却是有时偏重这一面，有时偏重那一面。在 20 世纪 70—80 年代，偏重数据处理者认识到初期的数据处理工作是计算机相对复杂而数据相对简单。因此，先有结构化程序设计的发展，随后产生面向功能分解的结构化设计与结构化分析。偏重于数据方面人员同时提出了面向数据结构的分析与设计。到了 20 世纪 80 年代，兴起了信息工程方法，使信息系统开发发展

到了新的阶段。

信息工程在实际应用中既表现出其优越性的一面，同时，也暴露了一些缺点，例如，过于偏重数据，致使应用开发受到影响。而面向对象方法则集成了以前各种方法的优点，避免了各自的一些缺点。

面向对象的分析方法是利用面向对象的信息建模概念，如实体、关系、属性等，同时运用封装、继承、多态等机制来构造模拟现实系统的方法。传统的结构化设计方法的基本点是面向过程，系统被分解成若干个过程。而面向对象的方法是采用构造模型的观点，在系统的开发过程中，各个步骤的共同的目标是建造一个问题域的模型。在面向对象的设计中，初始元素是对象，然后将具有共同特征的对象归纳成类，组织类之间的等级关系，构造类库。在应用时，在类库中选择相应的类。

4. 面向服务的方法

OO 的应用构建在类和对象之上，随后发展起来的建模技术将相关对象按照业务功能进行分组，就形成了构件（Component）的概念。对于跨构件的功能调用，则采用接口的形式暴露出来。进一步将接口的定义与实现进行解耦，则催生了服务和面向服务（Service-Oriented，SO）的开发方法。

从应用的角度来看，组织内部、组织之间各种应用系统的互相通信和互操作性直接影响着组织对信息的掌握程度和处理速度。如何使信息系统快速响应需求与环境变化，提高系统可复用性、信息资源共享和系统之间的互操作性，成为影响信息化建设效率的关键问题，而 SO 的思维方式恰好满足了这种需求。

目前，SO 方法是一个较新的领域，许多研究和实践还有待进一步深入。但是，它代表着不拘泥于具体技术实现方式的一种新的系统开发思想，已经成为信息系统建设的大趋势，越来越多的组织开始实施 SO 的信息系统。

14.2 信息系统工程

本节包括信息系统的概念、信息系统的内容和信息系统总体规划三部分内容。

14.2.1 信息系统工程的概念

系统是由相互作用和相互依赖的若干部分，按一定规律结合成的、具有特定功能的有机整体。系统有下述特性：

（1）集合性。系统是由许多元素有机地组成的整体。每个元素服从整体，追求全局最优。

（2）相关性。系统的各个组成部分之间是互相联系、互相制约的。

（3）目的性。任何系统都是有目的和目标的。

（4）层次性。一个系统往往由多个部门（或部分）组成。每个部门可看作一个小的系统，称为子系统，子系统之下又可划分为子子系统。系统具有层次结构。

（5）环境适应性。任何系统都是存在并活动于一个特定的环境之中，与环境不断进行物质、能量和信息的交换。系统必须适应环境。

1．系统的分类

按照系统功能划分：工业控制系统、信息管理系统、军事系统和经济系统等。

按照系统与外界的关系划分：封闭系统和开放系统。

按照系统的内部结构划分：开环系统和闭环系统等。按照抽象程度将系统分为：概念系统（描述系统的主要特征和大致轮廓）、逻辑系统（脱离实现细节的合理系统）和物理系统（实际存在的系统）。

2．系统工程从不同角度、不同的背景、不同的应用目的，系统工程有不同的定义：

“系统工程是为了更好地达到系统目标，而对系统的构成要素、组织结构、信息流动和控制机理等进行分析与设计的技术”（1967 年，日本工业标准 JIS）。

“系统工程是为了合理地开发、设计和运用系统而采用的思想、程序、组织和方法的总称”（1971 年，日本寺野寿郎，系统工程学）。

“系统工程是一门把已有的学科分支中的知识有效地组合起来用以解决综合性工程问题的技术”（1974 年，大英百科全书）。

“系统工程研究的是怎样选择工人和机器的最适宜的综合方式，以完成特定的目标”（1975 年，美国百科全书）。

“系统工程是组织管理系统的规划、研究、设计、制造、试验和使用的科学方法，是一种对所有系统都具有普遍意义的科学方法”（1982 年，钱学森等，论系统工程）。

“系统工程是按照系统科学的思想，应用信息论、控制论、运筹学等理论，以信息技术为工具，用现代工程方法去研究和管理系统的技术”（1984 年，宋健，系统工程和技术革命）。

归纳各种不同的定义，给出系统工程的定义如下：

系统工程是以研究大规模复杂系统为对象的一门交叉学科。它把自然科学和社会科学的某些思想、理论、方法、策略和手段等根据总体协调的需要，有机地联系起来，应用定量和

定性分析相结合的方法和计算机等技术工具，对系统的构成要素、组织结构、信息交换和反馈控制等功能进行分析、设计、制造和服务，从而达到最优设计、最优控制和最优管理的目的。

3．信息系统信息系统一般泛指收集、存储、处理和传播各种信息的具备完整功能的集合体。人们常说的信息系统大多数支持各部门和机构的管理和决策的信息系统，当前信息系统重要的特征是计算机和互联网技术的应用。

现代信息系统是以计算机为信息处理工具，以网络为信息传输手段的；它最大限度地屏蔽了时间和空间的限制，使人们能以最快捷的方式获取所需信息并加以利用。

计算机应用于企业是从最基础的数据处理开始的。随着企业业务需求的增长和技术条件的发展，人们逐步将计算机应用于企业局部的管理，如财会管理、销售管理、物资管理、生产管理等，即计算机应用发展到对企业的局部事务的管理，形成了所谓事务处理系统。在此基础上，逐步形成对企业全局性的、整体性的计算机应用，并发展形成管理信息系统。管理信息系统强调以企业管理系统为背景，以基层业务系统为基础，强调企业各业务系统间的信息联系，以完成企业总体人物为目标，它能提供企业各级领导从事管理需要的信息，但其收集信息的范围还更多地侧重于企业的内部。随着网络的普及，计算机信息系统已经从管理信息系统发展成为更强调支持企业高层领导决策的决策支持系统，即 DSS 阶段。

4．信息系统工程

将系统工程的理论、方法应用到信息系统，并结合信息系统自身特点，就形成了信息系统工程。信息系统工程应强调研究法的整体性、系统性，技术应用的综合性和项目管理的规范化、标准化。

研究方法的整体性就是把研究对象看作一个由若干个子系统有机结合的整体来分析和设计。对各子系统的技术首先要从实现整个系统技术协调的观点来考虑，从总体最优的需求来选择解决方案。研究方法的系统性要求研究方法反映和顺应客观事物和系统自身的特征和运动规律，例如面向对象分析方法引入对象、属性、方法等概念，相比于面向过程的分析方法更加能够表现自然对象主体特征，而类的概念又比较自然地体现了自然界客观事物的层次性和客观本质，因此被广泛采用。信息系统应用广泛、覆盖面宽，是一个高度综合性的学科领域，客观上要求综合运用各专业领域和信息技术知识，使各种知识技术无缝结合而达到系统整体优化的设计和实现目标。同时也对信息系统开发者提出较高要求，不仅自身要掌握牢固的信息技术知识，而且要有良好的与用户沟通的能力。

一个复杂的信息系统工程客观上存在两个并行工程，一个是工程技术进程，一个是对工

程技术进程的管理控制进程。后者包括工程的规划、组织、控制、进度安排，对各种方案进行分析、比较和决策、评价选定方案的技术效果等。项目管理的规范化、标准化对信息系统的设计、实现、运行、维护、升级都具有重要意义，是系统工程的关键。在研制信息系统时，必须按照系统的方法，明确划分各个工作阶段，确保每个阶段的工作都得到有效的控制管理，对各阶段的工作成果都有规范的文档和审查标准。

14.2.2 信息系统工程的内容

信息系统工程作为一门综合性技术，与多种学科和技术有着深刻的内在联系。总体上，它涉及社会和技术两大领域，并综合应用了管理科学、系统科学、数学、计算机科学、行为科学的研究成果，逐渐形成了信息系统工程的学科体系。

信息系统工程的内容主要包括几个方面：一是体系，二是技术，三是管理。

1．信息系统工程的体系构成

原信息产业部制定的《信息系统工程监理暂行规定》将信息系统分为信息网络系统、信息资源系统和信息应用系统三类，并对每一类的含义进行了界定：

信息网络系统是指以信息技术为主要手段建立的信息处理、传输、交换和分发的计算机网络系统。

信息资源系统是指以信息技术为主要技术手段建立的信息资源采集、存储、处理的资源系统。

信息应用系统是指以信息技术为主要手段建立的各类业务管理的应用系统。

以上三个系统一般情况下，并不是彼此独立的，而是一个大的信息系统的组成部分，只不过是不同的信息系统各有所侧重。

2．信息系统工程的技术构成

软件工程和信息工程是信息系统工程的技术基础，因此，信息系统工程首要的任务是实施软件工程和信息工程。由于信息系统工程其内容不仅包括软件工程和信息工程，而且，它面向组织管理，因而，信息系统工程还应包括组织中的业务流程等内容。软件工程是开发、运行、维护和修复软件的系统方法。其中，“软件”的定义为：计算机程序、方法、规则、相关的文档资料以及在计算机上运行时所必需的数据的集合。这里尤其要注意“软件”与“程序”两个概念的区别。

软件工程包括 3 个要素：方法、工具和过程。方法为软件开发提供了“如何做”的技术，包括项目计划与估算、需求分析、数据结构、系统总体结构的设计、算法过程的设计、编码、

测试及维护等；工具包括各种软件工具、开发机器和开发过程信息库，提供自动或半自动的软件开发环境；过程定义了方法使用的顺序、要求交付的文档资料、为保证质量和协调变化所需要的管理及软件开发各个阶段完成的里程碑。

信息工程是指以数据系统为基础，建立一个信息化企业所需要的一套相互关联的原则。信息工程的主要焦点是用计算机来存储和维护数据，而信息则是从这些数据提炼出来，来满足人们的某种需要的数据。

在信息化的过程中，要以数据为中心，而数据的存储和管理是通过各种数据系统软件来支持的。这是因为，一个企业的数据类型变化不能也不会太大。数据是按实体存储的。除了在极特殊的情况下需要加入新的实体类型外，在一项业务活动的生命周期中，实体类型一般不会变化，至少不会有太大变化，甚至实体的属性类型也很少变化。而经常变化的则是数据类型的值。因此，只有数据被正确地标识和结构化时，数据才有生命力，才能被灵活地使用。

由于基本数据类型是稳定的，而数据处理过程是趋于变化的，所以面向过程会经常遭受失败，而面向数据就有可能会成功。采用面向过程的技术所产生的系统，实施缓慢且难于变化，而信息工程则面向数据及数据处理，这样做就可以实现满足管理者不断变化的信息需求。一旦所需要的数据基础结构建立起来，就可以使用高级数据库语言和应用过程生成器工具很快地得到所要的结果。

无疑，当前的信息工程方法大量地吸收了软件工程的很多技术成果，因为从某种程度上来观察，软件工程实际上可认为是信息工程的一部分。信息工程方法的主要特点：一是以数据为中心，进一步的工作是建立主题数据库；二是将工程的实施划分为对业务系统的实施和对技术系统的实施。前者包含了软件的技术内容，而后者则包含了诸如硬件、网络等工程内容。实施信息工程就是将企业的业务系统与技术系统有机地结合起来。

3．组织流程管理

组织和流程是两个既有联系又有区别的概念。一个组织必然有工作，而有工作就必然有流程；任何流程必然发生在某一个具体的组织之内。因此，组织流程管理是一个组织管理的重点和中心内容，也是信息系统工程的重要内容。

组织流程管理模式的主要工作在于处理好组织内各流程之间的关系，合理地在各流程之间分配资源。因此，必须建立有效的组织保障，这样才能保证流程管理工作的连续性和长期性。有效的组织保障包括：

（1）建立流程管理机构，这一机构可归入管理流程之中。

（2）配备强有力的领导来负责内部的流程管理工作。

（3）制定各流程之间的动态关系规则。通过实施流程管理模式，传统组织中的组织图将不复存在，取代它的是流程管理图。

组织流程管理需要处理大量的信息，必须以快速而灵敏的信息网络来支持。通过流程管理信息系统，决策者可以及时掌握必需的决策信息。信息系统的建设，一方面构造公司内部的信息网络；另一方面要与公司外部的信息网络联结，充分利用外部的信息资源。

4．信息系统工程的管理信息系统工程首先是一项工程，而一项工程一般总是被定义为一个项目，因此，信息
系统工程管理也可以视同为项目管理。

所谓“项目”是指在特定的条件下，具有特定目标的一次任务，是在一定时间内，满足一系列特定目标的多项相关工作的总称。

以上关于项目的定义包含三层含义，一是项目是一次性的任务，且有特定的范围和要求；二是在一定的组织机构内，利用有限资源在规定时间内完成任务；三是任务要满足一定性能、质量、数量、技术指标等要求。所以可以用项目管理的思想和方法来指导信息系统的建设。

项目管理就是把各种资源应用于目标，以实现项目的目标，满足各方面既定的需求。项目管理由环境、资源、目标、组织等诸多要素构成，重点是成本、质量和进度的“三大” 控制。

在具体实施过程中，项目管理一般包含了以下几个方面的内容：

（1）任务划分。任务划分是把整个开发工作定义成一组任务的集合，这组任务又可以进一步划分成若干个子任务，进而形成具有层次结构的任务群。

（2）计划安排。依据划分完毕的任务即可制定出整个开发及项目管理计划，并产生完成任务的计划表。

（3）经费管理。经费管理在整个开发项目管理中处于重要的地位。项目经理可以运用经济杠杆来控制整个开发工作。

（4）审计控制。按照所采用的开发方法，应针对每一类开发人员制定出工作过程中的责任、义务、完成任务的质量标准等，按照计划对每项任务进行审计。

（5）风险管理。如何有效地管理和控制风险是保证系统实施成功的重要环节之一。特别是信息系统工程项目，风险管理更是重中之重。

（6）质量保证。质量管理应贯穿于整个项目始终。在项目规划阶段，就应该建立系统质量的度量模型和相应的机制，对项目质量提出总体的要求；在系统分析和设计阶段应对质量管理不断细化，按自顶向下的方式将总体要求划分成若干易于考核和度量的质量单元。

14.2.3 信息系统工程的总体规划

信息系统工程总体规划是信息系统工程生命周期的第一阶段，它决定了信息系统工程方向、规模、深度和效果。

1．信息系统工程总体规划的概念

信息系统工程总体规划是信息系统工程生命周期的第一阶段。这一阶段的主要目标是明确系统整个生命周期内的发展方向、系统规模和开发计划。信息系统工程建设是投资大、周期长、复杂度高的社会技术系统工程。科学的规划可以减少盲目性，使系统有良好的整体性、较高的适应性，建设工作有良好的阶段性，以缩短系统开发周期，节约开发费用。目前，我国开发的信息系统，单项应用的多，综合应用的少。有的系统适应性差，难于扩充。缺乏科学的规划是造成这种现象的原因之一，有些规模较大的项目，由于没有系统规划和科学论证，上马时轰轰烈烈，上马后困难重重、骑虎难下，不仅造成资金、人力的巨大浪费，而且为今后的系统建设留下隐患。信息系统工程总体规划是信息系统建设成功的关键之一，它比具体项目的开发更为重要。

凡事预则立，不预则废。科学的规划对于任何需要经过较长时间努力才能实现的事情都是非常重要的。现代社会组织，特别是企业的结构和活动内容都很复杂，实现一个组织的信息管理计算机化需要经过长期的努力，因而必须对一个组织的信息系统工程的建设进行规划，根据组织的目标和发展战略及信息系统工程建设和客观规律，并考虑到组织面临的内外环境，科学地制定信息系统工程的发展战略和总体方案，合理安排系统建设的进程。

（1）总体规划的需求随着科学技术的进步和社会经济的发展，特别是国际化和信息化的推进，信息系统工程建设的需求日趋紧迫。尽管信息系统工程已经有了很大的发展，但不少已经建成或正在建设的系统仍然存在一系列问题，主要如下：

系统建设与组织发展的目标和战略不匹配。

已建成的系统解决问题的有效性低，即系统建成后对管理并无显著改善。

不能适应环境变化和组织变革的需要。

组织结构陈旧，管理落后。

系统使用人员的素质较低。

系统开发环境落后，技术方案不合理。

系统开发及运行维护的标准、规范混乱。

资源短缺，投入太少，而对系统的期望又过高。

造成以上问题的原因是多方面的，其中一个主要原因就是人们更多地关心怎样建设一个信息系统工程，而对于建设一个什么样的信息系统工程却注意不够。对于系统的具体方案考虑较少，对总体方案与发展战略问题不够重视。总之，在系统建设中，往往缺乏科学的、有效的系统规划。

（2）总体规划的主要任务总体规划阶段的主要目标就是制定出信息系统工程的长期发展方案，决定信息系统工

程在整个生命周期内的发展方向、规模和发展进程。这样做能为以后的系统分析和设计打好基础。这个阶段的主要任务是：

制定信息系统工程的发展战略。主要是使信息系统工程的战略与整个组织的战略和目标协调一致。

确定组织的主要信息需求，形成信息系统工程的总体结构方案，安排项目开发计划。

制定系统建设的资源分配计划，即制定为实现开发计划而需要的硬软件资源、数据通信设备、人员、技术、服务和资金等计划，提出整个系统的建设概算。

（3）总体规划主要步骤

进行信息系统工程的总体规划一般包括以下几个阶段。

① 对当前系统进行初步的调查。系统分析师从各级干部、相似的企业和本企业内部收集各种信息，站在管理层的高度观察组织的现状，分析系统的运行状况。初步调查主要由两部分构成：

一般调查。一般调查包括组织的概括，企业的目标，现行系统运行情况，简单历史，企业的产品，产量，利税，体制及改革情况，人员基本情况，面临的问题，企业的中长期计划及主要困难等，使系统分析师对企业有一个初步轮廓。

信息需求初步调查。信息需求初步调查是整个初步调查的主要内容。通过调查组织系统的工作职责及活动来了解各职能机构所要处理的数据，估计各机构发生的数据量及频度。信息需求初步调查还应调查环境信息，包括内部环境和外部环境的信息。

② 分析和确定系统目标。这实际上可以由总经理和信息系统工程开发的领导小组确定，应包括服务的质量和范围、政策、组织及人员等。它不仅包括信息系统工程的目标，而且应有整个企业的目标。

③ 分析子系统的组成及基本功能。从上到下对系统进行划分，并且详细说明各个子系统应该实现的功能。

④ 拟定系统的实施方案。可以对子系统的优先级进行设定，以便确定子系统的开发顺

序。

⑤ 进行系统的可行性研究。

⑥ 编写可行性报告。

2．信息系统工程总体规划的目标信息系统工程总体规划的目标就是信息系统工程的方向和指导方针。

（1）总体规划目标的意义

任何一个组织或工作都要有一个明确的目标，特别是像信息系统工程这样复杂的事项，确定目标更为重要，它是工程中所有活动的指南，是工程成功与否的决定因素。

信息系统工程建设初期的总体规划是完成后续工作，如详细定义信息系统工程的需求、优化企业各类业务流程、在项目实施中控制需求范围，保证信息系统工程顺利实施、取得良好效果的关键所在。

实现一个信息系统工程是一项浩大的工程，它是以软件开发为主要内容的研制过程，与一般工程项目相比，有其特殊的规律，要想做好这项工作，在开发过程的各个阶段都要依据总体规划确定的目标、任务及总体开发方法，做好系统开发的组织管理工作、各个应用项目的设计，这是实现系统组织目标的基本保障。信息系统工程的总体规划是进行具体开发工作的必要准备和基本依据。

（2）总体规划目标的功能

信息系统工程总体规划实施成功与否的关键在于企业最高层领导的全力支持和高层管理干部的亲自参加。因为信息系统工程的总体规划本身需要对企业的战略目标、远景规划和当前各个部门的业务活动有着深入的理解和丰富的实践经验。许多工作应当由主要业务管理人员来做，而不能单纯依赖数据库设计人员，更不能靠编程人员代替完成。整个工作需要组织好一批业务管理骨干，需要一定的资金投入和实施时间。所以，没有企业最高领导始终如一的理解、支持，真抓、实干，这项工作就无法进行或者完成得很好。

这里要注意两个原则：

一是从组织的战略出发，而不是从系统的需求出发，这样就可以避免脱离组织宗旨和战略目标，走入为建设而建设的困境；

二是从业务的变革出发，而不是从技术的变革出发，这样有利于充分利用组织的现有资源来满足关键需求，从而避免信息系统工程无法有效地支持组织决策。

信息系统工程规划的主要目的是根据企业总体目标规划好统一的、既有集中式又有分布式、分期逐步实现的数据平台和应用平台的建设。

传统的应用开发不搞总体规划，分散地搞单项应用或互不相关的小系统的开发，短时间内是有效果的。但是随着项目的增多，数据收集、处理上的大量重复，数据冗余和不一致的问题将越来越严重。系统运行将陷入病态，甚至导致瘫痪。

在开发过程的各个阶段都要依据总体规划确定的目标、任务，以总体开发方法去工作，搞好系统开发的组织管理工作，做好各个应用项目的设计，这是实现系统组织目标的基本保障。信息系统工程的规划是进行具体开发工作的必要准备和基本依据。

信息系统工程总体规划的本质，是基于组织战略的信息系统工程战略。因此，在建设信息系统工程过程中，应当全面分析组织所处的环境、战略目标、组织结构、标准作业过程甚至它的文化，并从中找到所要建设的信息系统与组织的关系及其应当起的作用。

（3）总体规划目标的分析过程

总体规划目标的分析过程包括，确定组织的总体目标和确定信息系统工程规划的总体目标两个步骤。

① 确定组织的总体目标。在开发信息系统工程时，关键的任务是明确组织的总体目标。它的分析的步骤是：

根据系统调查的结果，进行分析，归纳出现行系统中的关键问题，做出问题表。

根据问题表，构造目标的层次结构，即目标树。在目标树中，最上层是总目标，以下各层是分目标或子目标，最下层是为实现目标而采取的具体措施，它是用来衡量目标是否切合实际的标准。

对目标树中的各项分目标进行分析。分析各项分目标之间的关系，确定解决目标冲突的方法，指出各项措施的考核指标。

将目标树按各层分目标在系统中所起的作用重新绘制。

② 确定信息系统工程规划的总体目标。信息系统工程的目标分析是指在组织总体目标分析的基础上，确立信息系统工程应在哪些方面发挥作用，以及如何发挥作用。通常，信息系统工程应该在下面几个方面发挥作用。

信息系统工程的辅助决策功能。在这方面，信息技术可以充分发挥迅速、准确的存储、检索、输出信息的能力和计算能力，还可以发挥信息系统工程的人－机系统优点，帮助决策者制定企业的长、中、短期决策。期望达到的目标有系统预测、计算机辅助决策、系统模拟与控制等。

信息系统工程的辅助管理功能。在这方面，信息技术可以实现部分或全部手工工作的自动化，如填制报表，生产经营数据统计，财务记账，人事档案的登录等。期望达到的目标是实现办

公自动化。

企业资源管理。随着计算机网络的普及，应用信息技术管理企业内外部资源，是信息系统工程的重要目标。按照经济学原理，企业的市场竞争力不在于拥有多少资源，而是资源是否得到优化配置。应用传统手工管理，由于其处理信息的效率低下，因而，很难做到优化配置。而信息系统工程，由于其强大的信息处理功能，可以对系统中各项经济活动的信息进行及时，甚至实时的处理，如生产经营、财务核算、辅助设计等，从而能够在瞬息万变的市场竞争中，抓住机会，战胜竞争对手。

（4）总体规划目标的使用信息系统工程总体规划目标制定出来以后，最重要的工作是正确、有效地使用它。要根据用户的需求和组织的现状，将目标进行分解和阐释。在总体目标的指导下，进一步规划系统的实施范围、功能结构、开发进度、投资规模、参加人员和组织保证等；在做好可行性分析的基础上，制定出实施方案，确定系统的总体结构和子系统的划分。以上工作完成以后，要组织有关专家对总体规划报告进行评估认证，根据认证意见修改或调整计划，必要时重新制订。

3．信息系统工程总体规划的范围

信息系统工程的总体规划包括，总体规划的层次、总体规划的任务、信息系统工程的功能范围、确定功能范围的步骤、系统总体结构分解、投资概算和总体规划的成果等内容。

（1）总体规划的层次信息系统工程的总体规划是一个涵盖面很广的概念。从层次上可以分为：信息战略规划、信息资源规划、信息系统工程建设规划和企业资源计划（ERP）。

这 4 个层次是一个从组织战略出发，由远及近、由粗到细，从抽象到具体的动态递进过程。而对于信息系统工程总体规划的理解存在不同意见，一种意见认为，信息系统工程总体规划就是信息系统工程建设规划；另一种意见认为，应将信息系统工程建设规划这一层细分成信息工程架构规划和信息工程方案规划。

① 信息战略规划。信息战略规划的基础是战略数据规划。战略数据规划这一概念是美国信息系统大师詹姆斯·马丁在 20 世纪 70 年代提出的。

② 信息资源规划。信息作为组织的一种资源，与物质、能源等其他资源一样，具有同等，甚至更重要的地位。因此，信息资源管理应该成为组织的一种新的管理职能。而做好信息资源管理的最重要的前提条件，就是做好信息资源规划。同时，信息资源规划也是信息系统工程规划中的重要内容。

信息资源规划是一个系统工程，包括：需求分析、集成化的管理信息系统的建设、信息资源管理基础标准的制定、业务概念设计模型的建立、信息资源网的构建等。

③ 信息系统工程建设规划。信息系统工程的建设需要比较长的时间周期，一般需要半年、一年，甚至几年时间。需要对工程进行科学的规划、组织、控制和进度安排，对各种方案进行分析、比较和决策等。对于一个组织的信息系统工程来说，为了便于管理和实施，可把整个工程看作一个或分成几个项目，对每一个项目都实施项目管理。

④ 企业资源规划。将在 13.4.2 节作详细论述。

（2）总体规划的任务

无论哪个层次的信息系统工程规划，其基本原理和过程都是一致的，大致都需要完成以下任务：

明确组织远景和使命。

确立组织发展战略和目标。

明晰组织业务及管理变革策略。

识别组织关键成功因素、分析关键性能指标、抽取信息需求。

建立总体信息工程框架。

提出可行性报告和总体规划方案。

信息系统工程的规划本质上可以对从完成业务战略出发到实现信息系统工程战略进行定位。

因此，在建设信息系统工程过程中，应当全面分析这个组织所处的环境，它的战略和目标，它的结构、标准作业过程甚至它的组织文化，并从中找到所要建设的信息系统工程与组织的关系及其应当起的作用。

（3）信息系统工程的功能范围根据已确定的系统目标和估算出的整个信息系统的信息量，考虑企业现有的客观条件，包括资金情况、设备条件、现场条件、技术水平、管理现状等，合理地确定系统的范围和功能。应注意，在不超越客观条件限制的条件下，使人、财、物得到充分利用，使系统的功能尽可能完善，保证系统目标的实现。

对于新建立的系统，可能要求现行的管理机构在组织上和功能上做某些调整和变动，以适应计算机的管理。在划分系统范围时，应按客观需要选择必要的系统结构和功能，不要受现行系统的限制。因为新系统在管理机制上，性能要优于现行系统，所以不能把现行体制搬到新的系统上，必要时，可以启动业务流程再造。

（4）确定功能范围的步骤确定系统的功能范围的步骤如下：

① 绘制出系统的总数据流程图。该图是系统分析阶段的各业务部门的数据流程图，综合绘制在一张图上。

② 根据系统方案的规定和用户的要求，结合现行系统的环境，确定系统的边界范围，并在总信息流程图上圈出。

③ 有关人员协商讨论。

④ 确定系统范围，并做出分析说明。

（5）系统总体结构分解为了将复杂的信息系统分解成便于理解和实现的部分，一般将信息系统分解为若干相对独立而又相互联系的子系统，即信息系统的主要系统。如果信息系统规模很大，为了方便信息系统的实现，还需将子系统再细分成若干个分系统。这是因为，首先，子系统间的相互关系仍过于复杂，分解之后，可以使这种关系更为明确、简单；其次，并不是在一个子系统中的所有过程都需要给予高优先级的支持，而应分出不同的优先级；最后，给定的子系统往往较大，难以一次实现，需要以分系统或几个分系统为单位来实现。

（6）投资概算

信息系统工程总体规划应当包括投资概算，它包括以下 4 个方面内容。

① 计算机系统软、硬件设备投资。将系统软、硬件配置表上列出的软、硬件设备按生产厂家提出的报价单，计算出它们的购置费，并汇总得出这部分的投资。在得出这部分的投资后，系统分析师应根据系统方案的实施步骤，考虑分期投资计划，列出与系统实施步骤相对应的购置计划。

② 系统开发费。系统开发费指从系统调研到系统全部实现这一过程中花费的研制时间和人力折合的费用，一般按人月或人周计算。系统分析师首先要确定所需的人员和能够投入的人员，再估计出开发周期，将这两个数据折合成“人月”或“人周”，最后得出开发费用。

③ 系统安装和维护费用。包括 3 方面内容：计算机软、硬件的安装和维护费用，这个费用一般根据计算机软、硬件生产厂家的报价估算；信息系统的安装和维护费用，系统分析师应该参考类似系统的情况，估计出人力和时间，再折合成费用，与所需物力折合的货币费用一起汇总成信息系统的安装和维护费用；基础设施的维修和改造费用，这里的基础设施包括机房、通信系统、供电系统和照明等。

④ 人员培训费。为使系统能够正常地运行和维护，需要对操作和维护人员进行定期培训，这种费用往往占很大的比重。

这 4 种费用也与系统的开发步骤有关，在系统方案报告中，也要与相应的开发步骤一起列出，并标明投资时间。

（7）总体规划的成果总体规划的成果是可行性分析报告和总体规划报告。

可行性分析也称为可行性研究。可行性研究已经成为新产品开发、工程投资等领域中决

策的重要手段。信息系统的开发同样也需要进行可行性研究，以便避免盲目投资，减少不必要的损失。

总体规划报告是信息系统工程总体规划的最终成果，经过一定的审批程序，以组织的名义发布实施，成为信息系统工程的指导性文件。

14.2.4 总体规划的方法论

制定信息系统工程总体规划需要有效的方法论支持，其方法多种多样，在此主要介绍较为著名的三种：业务系统规划法、关键成功因素法、战略目标集合转化法。

1．业务系统规划法

业务系统规划（Business Systems Planning，BSP）方法既是信息系统的重要规划方法，同时，也是信息系统工程总体战略规划的重要方法。

（1）BSP 的概念

BSP 方法是由 IBM 公司研制的指导企业信息系统规划的方法，虽然研制始于 20 世纪 70 年代，但其方法和思想至今仍有指导意义。它辅助企业信息系统规划，来满足其近期和长期的信息化需求。

实行 BSP 的前提是，在企业内有改善信息系统的要求，并且有为建设这一系统而建立总的战略的需要。因而 BSP 的基本功能是服务于信息系统建设的长期目标。

信息系统必须支持企业的战略目标。可以将 BSP 看成一个转化过程，即将企业的战略转化成信息系统的战略，因此，了解企业的战略就成了 BSP 重要内容之一。

① 信息系统的战略应当表达出企业的各个管理层次的需求。一般来说，不同层次的管理活动有着不同特点的信息化需求，因而有必要建立一个合理的框架，并据此来定义信息系统。首先，信息系统应强调对管理决策的支持。一般认为，在任一企业内同时存在着 3 个不同的层次，战略计划层：是决定组织目标、达到这些目标所需用的资源，以及获取、分配这些资源的策略的过程；管理控制层：通过这一过程，管理者确认资源的获取及组织的目标是否有效地使用了这些资源；操作控制层：保证有效率地完成具体的任务。

② 信息系统应该向整个企业提供一致的信息。信息的不一致性，源于“自下而上” 的系统数据处理。在企业的各部门，信息在形式上、定义上和时间上有差异。为了强调数据的一致性，有必要把数据作为一种资源来统一管理，它不应由一个局部的组织来控制，而应由一个中央部门来协调，使数据对企业有全面性的价值，被企业各单位共享。管理部门要负责

制定数据的一致性定义、技术实现，以及数据使用和系统安全性的策略和规程。

③ 信息系统应该适应组织机构和管理体制的改变。信息系统应具有适应性。在一个发展的企业中，数据处理系统决不要削弱或妨碍管理部门的应变能力，而应当有能力在企业的长期的组织机构和管理体制的变化中发展自己，而不受到大的冲击。为了实现上述目的，要有适当的关于信息系统的设计技术，BSP 采用了业务过程的概念，这种技术强调独立于组织机构和各种因素。对于任一类型的企业可以从逻辑上定义出一组过程，只要企业的产品或服务基本不变，则过程改变会极小。

④ 信息系统的战略规划，应当从总体信息系统结构中的子系统开始实现。支持整个企业需求的总信息系统一般规模都较大，因而有必要建立信息系统的长期目标和规划，从而形成了 BSP 对大型信息系统而言是“自上而下”的系统规划、“自下而上”的分步实现。应用 BSP，信息系统就能按部就班地以模块化方式进行建设，并照顾到企业的业务人员、资金情况和其他考虑。

总结起来，BSP 方法确立了信息系统建设的若干原则。方法本身是可以灵活运用的，即方法中的某些步骤和技巧可根据具体情况变化而做相应的调整，但它的基本原则不能违背，因为这些原则是 BSP 的灵魂。

（2）BSP 的目标

BSP 的目标主要是提供信息系统规划，用以支持企业短期的和长期的信息需要。其具体目标可归纳如下：

为管理者提供一种形式化的、客观的方法，明确建立信息系统的优先顺序，而不考虑部门的狭隘利益，并避免主观性。

为具有较长生命周期系统的建设和投资提供保障。由于系统是基于业务过程的，因而不因机构变化而失效。

为了以最高效率支持企业目标，BSP 提供数据处理和资源管理。

增加负责人的信心，使其坚信高效的信息系统能够被实施。

通过提供信息系统对用户需求的快速响应，从而改善信息系统管理部门和用户之间的关系。

应将数据作为一种企业资源加以确定，为使每个用户更有效地使用这些数据，要对这些数据进行统一规划、管理和控制。

由 BSP 所得到的规划不应当被看成是一成不变的，它只是在某一阶段对事物的最好认识。

BSP 方法的真正价值在于提供了下面的机会：一是创造一种环境和提出初步行动计划，使企业能依此对未来的系统和优先次序的改变做出反应，不致造成设计的重大失误。

二是定义信息系统的职能，并不断完善。

（3）BSP 方法实施步骤

① 确立项目。BSP 的经验说明，除非得到了最高领导者和某些最高管理部门参与的承诺，否则不要贸然开始实施 BSP。因为这项工作必须反映最高领导者关于企业的观点，其成果取决于管理部门能否向项目组提供企业的现状，以及他们对于企业的理解和对信息的需求。因此在一开始时就要对项目的范围和目标、应交付的成果取得一致意见，避免事后的分歧，这是至关重要的。

② 工作准备。在取得领导赞同以后，最重要的是选择项目组组长，要有一位企业领导用全部时间参加项目工作并指导项目组的活动。要确认参与研究的其他层次领导是否合适，并能正确地解释由他们所在部门得到的材料。

③ 主要活动。除了项目确立和准备工作外，BSP 还包含 11 项主要活动。

开始。BSP 首项活动是企业情况介绍，全体项目组成员要参加。重点内容有三方面：由管理部门负责人再次重申项目的目标，期望的成果和远景规划，以及与业务活动的关系；讨论有关企业的决策过程、组织职能、关键人物、存在问题、开发策略、敏感问题，以及用户对数据处理工作的支持等；由信息系统负责人和管理人员互相介绍本部门的情况，以便互相了解企业业务和信息化的历史和现状、目前的主要活动、计划中的变化和主要存在的问题。

定义业务过程。业务过程被定义为在企业资源管理中所需要的、逻辑上相关的一组决策活动。这些活动将作为安排同管理人员面谈、确定信息总体结构、分析问题、识别数据类，以及随后许多项目的基础。

定义数据类。业务过程被定义后，即要识别和定义由这些过程产生、控制和使用的数据。数据类指支持企业所必需的逻辑上相关的数据，即数据按逻辑相关性归成类，这样有助于数据库的长期开发。

分析现存系统支持能力。弄清目前的数据处理如何支持企业，进而对将来的行动提出建议。对目前存在的组织、业务过程、数据处理和数据文件进行分析，发现不足和冗余，明确责任，并进一步增进对业务过程的理解。

确定管理部门对系统的要求。BSP 方法必须考虑管理人员对系统的要求，并通过与高层管理人员的对话来确认项目组已做的工作，明确目标、问题、信息需求和它们的价值，并建立同最高管理部门的联系，争取他们的参与，使 BSP 项目组和管理部门间建立新的、更密切的关系。

提出判断和结论。通过与管理部门的会谈对所收集的材料作出确认、解释和补充。要对问题

进行分析并联系到业务过程，以便指导安排项目的优先顺序，并指明信息的改进将有助于解决问题。

定义信息总体结构。定义信息总体的结构是由对目前工作的情况转向对将来计划的综合的主要步骤。信息总体结构刻画出将来的信息系统和相应的数据，使系统和它们产生的数据结构化和条理化。由于此项工作是描绘将来信息系统的蓝图，因此全体项目组成员都要加以重视。确定总体结构中的优先顺序。项目组要确定系统和数据库开发优先顺序。对信息总体结构中的子系统的项目进行排列，然后根据确定的准则评定项目的重要性，从而确定开发顺序。

评价信息资源管理工作。为了使信息系统能高效率地开发、实施和运行，必须建立一个可控的环境。同时，信息系统的开发和运行过程必须加以优化，使其不断地随着技术和业务战略的变化而改变。

制定建议书和开发计划。开发计划是帮助管理部门对所建议的项目作出决策，这些项目由总体结构优先顺序和信息管理部门的建议来决定。开发计划要确定具体的资源、日程和其他项目间的关系，并需估计工作规模，以便管理部门进行调度。

工作成果报告。最后，项目组要向最高管理部门提交工作报告，等待最高管理部门的审查批准。

2．关键成功因素法

关键成功因素法（Critical Success Factors，CSF）是由麻省理工学院的一个研究小组开发的，并用于信息系统规划的一个有效方法。该方法能够帮助组织找到影响系统成功的关键因素，进行分析以确定组织的信息需求，从而为管理部门控制信息技术及其处理过程提供实施指南。

在每个企业中都存在着对企业成功起关键性作用的因素，称为关键成功因素。关键因素通常与那些能够确保组织生存和发展的方面相关。不同的业务活动，关键成功因素不同。即使同一类型的业务活动在不同时期的信息需求也可能不尽相同，同一信息系统的信息需求在不同时期也不相同。

希赛教育专家提示：关键成功因素法能够抓住主要矛盾，使目标的识别重点突出，为管理者提供一个结构化的方法，帮助企业确定其关键成功因素和信息需求。

（1）CSF 的确定关键成功因素的特征如下。

内部 CSF：针对机构的内部活动，如改善产品质量、提高工效等。

外部 CSF：与机构的对外活动有关，如，满足客户企业的进入标准、获得对方的信贷。

监控型 CSF：对现有业务流程等进行监控，如监测零件缺陷百分比。

建设型 CSF：适应组织未来变化的有关活动，如改善产品组合。

CSF 共分 4 层：行业的 CSF、组织的 CSF、部门的 CSF、管理者的 CSF，它们依次相互影响。可以通过内外渠道收集的数据按一定方法来验证 CSF，对于不易量化的 CSF 则多由管理者做出主观判断。若要用客观方法来量度，需相当高的创意，例如，使用德尔斐法或其他方法把不同人设想的关键因素综合起来。行业关键成功因素是在竞争中取胜的关键环节，可以通过层次分析法识别行业关键成功因素。

（2）CSF 实施步骤

CSF 法通过与管理者特别是高层管理者的交流，根据企业战略确定的企业目标，识别出与这些目标相关的关键成功因素及其关键性能指标。CSF 方法能够直观地引导高层管理者理清企业战略、信息化战略与业务流程之间的关系。

CSF 实施过程通常是：通过集成高层管理者的目标而确定成功因素，通过个人的成功因素的汇总，导出组织整体的决定性成功因素，然后据此建立能够提供与这些成功因素相关的信息系统。

第一步：了解组织的战略目标。

第二步：识别所有成功因素。可以通过与高级管理层进行交流，辨别其目标及由此产生的成功因素；也可以采用逐层分解的方法，引出影响系统战略目标的各种因素及影响这些因素的子因素。

第三步：确定关键成功因素。对所有成功因素进行评价，根据组织的现状和目标确定关键成功因素。

第四步：识别绩效指标和标准，以及测量绩效的数据。即给出每个成功因素的绩效指标和标准，以及用以衡量相应指标的数据。

关键成功因素与组织战略规划密切相关，组织战略描述组织期望的目标，关键成功因素则提供达到目标的关键路径和所需的测量标准。关键成功因素是为确保业务过程的成功需要完成最重要的工作，是业务过程的可观察、可测量的特征。它分布于组织的战略层、战术层、应用层及组织的各个方面，因此，需要对关键成功因素进行认真选择和度量，并对关键成功因素之间的关系进行动态调整。

（3）CSF 的优缺点

管理者必须面对环境的变化，在对环境分析的基础上认真考虑如何形成自己的信息需求，对于高层管理和开发 ESS、DSS 尤其适用，该方法要求高层管理就评价标准达成共识。该方法的缺点是：数据的汇总和数据分析过程比较随意，缺乏一种专门严格的方法将众多个人的

关键成功因素汇总成一个明确的整个组织的成功因素；由于个人和组织的成功因素往往并不一致，两者之间的界限容易被混淆，从而容易使组织的成功因素具有个人倾向性；由于环境和管理经常迅速变化，信息系统也必须做出相应调整，而用 CSF 法开发的系统可能无法适应变化了的环境；CSF 在应用于较低层的管理时，由于不容易找到相应目标的关键成功因子及其关键指标，效率可能会比较低。

3．战略目标集合转化法

战略目标集合转化法（Strategy Set Transformation，SST）将组织的战略看成一个“信息集合”，包括使命、目标、战略和其他战略变量，如管理水平、发展趋势以及重要的环境约束等。战略性系统规划就是把组织的战略集合转化为信息系统的战略集合，而后者由信息系统的系统目标、环境约束和战略规划组成。

该方法的步骤如下。第一步：识别和阐明组织的战略集合。首先考察组织是否有书面的战略规划，如果没有，就要去构造这种战略集合。其构造过程如下。可以采用以

（1）描绘出组织各类人员结构，如卖主、经理、雇员、供应商、顾客、贷款人、政府代理人、地区社团及竞争者等。

（2）识别每类人员的目标。

（3）对于每类人员识别其使命及战略。

第二步：将组织的战略集合转化为信息系统战略集合。这个转换过程包括对组织战略集合的每一个元素确定对应的信息系统战略元素。信息系统战略集合由系统目标、系统约束和系统设计原则组成。然后提炼出整个信息系统的结构，最后，选出一个较优的方案呈送管理层。

战略目标集合转化法所描述的是从组织的基本宗旨出发，得到对系统开发阶段的输入，其目的是产生一个与组织的战略和能力紧密相关的系统。但是由于不同组织的战略目标集的内容差别很大，所以转化过程还不能形成形式化的算法。

14.3 政府信息化与电子政务

政府信息化是传统政府向信息化政府的演变过程。具体地说，政府信息化就是应用现代信息技术、网络技术和通信技术，通过信息资源的开发和利用来集成管理和服务，从而提高政府的工作效率、决策质量、调控能力，并节约开支，改进政府的组织结构、业务流程和工作方式，全方位地向社会提供优质、规范、透明的管理和服务。

这个定义包含三个方面的内容：第一，政府信息化必须借助于信息技术和网络技术，离不开信息基础设施和软件产品；第二，政府信息化是一个系统工程，它不仅是与行政有关部门的信息化，还包括立法、司法部门及其他一些公共组织的信息化；第三，政府信息化并不是简单地将传统的政府管理事务原封不动地搬到互联网上，而是要对已有的组织结构和业务流程进行重组或再造。

这里需要说明的是，政府信息化的主要内容是电子政务。因此，在大多数情况下，电子政务可以作为政府信息化的同义语来使用。

14.3.1 我国政府信息化的历程和策略

20 世纪 90 年代以来，伴随着信息技术、特别是网络技术的飞速发展，信息化成为各国普遍关注的一个焦点。在国家信息化体系建设中，政府信息化又成为整个信息化中的关键。

1．我国政府信息化的发展历程

我国政府信息化最早起始于 20 世纪 80 年代末期“中国国家经济信息系统”的建设和运行。

当时，我国计划经济体制正在开始向市场经济体制转轨，社会发展对于经济信息的需求非常强烈。在这样的情况下，建设国家经济信息系统正是适应了国家和社会的多种需求。国家经济信息系统包括着重为国家宏观经济服务的主系统，以及各部门各行业的专业经济

信息系统在内的全国系统。同时，组建了国家经济信息中心作为国家经济信息系统的重要组成部分。国家经济信息中心是整个国家经济信息系统设计、规划、实施和技术协调的承担单位，是政府对全国经济信息事业的归口管理单位，它还负责经济信息政策的研究和经济信息系统技术规范和标准的制定。

国家经济信息系统不但为现今的政府信息化和电子政务提供了丰富的经验积累，也为企业信息系统的建设和运行起到了很好的示范作用。

到了 20 世纪 90 年代，随着信息技术的飞速发展和广泛应用，我国政府信息化也得到了长足的发展，其中最主要的成果有：

一是以“金”字头为代表的多项信息工程项目取得了突破性进展。从 1993 年起，我国开始实施金桥、金关、金卡“三金”工程。金桥工程直接为国家宏观经济调控和决策服务，通过建设政府的专用基础通信网，实现政府之间的相互连接，形成一个连接全国各省市区、400 多个城市，与几十个部委互联的专用网。金关工程主要是为提高外贸及相关领域的现代

化管理和服务水平而建立的信息网络系统。到 1999 年，已实现了银行、外汇管理机构及海关的计算机联网，在关税管理中发挥了重要作用。金卡工程是推动银行卡跨行业务的联营工作，现已取得了重要进展。在成功地实施“三金”工程的基础上，我国政府又实施了以金税工程为代表的一系列重大信息化工程。金税工程的建设完成，使我国税务管理提高到较高的层次，例如，该系统的增值税专用发票计算机稽核系统在增值税管理及减少直至杜绝偷漏增值税中发挥了很大作用。

二是政府上网工程初具规模。在“金”字系统工程取得重大进展的同时，从 1999 年起，又及时地推出包含“三金”在内的“十二金”工程。“十二金”工程是要重点推进的 12 个业务系统。这 12 个重点业务系统又可以分成三类：第一类是对加强监管、提高效率和推进公共服务起到核心作用的办公业务资源系统、宏观经济管理系统建设；第二类是增强政府收入能力、保证公共支出合理性的金税、金关、金财、金融监管（含金卡）、金审 5 个业务系统；第三类是保障社会秩序、为国民经济和社会发展打下坚实基础的金盾、金保、金农、金水、金质 5 个业务系统建设。

三是在“十二金”工程的带动下，各级政府都加强了电子政务的软件和硬件两方面的基础建设，建成了覆盖广泛的“两网、一站、四库”：“两网”是指政务内网和政务外网；政务内网主要是副省级以上政务部门的办公网，与副省级以下政务部门的办公网物理隔离。政务外网是政府的业务专网，主要运行政务部门面向社会的专业性服务业务和为业务需要在内网上运行的业务。两网之间物理隔离，政务外网与互联网之间逻辑隔离；“一站”，是政府门户网站；“四库”，即建立人口、法人单位、空间地理和自然资源、宏观经济 4 个基础数据库。

四是在中央的大力倡导下，各地在推动政府信息化方面健康发展，并在全国普遍实行了政府上网工程。到目前为止，全国绝大多数县级以上的政府都实现了电子政务。一些地区、部门在政府信息化方面已取得了显著成效。

有些发展较快的地区，如，上海、北京、广东等，还实施了信息港、数字城市等信息化工程项目。

深圳市率先在全国建成了深圳信息网。该网络充分利用邮电通信网、有线电视网、无线数据网、卫星网四大通信网络，构筑起全市政府部门统一的公共通信网络平台，成为涵盖市 5 套班子、6 个区及 88 个局委办，汇集几十个各类数据库的动态信息资源交汇体系。具体内容主要包括公共交换服务、虚拟专网、电子公务服务、市领导办公服务系统、应急指挥系统、多点电视会议服务系统、接入和信息发布系统、数字视频广播服务等。

北京市 2008 年成功地举办了奥运会，把奥运会办成了人文奥运、绿色奥运、科技奥运，

同时，也是一次信息化奥运。通过奥运，北京市，以及国家机关，至今已建成了高水平的公用信息平台和政务信息网络。具体内容有：建立了包括企业、人口、税收、统计、车辆、人才、市政等各种管理的一批数据库；全市 123 个机关、单位均在首都公用信息网平台上建立自己的网站；各级政府机关办公自动化程度明显提高等。逐步建成了体系完整、结构合理、高速宽带、互联互通的电子政务网络系统，全面开展网上交互式办公，从而基本实现政务信息化。

2．我国政府信息化的策略政府信息化的一个中心任务是实现由传统政务到电子政务的转变，这是一个牵一发而动全身的复杂问题。虽然近年来，计算机应用不断深入，互联网也在迅速普及，但总的来说，我国各级政府业务流程的信息化还有很长的路要走，各级政府信息系统建设也面临诸多问题。而要稳妥地解决这些问题，选择好政府信息化的策略十分重要。

（1）做好战略数据规划。信息工程方法是信息系统开发的有效方法。信息工程方法不仅适用于企业的信息化建设，也毫无疑问地适用于政府信息化建设。信息工程的基本原则之一就是信息系统建设应以数据为中心，面向数据，而不应该面向处理过程。因此，信息系统强调高层规划工作，即以战略数据规划为中心的总体规划和总体设计，有一套完整的 “自顶向下规划和自底向上设计相结合”的策略。在战略数据规划的指导下，搞好主题数据库建设。所谓主题数据库就是面向政府机构的业务主题的数据库。而应用开发应该在战略数据规划指导下，并且围绕主题数据库进行。

（2）面向主导业务流程。每一个政府部门，总有它自己的核心业务，由核心业务构成的业务流程是主导业务流程。电子政务应当面向主导业务流程，通过信息化优化、改造或重构业务流程。这样做具有事半功倍的效果。

（3）重视资源条件。政府信息化是一个系统工程，必然受到内外部环境的制约，也要受到政府机构本身资源状况的限制。因此，政府信息化首先要考虑的问题就是政府本身的财力、物力和人力的状况；同时，也要考虑政府机构内部工作人员的接受程度，以及外部相关单位或部门的认可程度；另外还要考虑系统运行以后的经济效益和社会效益。例如，从经济效益方面考虑，税收管理、财务管理、资源和计划管理、市场和投资管理等项目都应优先建设，我国的“金关”工程和“金税”工程的成功就是典型例子。从社会效益方面考虑，面向公民的各种服务系统、警察与公安系统、医疗与保健系统、环境保护和环境信息系统等都应优先建设。

（4）以人为本。政府信息化能否成功，最终取决于人及其素质，要看机构中是否有一支高水平的人才队伍。这支队伍的成员要熟练掌握信息技术，同时还要有娴熟的业务能力，

并使二者很好地结合。这就要求信息技术人员要能深刻理解政府业务，业务人员要学习信息技术。而对于领导来说，更应带头学习，建立起信息管理的观念，形成决策办事讲科学、使用信息技术工作的习惯。因为领导的行为和观念对机构成员来说是巨大的、无形的力量，它不仅对从事信息管理工作的专业人员是巨大的支持，而且对全体成员起着示范和带头作用。

信息化以人为本，要求不管是领导成员、技术人员还是业务人员都要在信息化过程中不断学习，学习信息技术，学习新的管理理论，转变观念以适应信息化进程，在政府信息化过程中，培养新的人才。

希赛教育专家提示：在政府信息化的过程中，应当特别注意培养一种“信息技术专家＋管理业务专家＋优秀领导者”的人才。这样的人才能够在政府信息化中起到主导和先锋作用。这样的人才就是系统分析师。

（5）设立 CIO。在发达国家，盛行着 CIO（Chief Information Officer，首席信息官员）职位。在我国，目前许多大型企业都在最高管理层中设立了 CIO 职位。在企业里，CIO 是相当于副总裁的高级职位。许多企业的实践证明，设立 CIO，对于企业信息化起到了很大的促进作用。CIO 起源并发展于美国。现在，美国的大型企业几乎都设立了 CIO，许多中小企业也设立了自己的 CIO。其实，在美国，最早的 CIO 并不是在企业中出现的，而是出现在政府。1980 年以后，为了从组织机构上保证和加强联邦政府各部门的信息资源管理活动，美国政府要求各部门都要设立 CIO 这一职位，并委派副部长或部长助理级的官员担任此职，从较高层次上全面负责本部门信息资源的开发利用，这就是最初的 CIO，人们习惯把 CIO 称作“政府 IT 沙皇”。虽然我国的情况与美国有很大的不同，但是，在政府部门设立 CIO 职位的必要性应当是一致的。

（6）加强规范化和标准化。我国大大小小的政府机构数以万计，如此巨大的电子政务建设规模，如果采用个体经济的办法任由部门各自开发自己的系统，不仅浪费大量的资源和时间，而且由于缺乏标准和规范，政府之间、政府部门之间的各种系统势必难以兼容，信息资源难以共享。

实际上，电子政务中包含许多的标准“零部件”，如人事、财务、计划、公文、档案、日程安排、国有资产、器材、图书资料、考勤管理及政府网站等，不下数十种。如果这些“零部件”都能做到规范化和标准化，不仅可以节约大量的资源，而且可以形成和支持一个相当大的软件产业。“零部件”规范化和标准化的关键则在于政府业务过程的规范化和数据模型的标准化。

从国外的经验来看，电子政务的标准化和规范化并不一定都需要通过行政命令来实现，有些

可以通过技术政策来引导和推进，有些则可以依赖于市场的作用，让市场占有份额大的产品成为事实上的标准或规范。建立政府与企业某种形式的伙伴关系有可能使双方都从中受益。

（7）充分利用社会资源。其实，政府信息系统的建设并不一定非要政府投资不可。因为政府的职责是完成法律赋予的职能，不是信息系统的开发。因此，在电子政务的发展中，政府的角色是准确地提出对信息系统的要求，实现对信息化的有效管理。系统开发的任务应该留给企业去做。如果每一个政府部门都建立一支队伍去搞部门的系统开发，成果不能实现商品化，则其成本必然较高，同时，部门自有的开发队伍由于其经验和技术的限制， 以及其视野的局限性，决定了这样开发出来的系统，其开放性、实用性都可能大打折扣，更为有问题的是，为内部人员非法修改系统和犯罪提供了机会。

14.3.2 电子政务的内容

电子政务实质上是对现有的、工业时代形成的政府形态的一种改造，即利用信息技术和其他相关技术，来构造更适合信息时代的政府组织结构和运行方式。因此，电子政务在概念、内容和技术形式上都区别于现有政务。

电子政务实质上是对现有的、工业时代形成的政府形态的一种改造，即利用信息技术和其他相关技术，来构造更适合信息时代的政府组织结构和运行方式。现有的政府组织形态是工业革命的产物，与工业化的行政管理的需求和技术经济环境相适应，已经存在了两百年以上。随着网络时代和网络经济的来临，管理正由传统的金字塔模式走向网络模式。政府的组织形态也必然由金字塔式的垂直结构向网状结构转变，从而减少管理的层次，以各种形式通过网络与企业和公民建立直接的联系。因此，电子政务的发展过程实质上是对原有的政府形态进行信息化改造的过程，通过不断的摸索和实践，最终构造出一个与信息时代相适应的政府形态。

在社会中，与电子政务相关的行为主体主要有三个，即：政府、企（事）业单位及公民。因此，政府的业务活动也主要围绕着这三个行为主体展开，即包括政府与政府之间的互动；政府与企（事）业单位，尤其是与企业的互动；政府与公民的互动。在信息化的社会中，这三个行为主体在数字世界的映射，构成了电子政务、电子商务和电子社区三个信息化的主要领域。电子商务在经历了一个发展热潮之后，目前正在向一个新的、更扎实的阶段发展；电子政务则是当前全球关注的热点，正在形成一个发展的热潮。

政府与政府，政府与企（事）业，以及政府与公民之间的互动构成了下面 5 个不同却

又相互关联的领域。

（1）政府与政府（Government To Government）。政府与政府之间的互动包括首脑机关与中央和地方政府组成部门之间的互动；中央政府与各级地方政府之间、政府的各个部门之间、政府与公务员和其他政府工作人员之间的互动。这个领域涉及的主要是政府内部的政务活动，包括国家和地方基础信息的采集、处理和利用，如人口信息、地理信息、资源信息等；政府之间各种业务流程所需要采集和处理的信息，如计划管理、经济管理、社会经济统计、公安、国防、国家安全等；政府之间的通信系统，包括各种紧急情况的通报、处理和通信系统；政府内部的各种管理信息系统，如财务管理、人事管理、公文管理、资产管理、档案管理等；以及各级政府的决策支持系统和执行信息系统。

（2）政府对企业（Government To Business）。政府面向企业的活动主要包括政府向企（事）业单位发布的各种方针、政策、法规、行政规定，即企（事）业单位从事合法业务活动的环境，包括产业政策、进出口、注册、纳税、工资、劳保、社保等各种规定；政府向企（事）业单位颁发的各种营业执照、许可证、合格证、质量认证等。“政府对企业”的活动实质上是政府向企业提供的各种公共服务，如构造一个良好的投资和市场环境，维护公平的市场竞争秩序，协助企业，特别是中小企业的发展，帮助企业进入国际市场和加入国际竞争，以及提供各种各样的政府信息的服务等。

（3）政府对公民。政府对公民的活动实际上是政府面向公民所提供的服务。政府对公民的服务首先是信息服务，让公民知道政府的规定是什么，办事程序是什么，主管部门在哪里，各种关于社区公安和水、火、天灾等与公共安全有关的信息，以及户口、各种证件和牌照等的管理等政府面向公民提供的各种服务。政府对公民的服务还包括各公共部门如学校、医院、图书馆、公园等面向公民的服务。

（4）企业对政府。企业面向政府的活动包括企业应向政府缴纳的各种税款，按政府要求应该填报的各种统计信息和报表，参加政府各项工程的竞、投标，向政府供应各种商品和服务，以及就政府如何创造良好的投资和经营环境，如何帮助企业发展等提出企业的意见和希望，反映企业在经营活动中遇到的困难，提出可供政府采纳的建议，向政府申请可能提供的援助。

（5）公民对政府。公民对政府的活动除了包括个人应向政府缴纳的各种税款和费用，按政府要求应该填报的各种信息和表格，以及缴纳各种罚款等外，更重要的是开辟公民参政、议政的渠道，使政府的各项工作不断得以改进和完善。政府需要利用这个渠道来了解民意，征求群众意见，以便更好地为人民服务。此外，报警服务（盗贼、医疗、急救、火警等），

即在紧急情况下公民需要向政府报告并要求政府提供的服务，也属于这个范围。

（6）政府对公务员。G2E 电子政务是政府机构通过网络技术实现内部电子化管理的重要形式，也是 G2G、G2B 和 G2C 电子政务模式的基础。G2E 电子政务主要是利用 Intranet 建立起有效的行政办公和员工管理体系，为提高政府工作效率和公务员管理水平服务。

当前，世界各国电子政务的发展就是围绕着上述 6 个方面展开的，其目标除了不断地改善政府、企业与公民三个行为主体之间的互动，使其更有效、更友好、更精简、更透明和更有效率之外，更强调在电子政务的发展过程中对原有的政府结构及政府业务活动组织的方式和方法等进行重要的、根本的改造，从而最终构造出一个信息时代的政府形态。

14.3.3 电子政务建设的过程模式和技术模式

电子政务建设的过程模式是电子政务目标的实现过程，而电子政务的技术模式则是其实现的技术手段。

1．电子政务建设的过程模式

电子政务发展的基本条件是要有明确的目标，同时，要落实相应的实施部门和所需的资源。其中，特别重要的是明确地定义电子政务的目标，以及通过做哪些事情或完成哪些项目来达到这些目标，这就是电子政务建设的过程模式。

（1）以用户为中心。在电子政务实现的早期阶段，各个政府部门的网站都是按照政府的组织结构来设计的。经过一段时间的实践发现，要真正为用户服务好，必须以用户为中心，按照用户的意向，来设计政府的网站。

（2）引进“客户关系管理”技术。“客户关系管理”是近年来在企业界非常流行的一种信息技术。它通过与客户的互动和信息交流，来掌握客户消费习惯和行为方式，以达到留住老客户、争取更多新客户，扩大市场占有率的目的。现在，这种技术也开始被引入电子政务之中，帮助政府改善与其“客户”——企业和公民的关系。因为，政府比任何企业或单位都有更多的“客户”，将“客户关系管理”技术引入电子政务之中，可以帮助政府更好地为有特殊需要的“客户”服务，从而建立新的、更好的政府与企业、政府与公民之间的关系。

（3）政府门户。政府门户网站已经成为电子政务发展较高阶段的一种基本形式，即通过一个门户网站可以进入政府的所有部门，或者可以进入任何一个由政府向用户所提供的服务项目。对于那些需要几个政府部门同时介入才能完成的事务处理，这种门户网站对用户来说极为方便。这种通过门户网站形成的用户与政府的互动，对于用户来说，政府的纵横交错

的结构是透明的。用户只需要在网上完成他所需要的与政府互动的事务处理，根本不需要知道在这件事情完成的过程中，他与哪些政府部门、哪些政府官员打过交道。

2．电子政务的技术模式

电子政务的技术模式由网络管理模式、信息资源管理模式和应用开发模式，以及网络安全、标准化等构成。电子政务通过一定的技术模式将现有的和即将建设的各个政府网络和应用系统联结起来，统一标准和规范，做到互联互通，成为一个统一的政府信息化平台。

（1）网络管理模式电子政务在网络管理上分为政府专网和通用网络两部分，包括专用网络、内部网络和外部网络。

专用网络。指政府部门之间的网络，因为对于机密信息的交换，需要在与外部网络进行物理隔离的专用网络上传输，以保证机密信息的绝对安全性。

内部网络。政府内部的办公网络，以局域网为主，有时需要有广域网，用于政府内部和政府部门之间一般的信息交换。内部网络具有传统数据网络的性能优点和共享数据网络结构的优点，同时，还能够提供远程访问，以及外部网络和内部网络的连接。

外部网络。对于为公众提供的信息及其他可公开的信息，可以利用政府网站等形式发布到 Internet 网上。

（2）信息资源管理模式

信息资源是电子政务的处理对象，也是电子政务的基础。采用何种模式进行信息资源管理，关系到电子政务的成败。政府部门的信息从内容上大致可以分成两类：一类是来自公文系统的文档型信息，另一类是来自数据处理系统的结构化信息。电子政务可以选用的信息资源管理模式有多种，目前主要有两种，即元数据管理模式和 XML 数据管理模式。

元数据管理模式。该模式可以为不同部门、不同级别的机构提供统一的数据管理和交换模式，为跨部门、跨行业的信息资源整合提供技术基础。元数据管理模式一般采用分布式的数据存储形式，通过元数据实现各级部门之间的信息检索和内容调用。元数据管理模式采用分类编目管理结构，对电子政务系统中的各类信息进行分类组织，从而达到知识管理和决策支持的目标。

XML 数据管理模式。在数据交换和共享的层面上，基于 XML 数据管理模式，建立统一的信息技术平台，实现不同系统的互联。它覆盖了信息处理的从数据采集、处理和传输，到信息管理、分析和共享的整个流程，将传统的管理信息系统提升到具有数据分析和共享功能的系统中，从数据中挖掘和提炼知识，为决策提供有力的支持。

其他的数据管理模式还有基于 Web 的数据库和数据仓库等。

（3）应用开发模式

应用开发是电子政务最关键的一环，也是体现电子政务价值的所在。电子政务的应用开发模式主要有：

政府与公务员（Government To Employee，G2E）。利用 Intranet 建立有效的行政办公体系，为提高政府工作效率服务。内容包括：电子公文、电子邮寄、电子规划管理、电子人事管理等。

政府对经济活动（Government ToBusiness，G2B）。利用互联网等网络手段为经济活动提供信息化支持，包括：电子商务、电子税务、电子金融、电子海关等基础设施服务。

政府部门与政府部门（Government To Government，G2G）。政府间的信息交换有助于不同部门间的协同办公，可以解决信息孤岛的问题，使目前很难实现的信息共享、交换、协同工作等问题得以较好的解决。

政府对公民服务（Government ToCitizen，G2C）。利用公共网络为公众提供广泛的信息服务，包括卫生、教育、法律、税务、金融等一系列的信息服务。

（4）电子政务的安全体系

电子政务系统中的重要组成部分就是安全体系。电子政务的安全体系包括物理安全、网络安全、信息安全及安全管理等方面。

在实施过程中，要在政府内外网之间实行物理隔离，在部门内网和政府专网之间实施逻辑隔离。内外网之间信息交流通过倒磁盘的手工方式、半自动方式或全自动隔离服务器的方式进行。同时，系统必须应用 CA 认证，加密传输，防火墙技术、VPN，漏洞检测与在线黑客监测预警，实时审计，网络防病毒，自动备份恢复等一系列安全技术。

信息安全系统以 PKI 技术为基础，围绕数字证书应用，为各种业务应用提供信息的真实性、完整性、机密性和不可否认保证，并在业务系统中建立有效的信任管理机制、授权控制机制和严密的责任机制。信息安全与应用紧密相关，可分为信息安全基础设施和信息安全应用产品两类。

信息安全基础设施产品为各种应用提供通用的安全服务，通过建立通用的安全接口来实现安全服务。主要包括：PKI、PMI、密钥管理。PKI 以公开密钥技术为基础，以数据机密性、完整性、身份认证和行为的不可否认为安全目的。信任服务体系提供基本 PKI 数字证书认证机制的实体身份鉴别服务，从而建立全系统范围内一致的信任基准，为实施电子政务提供支持。密钥管理基础设施（Key Management Infrastructure，KMI）提供统一的密钥管理服务，涉及密钥生成服务器、密钥数据库服务器和密钥服务管理器等组成部分。授权管理基础设施

（Grant Management Infrastructure，GMI）主要负责向应用系统提供与应用相关的授权服务管理，授权管理以资源为核心，将对资源的访问控制权统一交由资源的所有者进行访问控制。考虑到不同行业纵向业务系统中的授权管理体系和不同行政级别的横向行政管理系统中的授权管理体系并存，因而也存在一个信任链互连问题。通常，GMI 与 PKI 结合，有效提高授权控制能力。

管理性和技术性的安全措施是相辅相成的，在对技术性措施进行设计的同时，必须考虑安全管理措施。因为诸多的不安全因素恰恰反映在组织管理和人员使用方面，而这又是计算机网络安全所必须考虑的基本问题，所以应在整个安全体系设计时倍加重视。该类产品主要是帮助进行安全管理，如安全策略的制定、系统安全运行状况调查、安全事件的跟踪与处理、安全审计和证据采集、使用等。

（5）电子政务的标准化电子政务是一项系统工程，是国家信息化建设的重要领域，而标准化是电子政务重要

的支撑手段。国家信息化领导小组发布的《关于我国电子政务建设指导意见》规定了电子政务建设的指导思想和原则：统一规划，加强领导；需求主导，突出重点；整合资源，拉动产业；统一标准，保障安全。在阐释“统一标准，保障安全”原则时指出，“加快制定统一的电子政务标准规范，大力推进统一标准的贯彻落实。要正确处理发展与安全的关系，综合平衡成本和效益，一手抓电子政务建设，一手抓网络与信息安全，制定并完善电子政务网络与信息安全保障体系”。

为了加强电子政务标准化工作，国务院信息化工作办公室和国家标准化管理委员会成立了“国家电子政务标准总体组”（简称总体组）。总体组适时编写了《国家电子政务标准化指南》，并组织有关单位起草制定了六项电子政务相关标准，以指导我国电子政务的建设，促进其健康发展。

《国家电子政务标准化指南》共分为以下六个部分。

第一部分：总则。概括描述电子政务标准体系及标准化的机制。

第二部分：工程管理。概括描述电子政务工程管理须遵循或参考的技术要求、标准和管理规定。

第三部分：网络建设。概括描述网络建设须遵循或参考的技术要求、标准和管理规定。

第四部分：信息共享。概括描述信息共享须遵循或参考的技术要求、标准和管理规定。

第五部分：支撑技术。概括描述支撑技术须遵循或参考的技术要求、标准和管理规定。

第六部分：信息安全。概括描述保障信息安全须遵循或参考的技术要求、标准和管理规

定。

六项电子政务标准分别如下：

① 基于 XML 电子公文格式规范第一部分：总则，第二部分：公文体；

② XML 在电子政务中的应用指南；

③ 电子政务业务流程设计方法通用规范；

④ 信息化工程监理规范；

⑤ 电子政务数据元第一部分：设计和管理规范；

⑥ 电子政务主题词表编制规则。

可以相信，随着电子政务的深入发展，电子政务的标准化体系必将得到进一步的完善，从而为政府信息化做出更大贡献

14.4 企业信息化与电子商务

本节首先介绍企业信息化的概念、目的、规划、方法，然后再介绍 ERP、CRM、PDM（Product Data Management，产品数据管理）、企业门户、EAI、SCM 等内容，最后介绍电子商务的类型和标准。

14.4.1 企业信息化概述

企业信息化是指企业以业务流程的优化和重构为基础，在一定的深度和广度上利用计算机技术、网络技术和数据库技术，控制和集成化管理企业生产经营活动中的各种信息，实现企业内外部信息的共享和有效利用，以提高企业的经济效益和市场竞争力。

如果从动态的角度来看，企业信息化就是企业应用信息技术及产品的过程，或者更确切地说，企业信息化是信息技术由局部到全局，由战术层次到战略层次向企业全面渗透，运用于流程管理、支持企业经营管理的过程。这个过程表明，信息技术在企业的应用，在空间上是一个由无到有、由点到面、由浅到深、由低级到高级的过程；在时间上具有阶段性和渐进性，起初是战术阶段，经过逐步深化，发展到战略阶段；信息化的核心和本质是企业运用信息技术，进行隐含知识的挖掘和编码化，进行业务流程的管理。企业信息化的实施，一般来说，可以沿两个方向进行，一是自上而下，与企业的制度创新、组织创新和管理创新结合；二是自下而上，以作为企业主体的业务人员的直接受益和使用水平逐步提高为基础。

1．企业信息化的目的一般意义而言，企业信息化的目的就是要建立一个整体上相当于

人的神经系统的数字神经系统。这种数字神经系统，使得企业具有平稳和有效的运作能力，对紧急情况和商机做出快速反应，为企业内外部用户提供有价值的信息，以提高企业的核心竞争力。企业要应对全球化市场竞争的挑战，特别是大型企业要实现跨地区、跨行业、跨所有制、跨国经营的战略目标，要实施技术创新战略、管理创新战略和市场开拓战略，要将企业工作重点转向技术创新、管理创新和制度创新的方向上来，信息化是必然选择和必要手段。企业信息化涉及对企业管理理念的创新，管理流程的优化，管理团队的重组和管理手段的革新。

2．企业信息化的规划企业信息化一定要建立在企业战略规划基础之上，以企业战略规划为基础建立的企业管理模式是建立企业战略数据模型的依据。企业信息化就是技术和业务的融合。这个“融合”并不是简单地利用信息系统去对手工的作业流程进行自动化，而是需要从三个层面来实现：

首先，企业战略的层面。在规划中必须对企业目前的业务策略和未来的发展方向作深入分析。通过分析，确定企业的战略对企业内外部供应链的相应管理模式，从中找出实现战略目标的关键要素，分析这些要素与信息技术之间的潜在关系，从而确定信息技术应用的驱动因素，达到战略上的融合。

其次，业务运作层面。针对企业所确定的业务战略，通过分析获得实现这些目标的关键业务驱动力和实现这些目标的关键流程。这些关键流程的分析和确定要根据他们对企业价值产生过程中的贡献程度来确定。关键的业务需求是从对那些关键的业务流程的分析中获得的，它们将决定未来系统的主要功能。这一环节非常重要，因为信息系统如果能够与这些直接创造价值的关键业务流程相融合，这对信息化投资回报的贡献是非常巨大的，也是信息化建设成败的一个衡量指标。

再次，管理运作层面。虽然这一层面从价值链的角度上来说，属于辅助流程，但它对企业日常管理的科学性、高效性是非常重要的。另外，在企业战略层面的分析中，可以获得适应企业未来业务发展的管理模式，这个模式的实现是离不开信息技术的支撑的。所以，在管理运作层面的规划上，除了提出应用功能的需求外，还必须给出相应的信息技术体系，这些将确保管理模式和组织架构适应信息化的需要。

企业信息化规划的重要性是不言而喻的，但是，要防止一种倾向，就是把信息化规划片面地理解为信息技术规划，这样的观念是有害的。

企业战略数据模型分为数据库模型和数据仓库模型，数据库模型用来描述日常事务处理中的数据及其关系；数据仓库模型则描述企业高层管理决策者所需信息及其关系。在企业信

息化过程中，数据库模型是基础，一个好的数据库模型应该客观地反映企业生产经营的内在联系。数据库是办公自动化、计算机辅助管理系统、开发与设计自动化、生产过程自动化、Intranet 的基础和环境。

希赛教育专家提示：信息技术和网络技术都在飞速发展，企业信息化是多种类、多层次信息系统建设、集成和应用的过程，因而不是一蹴而就的事情，而是需要结合企业的实际，全面规划，分步实施。

3．企业信息化的方法企业信息化建设是一项系统工程，而不是单元技术的改造，它要涉及企业的方方面面，也就是会涉及企业所处的“生态系统”，个别单位或部分业务的信息化并不能代表整个企业的信息化。企业信息化建设与其说是一场技术变革，还不如说是对企业的经营管理和业务流程的一次革命，它借助于先进的信息技术和网络技术进行价值链重构。同时，企业信息化是一个不断发展、变化的过程，它没有终点，至少目前还看不到终点。企业信息化随着管理理念、信息技术和网络技术的发展而发展，是一个螺旋式上升的过程。而在这个过程中，企业使用什么方法实现信息化，就成为一个事关成败的大问题。

这里需要指出的是，企业信息化方法并不同于信息系统建设方法，这是因为，信息系统建设方法是一个具体的信息项目建设的方法，而企业信息化方法是整个企业实现信息化的方法，因此，企业信息化方法要比信息系统建设方法层次高、涉及面更广。

通过 30 年的发展，人们已经总结出了许多非常实用的企业信息化方法，并且还在探索新的方法。这里只简单介绍几种常用的企业信息化方法。

（1）业务流程重构方法。在 20 世纪 90 年代初，美国学者哈默和钱佩在其著作《企业重构》一书中系统地提出了企业业务流程重构的思想，对美国，以至于世界范围内的企业界产生了很大的影响，一时，企业业务流程重构形成了浪潮。企业业务流程重构的中心思想是，在信息技术和网络技术迅猛发展的时代，企业必须重新审视企业的生产经营过程，利用信息技术和网络技术，对企业的组织结构和工作方法进行“彻底的、根本性的”重新设计，以适应当今市场发展和信息社会的需求。现在，业务流程重构已经成为企业信息化的重要方法。特别是长期受计划经济体制影响的企业，采用业务流程重构方法来实现企业信息化更有现实意义。

（2）核心业务应用方法。任何一个企业，要想在市场竞争的环境中生存发展，都必须有自己的核心业务，否则，必然会被市场所淘汰。当然，不同的企业，其核心业务是不同的。例如，一个石油生产企业，原油的勘探开发生产就是它的核心业务。围绕核心业务应用计算机技术和网络技术是很多企业信息化成功的秘诀。

（3）信息系统建设方法。对大多数企业来说，建设信息系统都是企业信息化的重点和关键。因此，信息系统建设成了最具普遍意义的企业信息化方法。

（4）主题数据库方法。主题数据库就是面向企业业务主题的数据库，也就是面向企业的核心业务的数据库。有些企业，特别是大型企业，其业务数量浩繁，流程错综复杂。在这样的企业里，建设覆盖整个企业的信息系统往往很难成功，但是，各个部门的局部开发和应用又有很大弊端，会造成系统分割严重，形成许多信息孤岛，造成大量的无效或低效投资。在这样的企业里，应用主题数据库方法推进企业信息化无疑是一个投入少、效益好的方法。例如，对于一个油田企业来说，勘探开发无疑是它的核心业务，有一个大型油田企业，在十几年前，就投入巨大的人力、物力和财力开发“勘探开发数据库”。经过十几年的努力，目前，该数据库字符型数据已达 G 字节级，机器自动采集的数据已达 P 字节级，对企业生产经营发挥了巨大的作用，取得了巨大的经济效益。

（5）资源管理方法。资源是企业生存发展的根本保证，一个企业如果离开了资源，那它是无法生存的。而资源又包括很多类型，例如，有人力资源、物力资源等；同时，资源又可分为内部资源和外部资源。管理好企业的资源是企业管理的永恒主题。计算机技术和网络技术的应用为企业资源管理提供了强大的能力。因此，资源管理方法也就成了企业信息化的重要方法。目前，流行的企业信息化的资源管理方法有很多，最常见的有 ERP、 SCM 等。

（6）人力资本投资方法。人力资本的概念是经济学理论发展的产物。人力资本与人力资源的主要区别是人力资本理论把一部分企业的优秀员工看作一种资本，能够取得投资收益。人力资本投资方法特别适用于那些依靠智力和知识而生存的企业，例如，各种咨询服务、软件开发等企业。

14.4.2 企业资源规划

ERP 是一种融合了企业最佳实践和先进信息技术的新型管理工具，它在企业信息化中具有示范性和标志性的作用。ERP 是一种融合了企业最佳实践和先进信息技术的新型管理工具。它扩充了 MIS、MRPⅡ（Manufacturing Resources Planning，制造资源计划）的管理范围，将供应商和企业内部的采购、生产、销售及客户紧密联系起来，可对供应链上的所有环节进行有效管理，实现对企业的动态控制和各种资源的集成和优化，提升基础管理水平，追求企业资源的合理高效利用。ERP 是由美国 Gartner Group 于 20 世纪 90 年代初首先提出的。ERP 实质上仍然以 MRPⅡ为核心，但 ERP 至少在两方面实现了拓展，一是将资源的概念扩

大，不再局限于企业内部的资源，而是扩大到整个供应链条上的资源，将供应链上的供应商等外部资源也作为可控对象集成进来；二是把时间也作为资源计划的最关键的一部分纳入控制范畴，这使得 DSS 被看作是 ERP 不可缺少的一部分，将 ERP 的功能扩展到企业经营管理中的半结构化和非结构化决策问题。因此，ERP 被认为是顾客驱动的、基于时间的、面向整个供应链管理的制造资源计划。

ERP 的概念对应于管理界、信息界、企业界不同的表述要求，ERP 分别有着它特定的内涵和外延。对于企业来说，要理解 ERP，首先要明确什么是“企业资源”。简单地说，“企业资源”是指支持企业业务运作和战略运作的事物，既包括人们常说的人、财、物，也包括人们没有特别关注的信息资源；同时，不仅包括企业的内部资源，还包括企业的各种外部资源。因此，ERP 就是一个有效地组织、计划和实施企业的内外部资源的管理系统，它依靠 IT 的技术和手段以保证其信息的集成性、实时性和统一性。

1．ERP 的结构

ERP 是一个层次结构，可分为三个层次，即管理思想、软件产品、管理系统。

（1）ERP 的管理思想

ERP 最初是一种基于企业内部“供应链”的管理思想，是在 MRPⅡ的基础上扩展了管理范围，给出了新的结构。它的基本思想是将企业的业务流程看作一个紧密连接的供应链，将企业内部划分成几个相互协同作业的支持子系统，如财务、市场营销、生产制造、质量控制、服务维护、工程技术等。最早采用这种管理方式的是制造业，当时主要考虑的是企业的库存物料管理，于是产生了 MRP 系统，同时企业的其他业务部门也都各自建立了信息管理系统，诸如会计部门的计算机账务处理系统、人事部门的人事档案管理系统等，而这些系统早期都相互独立、彼此之间缺少关联，从而形成信息孤岛，不但没有发挥 IT 功能和作用，反而造成了企业管理的管理环节和管理部门的重复和不协调。

在这种情况之下，MRPⅡ应运而生。它围绕着“在正确的时间制造和销售正确的产品”这样一个中心目标，将企业的内外部资源进行集中管理。在一定意义上说，ERP 可以说是 MRPⅡ的一个扩展。第一，它将系统的管理核心从“在正确的时间制造和销售正确的产品”转移到了“在最佳的时间和地点，获得企业的最大增值”；第二，基于管理核心的转移，其管理范围和领域也从制造业扩展到了其他行业和企业；第三，在功能和业务集成性方面，都有了很大加强，特别是商业智能的引入使得以往简单的事务处理系统变成了真正智能化的管理控制系统。

（2）软件产品

随着应用的深入，作为 ERP 的载体——软件产品，也在向更高的层次发展，已经经历了三个阶段，最初，ERP 就是一个软件开发项目。这时的软件产品一般来说，费用高，耗时长，而且项目可控性很差，出现了所谓 ERP 成功率低的结果。后来，经过发展，ERP 产品发展成模块化，大大地提高了软件开发效率，但是，由于是产品导向，出现了削足适履的现象，因而，这时 ERP 的成功率还是不算高。现在，ERP 产品则发展到比较高的阶段。大多数 ERP 产品供应商都在模块化的基础上，把软件产品和软件服务进行集成，实现软件产品的技术先进性和个性化设计，为用户提供一体化的解决方案。

同时，先进的 IT 技术也为 ERP 提供了技术支持手段，如网络技术、Internet/Intranet 技术、条码技术、电子商务技术、数据仓库技术、远程通信技术等，使得各企业在业务往来和数据传递过程中实现电子方式连接；在管理技术上，在从内部到外部各个环节上，ERP 为企业提供了有效的管理工具。由于 ERP 为企业提供更多更好的功能，帮助企业实现管理信息化和现代化，因而使得企业市场竞争力和综合实力得到提高。

（3）管理系统

毫无疑问，管理系统是 ERP 的基础和依托。一个企业，它要根据市场预测制定全面的预算和计划，因此，企业必须实施动态管理。而一个动态的管理模式需要一个运行系统，而 ERP 正是这样一个系统。

ERP 是一个集成的信息系统，ERP 承诺建立跨越企业各个部门、各种生产要素和环境的单一应用原则下处理所有的事务，即意味着集成。这种集成应该包括人力资源、财务、销售、制造、任务分派和企业供应链等的各项管理业务。

具体而言，ERP 管理系统主要由六大功能目标组成：

一是支持企业整体发展战略经营系统。该系统的目标是在多变的市场环境中建立与企业整体发展战略相适应的战略经营系统，还需要建立与 Intranet、Internet 相连接的战略系统、决策支持服务体系等。

二是实现全球大市场营销战略与集成化市场营销，也就是实现在预测、市场规模、广告策略、价格策略、服务、分销等各方面进行信息集成和管理集成。

三是完善企业成本管理机制。建立全面成本管理系统，建立和保持企业的成本优势。

四是研究开发管理系统，保证能够迅速地开发适应市场要求的新的产品，构筑企业的核心技术体系，保持企业的竞争优势。

五是建立敏捷的后勤管理系统，强调通过动态联盟模式把优势互补的企业联合在一起，

用最有效和最经济的方式参加竞争，迅速响应市场瞬息万变的需求。这种敏捷的后勤管理系统能够具有缩短生产准备周期，增加与外部协作单位技术和生产信息及时交互，改进现场管理方法，缩短供应周期等功能。

六是实施准时生产方式，把客户纳入产品开发过程，把销售代理商和供应商、协作单位纳入生产体系，按照客户不断变化的需求同步组织生产，时刻保持产品的高质量、多样性和灵活性。

ERP 对于企业提高管理水平具有重要意义。ERP 首先为企业提供了先进的信息系统平台。ERP 系统软件不仅功能齐全、集成性强、稳定性好，能够提供准确的信息，而且具备可扩充性。其次，ERP 具有规范的基础管理，促进企业管理水平提高的功能，ERP 实质上就是一套规范的由现代信息技术保证的管理制度。最后，ERP 能够整合企业各种资源，提高资源运作效率。

2．ERP 的主要功能

ERP 为企业提供的功能是多层面的和全方位的。

一是支持决策的功能。ERP 在 MRPⅡ的基础上扩展了管理范围，给出了新的结构，将企业内部业务流程划分成几个相互协同作业的支持子系统，如财务、市场营销、生产制造等，并在功能上增加了质量控制、运输、分销、售后服务与维护，以及市场开发、人事管理等功能，把企业的制造系统、营销系统、财务系统等都紧密地结合在一起，可以实现全球范围内的多工厂、多地点的跨国经营运作，因而，能够不断地收到来自各个业务过程运作信息，并且提供了对质量控制、适应变化、客户满意度、效绩等关键问题的实时分析，从而有力地支持企业的各个层面上的决策。

二是为处于不同行业的企业提供有针对性的 IT 解决方案。ERP 已打破了 MRPⅡ只局限在传统制造业的格局，把应用扩展到其他行业，如金融业、通信业、零售业等，并逐渐形成了针对于某种行业的解决方案。这一点非常重要，这是因为，不论一个 ERP 软件的功能多么齐全，都无法覆盖所有行业中的特殊需求。一个企业由于其所在行业的原因，既有较为通用的需求，如采购、库存、计划、生产、质检、人事、财务等，还可能有一些与众不同的特殊需求，例如石油天然气行业中的勘探与开采、土地使用与租赁，电力行业中的输配电、电表的抄费计价，零售业中的补货、变价、促销等，这些都需要有特殊的功能来解决和管理，从而需要有一套针对该行业的解决方案。为此，有些 ERP 供应商除了传统的制造业解决方案外，还推出了商业与零售业、金融业、能源、公共事业、工程与建筑业等行业的解决方案，以财务、人事、后勤等功能为核心，加入每一行业特殊的需求。

三是从企业内部的供应链发展为全行业和跨行业的供应链。当前，任何一个企业都要在全球化的大市场中参与竞争，而竞争的规则就是优胜劣汰，因而，任何一个企业都不可能在所有业务上成为世界上的佼佼者。如果全部业务都由自己来承担，它必然面对所有相关领域的竞争对手。因此，只有联合该行业中其他上下游企业，建立一条业务关系紧密、经济利益相连的供应链实现优势互补，才能适应社会化大生产的竞争环境，共同增强市场竞争实力，因此，供应链的概念就由狭义的企业内部业务流程扩展为广义的全行业供应链及跨行业的供应链。这种供应链或是由物料获取并加工成中间件或成品，再将成品送到消费者手中的一些企业和部门的供应链所构成的网络，或是由市场、加工、组装环节与流通环节建立一个相关业务间的动态企业联盟来进行跨地区、跨行业经营，以更有效地向市场提供商品和服务来完成单个企业不能承担的市场功能。这样，ERP 的管理范围亦相应地由企业的内部拓展到整个行业的原材料供应、生产加工、配送环节、流通环节及最终消费者。在整个行业中建立一个环环相扣的供应链，使多个企业能在一个整体的 ERP 管理下实现协作经营和协调运作。把这些企业的分散计划纳入整个供应链的计划中，从而大大增强了该供应链在大市场环境中的整体优势，同时也使每个企业之间均可实现以最小的个别成本和转换成本来获得成本优势。例如，在供应链统一的 ERP 计划下，上下游企业可最大限度地减少库存，使所有上游企业的产品能够准确、及时地到达下游企业，这样既加快了供应链上的物流速度，又减少了各企业的库存量和资金占用。通过这种整体供应链 ERP 管理的优化作用，来到达整个价值链的增值。

这种在整个行业中上下游的管理能够更有效地实现企业之间的供应链管理，以此实现其业务跨行业、跨地区甚至跨国的经营，对大市场的需求作出快速的响应。在它的作用下，供应链上的产品可实现及时生产、及时交付、及时配送、及时地交达到最终消费者手中，快速实现资本循环和价值链增值，以最大限度地为产品市场提供完整的产品组合，缩短产品生产和流通的周期，使产品生产环节进一步向流通环节靠拢，缩短供给市场与需求市场的距离，既减少了各企业的库存量和资金占用，还可及时地获得最终消费市场的需求信息使整个供应链均能紧跟市场的变化。通过这种供应链 ERP 管理的优化作用，达到整个价值链的增值。

3．ERP 的主要功能模块

ERP 是将企业所有资源进行集成整合，简单地说是将企业的三大流：物流、资金流、信息流进行全面一体化管理的管理信息系统。

在企业中，一般的管理主要包括三方面的内容：生产控制（计划、制造）、物流管理（分销、采购、库存管理）和财务管理（会计核算、财务管理）。这三大系统本身就是一个集成

体，它们互相之间有相应的接口，能够很好地整合在一起来对企业进行管理。

希赛教育专家提示：随着企业对人力资源管理的重视和加强，已经有越来越多的 ERP 厂商将人力资源管理纳入了 ERP 系统。

表 14-1 是 ERP 系统的主要功能模块。

表 14-1 ERP 系统的主要功能模块

财会管理模块	会计核算模块	物流管理模块	分销管理模块
	财务管理模块		库存控制模块
生产控制管理模块	主生产计划		采购管理模块
	物料需求计划	人力资源管理模块	人力资源规划的辅助决策
	能力需求计划		招聘管理
	车间控制		工资核算
	制造标准		工时管理
			差旅费核算

14.4.3 客户关系管理

CRM 在坚持以客户为中心的理念的基础上，重构包括市场营销和客户服务等业务流程。CRM 的目标不仅要使这些业务流程自动化，而且要确保前台应用系统能够改进客户满意度、增加客户忠诚度，以达到使企业获利的最终目标。

1. CRM 的概念当今世界，几乎所有的企业都宣布坚持“以客户为中心”的理念。但是，怎样把一种好的理念变成企业真实的行动，却并不是一个轻而易举的事情。而引进客户关系管理无疑=是解决问题的重要举措。CRM 是一种旨在改善企业与客户之间关系的新型管理机制。它通过提供更快速、更周到的优质服务来吸引或保持更多的客户。CRM 集成了信息系统和办公系统等的一整套应用系统，从而确保了客户满意度的提高，以及通过对业务流程的全面管理来降低企业成本。

CRM 在坚持以客户为中心的理念的基础上，重构包括市场营销和客户服务等业务流程。CRM 的目标不仅要使这些业务流程自动化，而且要确保前台应用系统能够改进客户满意度、增加客户忠诚度，以达到使企业获利的最终目标。

需要强调的是脱离后台而只强调前台管理是不够的。只有以客户为中心的应用与能提供客户经验的内部后台系统的集成才可以为整个企业的运作带来所需要的效益。

CRM 实际上是一个概念，也是一种理念；同时，它又不仅是一个概念，也不仅是一种理念，它是企业参与市场竞争的新的管理模式，是一种以客户为中心的业务模型，并由集成了前台和后台业务流程的一系列应用程序来支撑。这些整合的应用系统保证了更令人满意的

客户体验，因而会使企业直接受益。

2．CRM 的背景

CRM 的出现体现了两个重要的管理趋势的转变。首先是企业从以产品为中心的模式向以客户为中心的模式的转变。这种转变有着深刻的时代背景，那就是随着各种现代生产管理和现代生产技术的发展，产品的差别越来越小，产品同质化的趋势则越来越明显，因此，通过产品差异化来细分市场从而创造企业的竞争优势也就变得越来越困难。其次，CRM 的出现还表明了企业管理的视角从“内视型”向“外视型”的转变。众所周知，Internet 及其他各种现代交通、通信工具的出现，使得世界变成了一个地球村，企业与企业之间的竞争，哪怕相隔千里万里，也都变成几乎是面对面的竞争。尤其是在我国，仅仅依靠 ERP 的“内视型”的管理模式已难以适应激烈的竞争，企业必须转换自己的视角，在向“外向型”转变的过程中整合自己的资源。

CRM 听起来是一个很好的概念，然而实施起来却不那么容易。因为，CRM 不只是一套产品，它还是触及到企业内许多独立部门的商业理念。

业界分析人士认为，企业的高层管理人员对 CRM 的认识至关重要，只有企业管理层接受了 CRM 的理念，CRM 才能在企业里成功地实施，因为只有技术显然是不够的。CRM 需要在整个企业范围内协调关系，开发信息资源。从主导 90 年代的 ERP 系统转变为将注意力集中在客户，通过市场营销和客户服务来优化业务价值的商业模式。在成功实施 CRM 解决方案之前企业需要认同这些新的、不同的商业技巧。企业的商业理念一定要反映在 CRM 应用上，并且在上至公司高层下到可能与客户发生关系的每位员工之间充分沟通。

3．CRM 的内容

业界一致认为，市场营销和客户服务是 CRM 的支柱性功能。这些是客户与企业联系的主要领域，无论这些联系发生在售前、售中还是售后。

（1）客户服务。客户服务是 CRM 的关键内容，是能否形成并保留大量忠诚客户的关键。随着市场竞争的深入，客户对服务的期望值也在不断地提高，已经超出传统的电话呼叫中心的范围。而呼叫中心正在向可以处理各种通信媒介的客户服务中心演变。电话互动必须与 E-mail、传真、网站以及其他任何客户喜欢使用的方式相互整合。随着越来越多的客户进入互联网通过浏览器来查看他们的订单或提出询问，自助服务的要求发展得也越来越快。客户服务已经超出传统的帮助平台。“客户关怀”的术语如今用来拓展企业对客户的职责范围。而与客户保持积极主动的关系是客户服务的重要组成部分。客户服务能够处理客户各种类型的询问，包括有关的产品、需要的信息、订单请求、订单执行情况等，还包括高质量的现场

服务。

（2）市场营销。营销自动化包括商机产生、商机获取和管理，商业活动管理及电话营销等。初步的大众营销活动用于首次客户接触，接下来是针对具体目标受众的更加集中的商业活动。个性化需求很快成为营销规范，客户的喜好和购买习惯都被列入商家关注的重点。旨在更好地向客户行销、带有有关客户特殊需求信息的目录管理和一对一行销应运而生。市场营销迅速从传统的电话营销转向网站和 Email。这些基于 Web 的营销活动给潜在客户更好的体验，使潜在客户以自己的方式、在方便的时间查看其所需要的信息。销售人员与潜在客户的互动行为、将潜在客户发展为真正客户并保持其忠诚度是使企业盈利的核心因素。为了获得最大的价值，企业管理层必须与销售人员合作，并对这些商业活动进行跟踪，以激活潜在消费并进行成功/失败研究。市场营销活动的费用管理及营销事件（如贸易展和研讨会)对未来计划的制定至关重要。

（3）共享的客户资料库。共享的客户资料库把市场营销和客户服务连接起来。集成整个企业的客户信息会使企业从部门化的客户联络提高到与客户协调一致的高度。如果一个企业的信息来源相互独立，那么这些信息中必然会存在大量重复、互相冲突的成分。这对企业的整体运作效率将产生负面影响。而动态的、能够被不同部门共享的客户资料库则是企业的一种宝贵资源，同时，它也是 CRM 的基础和依托。

（4）分析能力。CRM 的一个重要方面在于它具有使客户价值最大化的分析能力。如今的 CRM 解决方案在提供标准报告的同时又可提供既定量又定性的即时分析。

深入的智能分析需要统一的客户数据作为切入点，并使所有企业业务应用系统融入到分析环境中，通过对客户数据的全面分析、评估客户带给企业的价值及衡量客户的满意度，再将分析结果反馈给管理层，这样便增加了信息分析的价值。企业决策者会权衡这些信息做出更全面、更及时的商业决策。

4.CRM 的解决方案和实施过程

CRM 的根本要求就是与客户建立起一种互相学习的关系，即从与客户的接触中了解他们在使用产品中遇到的问题，以及对产品的意见和建议，并帮助他们加以解决。在与客户互动的过程中，了解他们的姓名、通信地址、个人喜好及购买习惯，并在此基础上进行 “一对一”的个性化服务，甚至拓展新的市场需求。例如，用户在订票中心预订了机票之后，CRM 就会根据了解的信息，向用户提供唤醒服务或出租车登记等增值服务。因此，可以看到，CRM 解决方案的核心思想就是通过跟客户的“接触”，搜集客户的意见、建议和要求，并通过数据挖掘和分析，提供完善的个性化服务。

一般说来 CRM 由两部分构成，即触发中心和挖掘中心，前者指客户和 CRM 通过电话、传真、Web、E-mail 等多种方式“触发”进行沟通；挖掘中心则是指对 CRM 记录交流沟通的信息进行智能分析。由此可见，一个有效的 CRM 解决方案应该具备以下要素：

（1）畅通有效的客户交流渠道（触发中心）。在通信手段极为丰富的今天，能否支持电话、Web、传真、E-mail 等各种触发手段进行交流，无疑是十分关键的。

（2）对所获信息进行有效分析（挖掘中心）。

（3）CRM 必须能与 ERP 很好地集成。作为企业管理的前台，CRM 的市场营销和客户服务的信息必须能及时传达到后台的财务、生产等部门，这是企业能否有效运营的关键。

CRM 的实现过程具体说来，包含三方面的工作。一是客户服务与支持，即通过控制服务品质以赢得顾客的忠诚度，例如，对客户快速准确的技术支持、对客户投诉的快速反应、对客户提供产品查询等；二是客户群维系，即通过与顾客的交流实现新的销售，例如，通过交流赢得失去的客户等；三是商机管理，即利用数据库开展销售，例如，利用现有客户数据库做新产品推广测试，通过电话促销调查，确定目标客户群等。

5．CRM 的价值

CRM 之所以受欢迎是因为好的客户关系管理对客户和企业都有益。CRM 用户从不断加强的客户关系管理中明显获益。好的服务不但令人愉快，更能带来巨大价值。带有客户服务的产品的总价值明显高于产品自身。

从另一方面看，企业实施 CRM 并非出于利他原则，而是认识到客户是其真正的财富。统计显示，68%的客户离开厂家是因为得不到令人满意的客户服务，而企业 80%的收入来源于老客户。CRM 的成功应用，其效果是显而易见的。

较高的满意度，使得企业能够保留老客户，并不断增加新客户；

识别利润贡献度最高的客户并给以相应的优厚对待；

通过有效目标市场定位，来降低营销成本；

引导潜在消费至适当的销售渠道；

提供正确的产品来增加销售（交叉销售/纵向销售）；

简化部门工作流程来缩短销售周期；

通过集中共同活动以减少多余运作；

减少由于多个不协调的客户交互点而产生的差错，节省费用；

利用客户喜欢的沟通渠道来增加对客户需求的了解；

参照与其他客户的联络记录和经验，与目前的客户进行沟通；

根据对以前绩效的分析评估未来的销售、营销和客户服务活动；

由于 CRM 对企业的重大影响，实施 CRM 项目时需要整个企业范围内的认识与运作。为保持竞争优势，企业必须投资于 CRM 技术，同时要建立新的业务模型。所有客户信息的集中是成功实施的 CRM 的核心。CRM 这一强有力的企业策略将提高销售、客户忠诚度和企业的竞争优势。

14.4.4 产品数据管理

PDM 是工程数据管理、文档管理、产品信息管理、技术数据管理、技术信息管理、图像管理，以及其他产品定义信息管理的集成管理框架技术。

1．PDM 简介

自 20 世纪 80 年代企业实施信息化以来，各种各样的信息系统的开发及信息工具的使用给企业带来了丰富的成果，但同时，其弊端也不断显现，最为明显的是信息化带来了数据爆炸和数据混乱的问题。各种高效的信息工具的数据处理能力和存储能力不断提高，产生的数据呈几何级数式地增长。大量的数据文件，由于缺少统一的管理和调度，本来是宝贵的资源却变成了“死数据”或“垃圾数据”，无法被应用。

在这种情况下，PDM 应运而生。早期的 PDM 系统是在 20 世纪 80 年代出现的，当时功能比较单一，主要是产品信息管理，现在已经发展为支持更多功能的信息系统，如企业组织内各种有关产品的数据管理，包括文字文件、图形文件、产品结构形式、电子数据发布和更改过程，以及基于设计能力的构件技术。

由于 PDM 特别关注企业的管理需求，从而使得从产品的概念设计到产品生产的整个过程，以及过程之外所产生的数据被科学、高效地管理起来。20 世纪末的 PDM 继承并发展了“集成制造”等技术的核心思想，在系统工程思想的指导下，用整体优化的观念对产品设计数据和设计过程进行描述，规范产品生命周期管理，保持产品数据的一致性和可跟踪性。PDM 的核心思想是设计数据的有序、设计过程的优化和资源的共享，通过人、过程、技术三者的平衡使虚拟制造过程进一步增值。

PDM 虽然已经得到较广泛的应用，但至今还没有一个统一的定义，下面是国际上三个业界的权威分别给出的三种不同的定义：

PDM 是一门用来管理所有与产品相关信息（包括零件信息、配置、文档、计算机辅助设计文件、结构、权限信息等）和所有与产品相关过程（包括过程定义和管理）的技术。

PDM 是为企业设计和生产构筑一个并行产品开发环境（由供应、工程设计、制造、采购、销售与市场、客户构成）的关键技术。一个成熟的 PDM 系统能够使所有参与创建、交流、维护设计意图的人在整个信息生命周期中自由共享和传递与产品相关的所有异构数据。PDM 系统是一种软件框架，利用这个框架可以帮助企业实现对与企业产品相关的数据、开发过程，以及帮助使用者进行集成与管理，可以实现对设计、制造和生产过程中需要的大量数据进行跟踪和支持。

PDM 系统是帮助产品设计师、制造工程师及其他人员有效管理产品数据及产品开发过程的工具，目标是跟踪、组织、访问和管理产品设计、开发、修改和生产，甚至维修全过程中的所有数据和信息。PDM 能够跨越时间和操作环境，实现数据的无缝连接和移动，保证正确的数据、在正确的时间、以正确的格式、出现在正确的位置，进而推动产品尽快地投入市场并能有效地平衡生产能力。目前，PDM 已广泛地应用于航空航天、汽车、机械、电子等生产制造领域。

PDM 系统的用户主要有三类：一是信息的使用者，他们要求最简单的用户界面；

二是数据的创造者，如机械工程师和电子工程师等，他们希望 PDM 系统能够很好地融入到产品设计应用中；

三是系统管理员，他们面对的是最复杂的用户界面。

由于 PDM 发展很快，用户的需求也在不断变化，所以 PDM 软件产品版本更新很快，但总的趋势是功能模块化、操作简单化，并针对不同用户提供不同的界面。

2．企业对 PDM 的需求

目前，在现代企业中，每天大约有三分之二到四分之三的设计、管理、工程技术人员不是专心于他们自己的本职工作，而是开会、讨论、协调、调度、等待或在处理各种信息，其目的主要是促使他们之间相互理解。由于工程设计到制造控制系统缺乏产品数据的统一管理，导致产品构型工时增加 20%，成本上升 10%。所以产品数据管理是当代企业管理的瓶颈，已逐步引起工程技术界的普遍注意，并开始对它进行研究和开发。

然而，PDM 系统强大的功能妥善地解决上述问题，为企业更好地实施 CIMS 和并行工程提供了底层支持。PDM 能够描述复杂数据的类型和结构，动态地定义和修改数据模式和严格地约束管理等。实施 PDM 可以：

一是在企业内部建立起完整的、统一的、共享的数据模型，保证各部门的产品信息一致。

二是缩短产品的上市时间。在需要数据的时候立即得到这些数据，加快任务的完成；支持并行工程；允许授权小组的成员随时访问最新的版本的相关数据。

三是适应多品种小批量生产方式。通过产品结构和配置管理为用户提供了系列产品的有效管理方法，并可以快速地响应市场的需求，实现“面向订单”的生产方式。

四是提高设计效率和提高生产效率，降低产品成本。

五是提高设计与制造的准确性，提高产品质量。

六是保护数据完整性。PDM 系统提供权限控制和变更管理确保产品数据的准确和安全。

七是更好地控制项目。项目管理功能提供用户对项目的进展情况进行实施监控，确保项目顺利进行和如期完成。

八是实现全面的质量管理。PDM 系统可以建立适应 ISO9000 系列验证和全面质量管理的环境，通过在产品全生命周期内的工作流程管理确保了产品的最终质量。九是建立起企业级的协同工作平台，为最终实现企业的电子商务打下坚实的基础。

3．PDM 的发展过程

PDM 的核心思想是设计数据的有序、设计过程的优化和资源的共享。PDM 技术的发展可以分为以下三个阶段：

（1）配合 CAD（Computer Aided Design，计算机辅助设计）使用的早期简单的 PDM 系统。PDM 技术出现初期，大多是由各 CAD 供应商推出的配合 CAD 产品的系统，主要局限在工程图纸的管理，解决了大量工程图纸、技术文档及 CAD 文件的计算机管理问题。这是第一代 PDM 产品。主要表现形式为各类文档管理或图纸管理的软件系统等。

（2）产品数据管理。20 世纪 90 年代初中期，出现了专业化的 PDM 产品，如 SDRC 公司的 Metaphase、IDS 公司的 iMAN、IBM 公司的 PM、SmartSolution 公司的 SmarTeam 等。与第一代 PDM 产品相比，在第二代 PDM 产品中出现了许多新功能，如对产品生命周期内各种形式的产品数据的管理能力、对产品结构与配置的管理、对电子数据的发布和工程更改的控制，以及基于成组技术的零件分类管理与查询等，同时软件的集成能力和开放程度也有较大的提高，少数优秀的 PDM 产品可以真正实现企业级的信息集成和过程集成。第二代 PDM 产品目前被广泛使用。

（3）产品协同商务（Collaborative Product Commerce，CPC）或 PDM 标准化。第三代 PDM 产品具有代表性的有：PTC 公司的 Windchill、MatrixOne 公司的 eMatrix 等。这些产品建立在 Internet 平台、CORBA 和 Java 技术基础之上，并且是基于分布式计算框架，做到了与计算机软硬件平台无关和用户界面的统一，支持以“标准企业职能”和“动态企业” 思想为中心的新的企业信息分析方法，可以进行企业信息建模的分析和设计，实现包括文档管理、生命周期管理、工作流管理、产品结构管理、视图管理、变更管理、客户化应用等功能。

第三代 PDM 适应了信息时代广义企业异地协同开发、制造和管理产品的要求。

4．PDM 主要功能模块和内容

虽然 PDM 是一个很复杂的系统，但从结构上讲，它基本上可以分成面向信息集成的系统集成工具和面向用户的 PDM 功能模块两部分。

（1）数据基库：PDM 的核心，它一般建立在关系型数据库系统的基础上，主要保证数据的安全性和完整性，并支持各种查询和检索功能。

（2）产品配置管理：以数据基库为底层支持，以 BOM 为组织核心，把定义最终产品的所有工程数据和文档联系起来，对产品对象及其相互之间的联系进行维护和管理。

（3）工作流管理：主要实现产品设计与修改过程的跟踪与控制。包括工程数据的提交与修改、管理和监督、文档的分布控制、自动通知等。

（4）分类及检索功能：与面向对象的技术相结合，将具有相似特性的数据与过程分为一类，并赋予一定的属性和方法，使用户能够在分布式环境中高效地查询文档、数据、零件、标准件等对象。常用的分类技术有：使用智能化的零件序号、成组技术、搜索/检索技术、零件建库技术。

（5）项目管理：在项目实施过程中实现其计划、组织、人员及相关数据的管理与配置，进行项目运行状态的监视，完成计划的反馈。到目前为止，许多 PDM 系统只能提供工作流程活动的状态信息。

PDM 的传统模块的功能已经很成熟，PDM 供应商会推出一些专用模块，将原先使用专用工具完成的功能转化为 PDM 的可选模块。例如，项目管理、工程更改、供应商和零部件管理原先有很多专用软件，现在 PDM 供应商已把这些功能囊括在 PDM 中或者提供独立的专用模块完成。

以某著名的 PDM 产品为例，典型的 PDM 系统包括以下功能：数据仓库和文档管理、工作流程/过程管理、产品结构/配置管理、查看和圈阅、设计检索和零件库、项目管理、工具和集成件。其他功能诸如扫描和图像服务、电子协作、通信和通告、数据传输和数据翻译等。

14.4.5 企业门户

随着互联网的快速发展，企业门户已经成为企业优化业务模式、扩展市场渠道、改善客户服务，以及提升企业形象和凝聚力的强有力手段。企业门户之所以具有极大的吸引力，关

键在于它具备广泛的用途和灵活、全面的模型。随着电子商务的发展，企业门户已经成为新型办公环境的重要组成部分。从电子商务应用到企业内部的信息系统，所有用户友好型信息搜集系统都以基于各种技术的企业门户的形式出现。不过，如果要给企业门户下一个确切的定义，目前还做不到，因为还没有一个公认的企业门户标准。

1．企业门户的功能通常，企业需要更高效能且技术统一的平台，以整合当前的网上业务，同时让系统本身能够随时便利升级，以支持未来网上业务的发展。建设集多种功能（如客户关系管理、网上销售、知识管理、内容管理等）于一身的企业门户网站，成为势在必行的上网策略。

一直以来，门户网站仍局限于提供内容、电子邮箱及搜索引擎等基本功能，针对的主要是大众消费类市场；随着互联网应用于企业市场，企业将各类型业务搬到一个开放统一而且安全度很高的网上平台，便成为其电子商务架构中的重要环节。

据相关独立分析员预测，门户网站的趋势将会主导今后几年的企业计算机应用潮流。电子商务需要有更明确的投资回报评估，由此也导致企业对门户网站技术的需求急剧增加。企业门户网站已经显现出提升竞争力的功用：一方面可以让雇员更方便地存取信息，另一方面又可以加强与客户和伙伴之间的联系。

值得一提的是，不同的企业将不尽相同的网络系统连接至单一开放式企业门户网站上，可以大大降低管理成本。因此，企业门户的主要功能有：

（1）能够将一个机构现有的互联网址和服务完全合并而且相互兼容。

（2）能够支持开放标准和应用编程接口，让平台得以轻易容纳新的应用程序。

（3）能够接入一个由支持企业门户网站架构的伙伴和专业服务公司所组成的网络。

（4）能够多渠道接入网站，如互联网至公司内联网、话音网络、无线网络等。

（5）能够以统一的服务作为企业门户网站各种服务的基础，让用户享有多种便利，如一次登入、个人化接口等。当用户进入门户网站的不同部分时，系统可以记住用户的身份以提供合适的信息。

总之，门户网站应该是一个起点，引领用户接触企业最重要的信息、应用和服务。门户网站并非仅为个人计算机用户标准应用而设，它应该能够根据用户的身份、意向、接入方式、接入设备（如移动电话）等设定个性化的信息内容。

2．企业门户的分类按照实际应用领域，企业门户可以划分为三类：信息门户、知识门户和应用门户。

（1）企业信息门户。企业信息门户（Enterprise Information Portal，EIP）的基本作用是

为人们提供企业信息，它强调对结构化与非结构化数据的收集、访问、管理和无缝集成。这类门户必须提供数据查询、分析、报告等基本功能，企业员工、合作伙伴、客户、供应商都可以通过企业信息门户非常方便地获取自己所需的信息。

对访问者来说，企业信息门户提供了一个单一的访问入口，所有访问者都可以通过这个入口获得个性化的信息和服务，可以快速了解企业的相关信息。对企业来说，信息门户既是一个展示企业的窗口，也可以无缝地集成企业的业务内容、商务活动、社区等，动态地发布存储在企业内部和外部的各种信息，同时还可以支持网上的虚拟社区，访问者可以相互讨论和交换信息。

在目前企业门户的应用中，信息门户被企业所广泛认同。实际上，各企业建立的企业网站都可以算作企业信息门户的雏形。

（2）企业知识门户。企业知识门户（Enterprise Knowledge Portal，EKP）是企业员工日常工作所涉及相关主题内容的“总店”。企业员工可以通过它方便地了解当天的最新消息、工作内容、完成这些工作所需的知识等。通过企业知识门户，任何员工都可以实时地与工作团队中的其他成员取得联系，寻找到能够提供帮助的专家或者快速地连接到相关的门户。不难看出，企业知识门户的使用对象是企业员工，它的建立和使用可以大大提高企业范围内的知识共享，并由此提高企业员工的工作效率。

当然，企业知识门户还应该具有信息搜集、整理、提炼的功能，可以对已有的知识进行分类，建立企业知识库并随时更新知识库的内容。目前，一些咨询、服务型企业已经开始建立企业知识门户。

（3）企业应用门户。企业应用门户（Enterprise Application Portal，EAP）实际上是对企业业务流程的集成。它以商业流程和企业应用为核心，把商业流程中功能不同的应用模块通过门户技术集成在一起。从某种意义上说，可以把企业应用门户看成是企业信息系统的集成界面。企业员工和合作伙伴可以通过企业应用门户访问相应的应用系统，实现移动办公、进行网上交易等。

以上三类门户虽然能满足不同应用的需求，但随着企业信息系统复杂程度的增加，越 来越多的企业需要能够将以上三类门户有机地整合在一起的通用型企业门户。按照 IDC 的定义，通用型的企业门户应该随访问者角色的不同，允许其访问企业内部网上的相应应用和信息资源。除此之外，企业门户还要提供先进的搜索功能、内容聚合能力、目录服务、安全性、应用/过程/数据集成、协作支持、知识获取、前后台业务系统集成等多种功能。给企业员工、客户、合作伙伴、供应商提供一个虚拟的工作场所。

3．企业门户的要素当前，一些企业已经在利用不同的平台和多种互联网／内联网服务开展网上运营。企业门户网站最重要的目标，是将多个系统整合到一个具有可扩充性的平台上，为提供多元化的网上服务做好准备，以最少的投资赚取最高收益。企业可以在基本平台上对各种应用程序加以整合，同时做到支持第三方应用程序所需的标准。

以下是建立互联网服务时应考虑的基本要素：

（1）战略性思维。评估未来的需求，并将这些需要与影响业务发展的因素一并考虑，例如处理客户数据时个人隐私及安全问题。

（2）为用户所需要的不同类型门户网站建立一个门户网站架构。

（3）寻找合适的技术供货商——既能够支持各主要标准，并能够将其基本门户网站架构与其他供货商的应用程序整合起来。

（4）确定所要建立的门户网站类型，如销售门户网站或知识管理门户网站。制定可量化的目标，并清楚界定投资回报。如果对进展感到满意，就可逐步实行门户网站策略的其他元素。

（5）首先小规模地试办项目，确保有一个可行的工作环境。接着，如果用户的工作队伍决定加入新服务，就可相应地扩充项目。

14.4.6 企业应用集成

EA 伴随着企业信息系统的发展而产生和演变。企业的价值取向是推动应用集成技术发展的原动力，而应用集成的实现反过来也驱动公司竞争优势的提升。EAI 技术将进程、软件、标准和硬件联合起来，在两个或更多的企业信息系统之间实现无缝集成，使它们就像一个整体一样。EAI 一般表现为对一个商业实体（例如一家公司）的信息系统进行业务应用集成，但当遇到多个企业系统之间进行商务交易时，EAI 也表现为不同公司实体之间的企业系统集成，例如，B2B 的电子商务。

1．EAI 的简要历史

计算机广泛商业应用开始于 20 世纪 70 年代。当时，企业应用的主要目标是利用计算机来代替一部分烦琐的重复性手工工作，借以提高生产效率。这时还没有企业数据集成的需求。

到了 20 世纪 80 年代，许多企业，特别是大型跨国公司在信息系统上投入了巨资，建立了众多的应用信息系统，以帮助企业进行内部或外部业务的处理和管理。由于企业的传统

职能结构，企业整体功能被各个部门所分割，使得信息系统也自然为各个部门所独占，其结果是导致众多关键的信息被封闭在相互独立的系统中，形成一个个所谓的信息孤岛。

如何将众多的信息孤岛联系起来，以便让不同的系统之间交互信息，EAI 就作为一个企业的需求被提了出来，这时，EAI 的价值和必要性也开始体现。

企业在追求效率和控制成本，或在兼并和收购过程中，对应用集成技术提出了更高的要求，特别是电子商务的兴起。电子商务，这一基于 Internet 的新的商务模式直接导致新的系统集成结构的出现，像 Web 服务技术等。特别是 20 世纪 90 年代，ERP 应用开始流行，也要求新的信息系统能够支持已经存在的应用和数据，这就必须引入 EAI。还有应用供应链管理、Web 应用集成等也对 EAI 起到推动作用。

2．EAI 的内容

EAI 的内容极为广泛，同时，其意义也十分重大，它是企业信息化发展到较高阶段的标志。因为，在企业范围内现有的应用系统和数据库有可能既有几年前的老系统，还可能包括新建系统，需要对它们进行无缝地集成；不同的系统和应用可能包括同样的数据，从而造成了数据冗余、数据的不一致，需要尽量减少数据，并保持所有的数据版本同步更新；企业在激烈的市场竞争中，经常根据需要调整业务流程，必然影响到信息系统的结构和数据，或是建立新的系统。

总之，EAI 是企业信息系统集成的科学、方法和技术，其目的就是将企业内的应用彼此连接起来，或在企业之间连接起来。

EAI 主要包括两方面：企业内部应用集成和企业间应用集成。EAI 包括的内容很复杂，涉及结构、硬件、软件及流程等企业系统的各个层面。

（1）企业内的集成

企业内的应用集成，就是要解决在企业内部的业务流程和数据流量的问题，包括业务流程是否进行自动流转，或怎样流转，以及业务过程的重要性。对于应用集成，这点非常重要，因为从本质上讲，企业应用集成就是维持数据正确而自动地流转。同时，不同的 EAI 解决方案采取不同的技术途径，而不同的技术途径也就决定了 EAI 处于不同的层次，从应用和技术上综合考虑，EAI 分为界面集成、平台集成、数据集成、应用集成和过程集成。

① 界面集成。这是比较原始和最浅层次的集成，但又是常用的集成。这种方法就是把用户界面作为公共的集成点，把原有零散的系统界面集中在一个新的、通常是浏览器的界面之中。

② 平台集成。这种集成要实现系统基础的集成，使得底层的结构、软件、硬件及异构网络的特殊需求都必须得到集成。平台集成要应用一些过程和工具，以保证这些系统进行快速安全的通信。

③ 数据集成。为了完成应用集成和过程集成，必须首先解决数据和数据库的集成问题。在集成之前，必须首先对数据进行标识并编成目录，另外还要确定元数据模型，保证数据在数据库系统中分布和共享。

④应用集成。这种集成能够为两个应用中的数据和函数提供接近实时的集成。例如，在一些 B2B 集成中实现 CRM 系统与企业后端应用和 Web 的集成，构建能够充分利用多个业务系统资源的电子商务网站。

⑤ 过程集成。当进行过程集成时，企业必须对各种业务信息的交换进行定义、授权和管理，以便改进操作、减少成本、提高响应速度。过程集成包括业务管理、进程模拟等，还包括业务处理中每一步都需要的工具。

（2）企业间应用集成

EAI 技术可以适用于大多数要实施电子商务的企业，以及企业之间的应用集成。EAI 使得应用集成架构里的客户和业务伙伴，都可以通过集成供应链内的所有应用和数据库实现信息共享。

传统的 B2B 商务应用了诸如 EDI（Electronic Data Interchange，电子数据交换）和专用 VAN（Value Added Network，增值网络）的技术。然而今天，大多数 B2B 商务则采用了实时性更强的、基于 Internet 的技术，如基于 Internet 的消息代理技术、应用服务器，以及像 XML 等新的数据交换标准。

许多公司的供应链系统也可能包括交易系统，新的 EAI 技术可以首先在交易双方之间创建连接，然后再共享数据和业务过程。当然，他们如今不再使用 VAN，而采用 Internet 作为传输介质。

企业要顺利开展电子商务，都希望其所有的应用之间，以及与其商业伙伴之间都能够实现无缝而及时的通信，这一目标在以前是比较难于实现的，因为，EAI 解决方案比较昂贵，直到新一代支持 EAI 的中间件的出现，才改变了这一面貌。

和 B2B 商务有所不同，B2C 商务需要信息能被更广泛的企业之外的人或客户访问到，所以企业应用要能支持基于 Web 的销售和信息共享。显而易见，B2B 和 B2C 方面的需要促进了 EAI 技术的发展。

3．集成技术的发展

展望目前市场主流的集成模式有三种，分别是面向信息的集成技术、面向过程的集成技术和面向服务的集成技术。

在数据集成的层面上，信息集成技术仍然是必选的方法。信息集成采用的主要数据处理技术有数据复制、数据聚合和接口集成等。其中，接口集成仍然是一种主流技术。它通过一种集成代理的方式实现集成，即为应用系统创建适配器作为自己的代理，适配器通过其开放或私有接口将信息从应用系统中提取出来，并通过开放接口与外界系统实现信息交互，而假如适配器的结构支持一定的标准，则将极大地简化集成的复杂度，并有助于标准化，这也是面向接口集成方法的主要优势来源。标准化的适配器技术可以使企业从第三方供应商获取适配器，从而使集成技术简单化。

面向过程的集成技术其实是一种过程流集成的思想，它不需要处理用户界面开发、数据库逻辑、事务逻辑等，而只是处理系统之间的过程逻辑，与核心业务逻辑相分离。在结构上，面向过程的集成方法在面向接口的集成方案之上，定义了另外的过程逻辑层；而在该结构的底层，应用服务器、消息中间件提供了支持数据传输和跨过程协调的基础服务。对于提供集成代理、消息中间件及应用服务器的厂商来说，提供用于业务过程集成是对其产品的重要拓展，也是目前应用集成市场的重要需求。

基于 SOA 和 Web 服务技术的应用集成是业务集成技术上的一次重要的变化，被认为是新一代的应用集成技术。集成的对象是一个个的 Web 服务或者封装成 Web 服务的业务处理。Web 服务技术由于是基于最广为接受的、开放的技术标准（例如，HTTP、SMTP 等），支持服务接口描述和服务处理的分离、服务描述的集中化存储和发布、服务的自动查找和动态绑定及服务的组合，成为新一代面向服务的应用系统的构建和应用系统集成的基础设施。

14.4.7 供应链管理

供应链管理是从源头供应商到最终消费者的集成业务流程。它不仅为消费者带来有价值的产品和服务，还为顾客带来有用的信息。

1．供应链管理的定义

SCM 的核心是供应链。供应链是指一个整体的网络，用来传送产品和服务，从原材料开始一直到最终客户（消费者），它凭借一个设计好的信息流、物流和现金流来完成。现代意义的供应链是利用计算机网络技术全面规划供应链中的商流、物流、信息流、资金流等并进行计划、组织、协调和控制。

供应链有两层含义，一层含义是任何一个企业内部都有一条或几条供应链，包括从生产到发货的各个环节；另一层含义是一个企业必定处于市场更长的供应链之中，包括从供应商的供应商到顾客的顾客的每一个环节。供应链是企业赖以生存的商业循环系统，是企业电子商务中最重要的课题。统计数据表明，企业供应链可以耗费企业高达 25%的运营成本。

供应链管理是从源头供应商到最终消费者的集成业务流程。它不仅为消费者带来有价值的产品和服务，还为顾客带来有用的信息。供应链管理至少包括以下六大应用功能：需求管理（预测和协作工具)、供应链计划（多工厂计划)、生产计划、生产调度、配送计划、运输计划。新型的供应链管理借助于 Internet 使这个“供应群”能够实现大规模的协作，成为企业降低成本、提高经营效率的关键。

而在计算机广泛应用之前，企业经常出现因信息传递太慢或错误而误导生产及存货计划的现象。20 世纪 90 年代，一些计算机的制造商，如 HP，或生产家庭用品的企业，如宝洁，开始将信息系统作上、下游整合，希望通过正确和快速的信息传递，以及对信息的分析和整合，达到快速反映市场的需求，从而降低库存等目的。因此，有效的供应链管理是建立在高质量的信息传递和共享的基础之上的。

2．供应链与物流

供应链与物流的关系极为密切，而且不可分割。供应链管理是一种管理方法或思想，而物流是在现实经营活动中的物质运动，供应链管理思想是从物流管理的实践中提取出来的，管理的对象是物流；物流分为采购物流、生产物流、销售物流，而供应链管理将这些全部纳入到一个管理体系之中，在供应商、分销商、零销商之间搭建起一个流畅的通道，建立起一个信息共享的机制，从而优化整个供应链，达到降低成本、提高效率等目的。物流的概念诞生在 20 世纪 20 年代的美国，当时更多是指商品的移动，怎样通过一个载体把商品从生产者手中送到消费者手中。到了 20 世纪 80 年代，人们发现以前的概念只是消费物流，忽视了两个环节，即采购环节的原材料物流，以及在企业内部进行加工生产的生产物流，于是人们又提出来一个整体现代的物流概念。原材料物流对企业来说可能更有意义，因为从采购环节来控制原材料的成本，可以大大降低企业的整体产品的成本，提高产品的竞争力，所以人们这时候发现，通过这种物流的管理，给企业带来的效益是非常大的，这是物流从狭义到广义的变化。

3．供应链管理是一种管理思想

随着 Internet 的普及，物流管理很自然地上升为供应链管理。因为在整个交易过程中可能会存在一些矛盾和冲突，供应链管理可以起到弥合整个体系中的矛盾和冲突。例如，以

前可能由分销商承担中间运输环节的工作，从供应商处取货送到零售商。后来，零售企业可能根据自己的效益和规模组建了自己的配送中心。这时分销商自己的物流体系可能就发挥不了太大作用，并且，为了按时将零售商需要的商品送到，分销商还需要备好库存，从而加大了成本。这就形成了利益冲突。而通过供应链管理的思想和方法协调供应商、分销商、零销商之间的关系，明确各自在整个体系中所处的角色，搭建一个良好的合作框架。这是各方进一步协同合作的基础。供应链管理一个重要的前提是信息共享，而各种版本 SCM 产品，其核心功能其实是信息传递。如果没有 SCM，也可以依据这样的思想进行人工的信息传递和管理，如派人到超市查看自己产品的库存等，只是这样效率比较低。

4．供应链管理的运作模式

供应链中的信息流覆盖了从供应商、制造商到分销商，再到零售商等供应链中的所有环节。其信息流分为需求信息流和供应信息流，这是两个不同流向的信息流。当需求信息（如客户订单、生产计划、采购合同等）从需方向供方流动时，便引发物流。同时供应信息（如入库单、完工报告单、库存记录、可供销售量、提货发运单等）又同物料一起沿着供应链从供方向需方流动。

由于供应链中的企业是一种协作关系和利益共同体，因而供应链中的信息获取渠道众多，对于需求信息来说既有来自顾客也有来自分销商和零售商的；供应信息则来自于各供应商，这些信息通过供应链信息系统而在所有的企业里流动与分享。对于单个企业情况来说，由于没有与上下游企业形成利益共同体，上下游企业也就没有为它提供信息的责任和动力，因此单个企业的信息获取则完全倚赖于自己的收集。

处于供应链核心环节的企业要将与自己业务有关（直接和间接）的上下游企业纳入一条环环相扣的供应链中，使多个企业能在一个整体的信息系统管理下实现协作经营和协调运作，把这些企业的分散计划纳入整个供应链的计划中，实现资源和信息共享，增强了该供应链在市场中的整体优势，同时也使每个企业均可实现以最小的个别成本和转换成本来获得成本优势。这种网络化的企业运作模式拆除了企业的围墙，将各个企业独立的信息孤岛连接在一起，通过网络、电子商务把过去分离的业务过程集成起来，覆盖了从供应商到客户的全部过程。对供应链中的企业进行流程再造，建立网络化的企业运作模式是建立企业间的供应链信息共享系统的基石。

统一的信息系统架构是决定信息能否共享的物质技术基础，主要包括：为系统功能和结构建立统一的业务标准和建立统一的信息交流规范体系等。因为即使某些细节之处没有遵循共同的标准也会影响数据交流和信息共享。例如，供应链中的企业通过 EDI 进行数据交换

时，双方必须严格遵守文件的标准格式，任意一方擅自改动格式都将导致对方的系统无法正常工作。

5．供应链管理的技术支持体系

供应链信息系统的建立需要大量信息技术来支持，这是因为供应链管理涉及众多的领域：产品（服务)设计、生产、市场营销（销售）、客户服务、物流供应等。它是以同步化、集成化生产计划为指导，通过采用各种不同信息技术来提高这些领域的运作绩效。

信息技术对供应链的支撑可分为两个层面。

第一个层面是由标识代码技术、自动识别与数据采集技术、电子数据交换技术、互联网技术等基础信息技术构成。

第二层面是基于信息技术而开发的支持企业生产。

在具体集成和应用这些系统时，不应仅仅将它们视为是一种技术解决方案，而应深刻理解它们所折射的管理思想，涉及的技术和方法主要有：销售时点信息系统、电子自动订货系统、计算机辅助设计和计算机辅助制造、ERP 和 MRPII、CRM、电子商务等。

14.4.8 电子商务概述

电子商务是一项涉及全球的全新业务和全新服务，是网络化的新型经济活动，它不仅仅是基于互联网的新型交易或流通方式，还是基于互联网、广播电视网和电信网络等电子信息网络的生产、流通和消费活动。

1．什么是电子商务

电子商务（Electronic Commerce，EC）是指买卖双方利用现代开放的 Internet，按照一定的标准所进行的各类商业活动。主要包括网上购物、企业之间的网上交易和在线电子支付等新型的商业运营模式。产品可以是实体化的，如计算机、汽车、电视，也可以是数字化的，如新闻、影像、软件；也可以直接提供服务，如安排旅游、远程教育等。

电子商务分三个方面：即电子商情广告、电子选购和交易及电子交易凭证的交换、电子支付与结算以及网上售后服务等。

参与电子商务的实体有四类：顾客（个人消费者或集团购买）、商户（包括销售商、制造商、储运商）、银行（包括发卡行、收单行）及认证中心。

狭义的电子商务是指利用 Web 提供的通信手段在网上买卖产品或提供服务；广义的电子商务除了以上内容外还包括企业内部的商务活动：如生产、管理、财务等；以及企业间的

商务活动：把买家、卖家、厂家和合作伙伴通过 Internet、Intranet 和 Extranet 连接起来所开展的业务。从最初的电话、电报、电子邮件，到二十多年以前开始的电子数据交换 EDI，都可以说是电子商务的雏形；到今天，电子商务已经延伸到商务的各个方面；人们可以通过网络进行原材料查询、采购、产品展示和订购，再到出货、储运及电子支付等一系列完整的贸易过程。从更广泛意义上来说，未来 Internet 上的活动，都将是电子商务。

要实现完整的电子商务会涉及很多方面，除了买家、卖家外，还要有银行或金融机构、政府机构、认证机构、配送中心等机构的加入才行。由于参与电子商务中的各方在物理上互不谋面，因此整个电子商务过程并不是物理世界商务活动的翻版，网上银行、在线电子支付等条件和数据加密、电子签名等技术在电子商务中发挥着重要的不可或缺的作用。

2．电子商务的类型

可以对电子商务按参与电子商务交易的对象、电子商务交易的商品内容和进行电子商务的企业所使用的网络类型等对电子商务进行不同的分类。

按参与交易的对象分类，电子商务可以分为以下几类：

（1）企业与消费者之间的电子商务（Business to Customer，B2C）。企业与消费者之间的电子商务是人们最熟悉的一种电子商务类型。网上商店利用 Internet 提供的双向交互通信，完成网上购物的过程。这类电子商务主要是借助于 Internet 所开展的在线式销售活动。最近几年随着 Internet 的发展，这类电子商务的发展异军突起。例如，在 Internet 上目前已出现许多大型超级市场，所出售的产品一应俱全，从食品、饮料到电脑、汽车等，几乎包括了所有的消费品。由于这种模式节省了客户和企业双方的时间和空间，大大提高了交易效率，节省了各类不必要的开支，因而这类模式得到了人们的认同，获得了迅速的发展。

（2）企业与企业之间的电子商务（Business to Business，B2B）。两个或若干个有业务联系的公司通过 B2B 模式彼此连接起来，形成网上的虚拟企业圈。例如，企业利用计算机网络向它的供应商进行采购，或利用计算机网络进行付款等。B2B 具有很强的实时商务处理能力，使企业能以一种安全、可靠、简便、快捷的方式进行企业间的商务联系活动。

（3）消费者与消费者之间的电子商务（Customer to Customer，C2C）。C2C 电子商务平台就是通过为买卖双方提供一个在线交易平台，使卖方可以主动提供商品上网拍卖，而买方可以自行选择商品进行竞价。

（4）O2O 即 Online To Offline（在线离线/线上到线下），是指将线下的商务机会与互联网结合，让互联网成为线下交易的平台，这个概念最早来源于美国。O2O 的概念非常广泛，既可涉及到线上，又可涉及到线下,可以称为 O2O。如团购就属于一种典型的 O2O。

14.6 知识管理与商业智能

知识管理是企业信息化发展的高级阶段，而商业智能则是知识管理的实际应用。

14.6.1 知识管理

知识管理是信息化时代重要的管理理论和管理方法，管理大师彼得·德鲁克早在一九六五年即预言："知识将取代土地、劳动、资本与机器设备，成为最重要的生产因素。"在信息化的过程中，知识管理成为构建企业核心竞争力，获得市场竞争优势的有力武器。

知识管理可以定义为：在组织中建构一个人文与技术兼备的知识系统，让组织中的信息与知识，通过获得、创造、分享、整合、记录、存取、更新等过程，实现不断创新。同时，这种创新知识又不断回馈到组织之内，从而使得组织的知识不间断地累积和升华，进而转化为企业的智慧资本。

在信息时代里，知识已成为最主要的财富来源。21 世纪的组织，最有价值的资产是组织内的知识工作者和他们的生产力。而知识工作者就是最有生命力的资产，组织和个人的最重要任务就是对知识进行管理。知识管理将使组织和个人具有更强的竞争实力，并做出更好的决策。

早在 20 世纪 80 年代，斯坦福大学的保罗·罗默教授就曾提出了经济增长四要素理论，其核心思想是把知识作为经济增长最重要的要素，他认为：首先，知识能提高收益；其次，知识需要投资；第三，知识与投资存在良性循环关系，投资促进知识，知识促进投资。在信息化过程中，公司中最大的资产，就是继资本、劳动之后脱颖而出的第三资源，即知识资源。

由于知识是企业最重要的战略性资源，因此，知识管理就成为企业的一个重要的战略任务。在国际化和信息化的大背景下，企业要在激烈的市场竞争中胜出，就要运用集体的智慧提高应变能力和创新能力，即，必须有创造和运用知识的能力，同时，也为企业实现显性知识和隐性知识共享提供新的途径。

人在获取知识的过程中，总是离不开信息，总是在与信息打交道，因此，知识管理的过程也是对信息资源管理的过程。毫无疑问，知识管理应以人为中心，以信息为基础，以知识创新为目标，将知识看作一种可开发资源。简单说，知识管理就是人在企业管理中对其集体的知识与技能的获得与运用的过程。

2．知识管理的工具和手段

知识管理的工具和手段为知识管理提供了实施条件，使得知识管理能够为企业提升竞争

力服务。

知识管理工具是实现知识的生成、编码和转移技术的集合。知识管理工具主体是以计算机为基础的技术集合，同时，也包括传统的知识管理工具，例如，人们经常用的纸和笔等。

（1）知识管理工具的范畴

基于计算机技术的管理工具分为三类，即，数据管理工具、信息管理工具和知识管理工具。从广义的角度看，知识管理工具是以上三类工具的总和，即数据管理工具和信息管理工具都可看作知识管理工具；从狭义的角度看，知识管理工具又区别于其他两类工具，也就是说，数据管理工具和信息管理工具还不是知识管理工具。知识管理工具不仅是数据管理工具和信息管理工具的改进，更是在更高的层次上的发展和创新。这是因为：数据、信息和知识是三个不同层次的事物。数据是基础，是“原生态”。而信息，按照信息论创始人申农的说法，“信息是不确定性的减少”，因此，信息是经过加工处理而具有某种使用价值的数据。知识是高级的信息，是蕴涵更高价值的信息。

数据管理工具的管理对象是数据，手工数据，如销售数据、库存记录、各种台账报表等，基于计算机技术的有数据库、数据仓库、搜索引擎、数据建模工具等。

信息管理工具的管理对象是信息。现在人们经常使用的主要是基于计算机技术的工具，如电子交换系统、决策支持系统、管理信息系统等。

知识管理工具的管理对象是知识，知识管理工具能够帮助人们实现知识管理的自动化或半自动化，如专家系统、知识库等。

（2）知识管理工具的分类

从企业中知识的生命周期来看，知识管理可以分为知识的生成、知识的编码和知识的转移。相应的，知识管理工具也分为三类，即知识的生成、编码和转移工具。

① 用于知识生成的工具。知识的创造对于一个企业来说极端重要，它是企业具有长久生命力的保证。知识的生成包括产生新的想法、发现新的商业模式、发明新的生产流程及对原有知识的整合。企业内部的知识产生有多种模式，如知识的获取、综合、创新等。不同方式的知识产生模式应有不同的工具对其进行支持。使用比较多的用于知识产生的工具有：搜索引擎、数据挖掘技术、用于知识合成的工具、辅助创新知识的工具。

② 用于知识编码的工具。知识在产生出来后，只有通过共享和交流才能发挥其巨大的价值。知识编码则是通过标准的形式表现知识，使知识能够方便地被共享和交流。知识编码的困难在于，知识几乎不能以离散的形式予以表现。知识编码工具可以分为知识仓库和知识地图。

③ 用于知识转移的工具。知识的价值在于流动。许多案例表明，如果不同的部门相互交流各自的经验和知识，那将会产生巨大的效益，因此知识的传播对于提高知识的价值是十分重要的。这个规律适用于组织或个人。在知识流动的过程中，存在许多使知识不能毫无阻力地任意流动的障碍。这些障碍可以分成三类：时间差异、空间差异和社会差异。企业需要根据各种障碍的特点，设计相应的制度和工具，使企业的知识更有效地流动。

（3）对知识管理工具的评价

知识管理工具是企业实施知识管理的物质基础，在企业实施知识管理过程中发挥着重要的作用，它有助于企业知识的获取和累积，有利于企业员工进行知识集成和创新，促进企业的知识共享与利用，最终将增强企业的创新和竞争能力。但是，现有的知识管理工具还存在着不足：

一是功能不完整。作为以知识管理为目的的知识管理工具，还不能很好地支持企业内知识的有效获得、共享与传播，其功能还处于较低的层次，达不到大规模应用的程度；

二是集成度不高。现有的知识管理工具几乎都是针对某一具体任务。分别采用不同方法和技术开发的各类工具，往往会导致知识被分割或隔离；

三是协同性不够。知识管理的一个重要功能是促进企业员工、各部门之间的协同工作，而现有的知识管理工具对这方面的支持是不够的；

四是可重构性差。企业需要从多处引进各种不同的知识管理工具，然后建立一套适合于自己的知识管理系统。这就要求这些工具应具有可重构性，而现有知识管理工具在这方面的性能是欠缺的。

14.6.2 商业智能

商业智能（Business Intelligence，BI）是企业对商业数据的搜集、管理和分析的系统过程，目的是使企业的各级决策者获得知识或洞察力，帮助他们做出对企业更有利的决策。

早在 20 世纪 90 年代末，商业智能技术就被一家计算机权威杂志评选为未来几年最具影响力的 IT 技术之一。但商业智能技术并不是基础技术或者产品技术，它是数据仓库、联机分析处理 OLAP 和数据挖掘等相关技术走向商业应用后形成的一种应用技术。

商业智能系统主要实现将原始业务数据转换为企业决策信息的过程。与一般的信息系统不同，它在处理海量数据、数据分析和信息展现等多个方面都具有突出性能。

商业智能系统主要包括数据预处理、建立数据仓库、数据分析及数据展现 4 个主要阶

段。数据预处理是整合企业原始数据的第一步，它包括数据的抽取、转换和装载三个过程。建立数据仓库则是处理海量数据的基础。数据分析是体现系统智能的关键，一般采用联机分析处理和数据挖掘两大技术。联机分析处理不仅进行数据汇总/聚集，同时还提供切片、切块、下钻、上卷和旋转等数据分析功能，用户可以方便地对海量数据进行多维分析。数据挖掘的目标则是挖掘数据背后隐藏的知识，通过关联分析、聚类和分类等方法建立分析模型，预测企业未来发展趋势和将要面临的问题。在海量数据和分析手段增多的情况下，数据展现则主要保障系统分析结果的可视化。

一般认为，数据仓库、OLAP 和数据挖掘技术是商业智能的三大组成部分。有关这方面的详细知识，请阅读 2.8 节和 2.9 节。

14.7 业务流程重组

1990 年，美国迈可·哈默（Michael Hammer）博士首先提出了业务流程重组（BusinessProcess Reengineering，BPR）的概念。哈默认为，BPR 是对业务流程进行根本反思，要对其进行重新设计，从而使得衡量现代企业绩效的关键指标，如成本、质量、服务和速度等得到奇迹般的改善。

哈默对 BPR 的定义较全面地反映了业务流程重组的本质特征，这就是以业务流程为核心、对业务流程进行根本反思、彻底重新设计业务流程，使企业发生跨越式的发展。

以往的企业管理的变革和改进都是基于传统的职能分工理论，因此，其效果往往不佳，造成“膨胀——精简——再膨胀——再精简”的恶性循环。而 BPR 理论从根本上打破了职能分工理论的局限，把业务流程作为基础和核心，作为管理变革的对象。

所谓业务流程，是指为了完成某一目标或任务而进行的一系列跨越时空的逻辑相关活动的有序集合。通过考察业务流程的发生、发展和终结，确定、描述、分析、分解整个业务流程，重构与业务流程相匹配的企业运行机制和组织机构，实现对企业全流程的有效管理和控制，能够使企业真正着眼于流程的结果，消除传统管理中只注重某一环节而无人负责全流程的弊端。

2．BPR 的内容

BPR 彻底地抛弃了传统的职能管理模式，对员工角色、管理观念、组织机构带来巨大变革。

BPR 强调 4 个核心内容，即根本性、彻底性、戏剧性和流程。

（1）根本性。BPR 强调要进行根本性的再思考，各方面都要关注流程，因为它是企业的核心问题。

（2）彻底性。彻底性是要求对 BPR 进行追根溯源，对既定存在的事物不是进行小修小补，而是要进行彻底的改造。例如，BPR 彻底改变了员工的地位，员工不再是被动的命令执行者，而是被赋予足够的权力和负有与之匹配的责任的流程主人。BPR 鼓励员工在权力范围内自己决策，高层管理人员只是进行必要的指导和协调。同时，BPR 对员工的素质要求更高了，要求员工能胜任多层次的工作。在工作开始阶段，BPR 更强调观念教育而非技能培训。

（3）戏剧性。戏剧性表明 BPR 完全抛弃传统管理观念，不是追求稍有改善，而是充分强调结果的满意度；在衡量员工的业绩标准上，BPR 注重员工创造的价值，倡导创新，这与传统的“多劳多得”有本质的不同；BPR 彻底否定企业中大量的低效劳动和无效监督控制；另外，员工的晋升是靠综合能力而非一时的工作业绩。

（4）流程。BPR 主张不是企业的业务流程的简单改善，而是要创建全新的组织机构，打破以专业分工理论为基础的职能部门管理框架，建立以流程工作小组为单元的管理模式，形成扁平式管理机构，大大压缩了管理层级，不但提高了管理效率，增强组织柔性，而且节约了中间管理层所产生的巨额费用。

3．BPR 的作用

BPR 的实施将使企业发生根本性的变革，增强企业的活力，给企业带来巨大的经济效益。

（1）BPR 的实施使企业更贴近市场。企业为了提高顾客满意度，将主动进行市场调研，预测市场需求，及时掌握市场走向。同时，管理层级的压缩，使高层管理人员与第一线业务人员和顾客之间缩短了距离，能够直接获取一线人员的意见，在第一时间感知顾客对产品的反应和新的需求，从而及时调整经营决策。

（2）BPR 使生产成本成倍压缩。BPR 吸收了先进的管理理论和技术，利用并行工程等思想，可大幅度压缩产品的开发周期，加快产品的更新换代频率；同时，BPR 以业务流程为核心，彻底消除了传统管理模式中人为因素的干扰，减少了中间环节，降低协调、控制成本。另外，BPR 否定了传统管理模式中的多余监控，从而减少管理层级，使得管理成本大大降低。

（3）BPR 使产品质量得到全面提升。BPR 将全面质量管理思想贯穿于整个流程中，从市场调研阶段开始就把产品质量作为重要指标来监控。BPR 考虑了市场反馈信息的作用，

采用柔性制造系统等先进理论开发、生产产品，可以最大限度地保证产品质量的全面提升。

（4）服务质量更趋完美。由于 BPR 彻底抛弃了职能分工的思想，确立了以流程为核心的观念，因此，企业所有员工都把满足顾客最大需求作为自己工作的首要目标。在工作方式上，员工由被动服务变为主动服务，传统管理模式下企业管理人员的许多监管工作已变得多余，员工工作的主动性和自觉性大大提高，企业的整体服务水平上升了一个层次，服务质量更趋完美。

由于 BPR 是企业管理的一场革命，通过根本性的变革建立以业务流程为核心的运营机制和组织机构，可能会给企业带来巨大的震荡。所以，企业实施 BPR 在有可能获得巨大效益的同时，也承受了较大的风险，一旦 BPR 失败，那将会给企业带来难以挽回的后果。

4．BPR 遵循的原则

BPR 在追求顾客满意度和员工追求自我价值实现的流程中带来降低成本的结果，从而达到效率和效益改善的目的。BPR 在注重结果的同时，更注重流程的实现，并非以短期利润最大化为追求目标，而是追求企业能够持续发展的能力，而为达此目标必须坚持流程中心原则、团队式管理原则和顾客导向原则。

（1）流程中心原则企业业务流程，特别是关键业务流程总是在最大程度上体现了企业的总体目标和用户价

值，因而，流程式管理模式最主要的特点是企业的一切工作都是围绕结果而不是围绕工序或分工。因而，BPR 注重的是业务流程整体最优，通过理顺和优化业务流程，使得业务流程中每一个环节上的活动尽可能实现最大化增值，尽可能减少无效的或不增值的活动，并从整体最优的目标出发，设计和优化业务流程中的各项活动，消除本位主义和利益分散主义。

（2）团队管理原则在流程式管理模式下，企业的组织结构必须服从业务流程，使组织扁平化，而要做到

这些，就必须坚持另一个重要原则——团队式管理原则。在 BPR 中，首先是设计、重组业务流程，而后依据业务流程建立或改造企业组织，尽量消除或弱化“中间层”。这不仅降低了管理费用和成本，更重要的是提高了组织的运转效率及对市场的反应速度。

员工素质的提高是 BPR 取得成功的前提条件。在以流程为中心的管理模式下，员工的积极性和主动性必然高于以往，这是因为他们不再满足于从事单调、简单的工作，而是承担一定的责任，有一定的权力，在工作中能充分发挥自我，有成就感。而要使员工的积极性和主动性能够得以长期保持，最有效的途径就是组成团队。

（3）客户导向原则

BPR 理论的出现是与世界经济的发展，社会环境的变化，科学技术的进步，新技术、新方法的推广应用，尤其是信息技术的迅速发展所分不开的，而信息技术的发展才能保证顾客导向原则贯彻到底。因此，利用信息技术能够有效地帮助企业 BPR 得以很好地实施，例如，利用建模的信息工具可以重新设计经营流程；采用计算机网络、数据库和多媒体等技术建立的信息网络，能够加快信息传递，实现信息共享，其结果是将传统的串行工作方式变为并行工作方式，将企业组织结构由垂直型变为水平型，使企业成为协同工作的组织，使得企业的业务流程，特别是关键业务流程与市场接通，与顾客接通。

另一方面，科学技术的发展和管理模式的日臻完善，也为 BPR 创造了条件。例如，在加工制造行业，柔性制造系统是一种能高效率、高质量地进行多品种、中小批量生产的自动化加工系统，利用它，企业可以快速响应市场变化，满足顾客多样化和个性化需求。一些现代管理模式，如精密生产、准时制造和全面质量管理等，提倡以顾客为中心，以及坚持增值第一和质量第一的理念，都体现了顾客导向的原则。

第 15 章：基于中间件的开发

20 世纪 90 年代初，C/S 计算模式成为主流，将数据统一存储在数据服务器上，而有关的业务逻辑都在客户端实现。但是，这种两层结构的模式由于服务器依赖于特定的供应商，数据存取受到限制，再加上难以扩展到广域网或互联网等原因，极大地阻碍着计算机软件系统的发展。有人提出将客户端的业务逻辑独立出来，形成第三层。在这种三层结构中，客户端仅仅是处理图形用户界面，在设计和实现时需要开发的，仅是在应用服务器上的业务逻辑部分的软件。

随着 Internet 及 WWW 的出现，计算机的应用范围更为广阔，许多应用程序需在网络环境的异构平台上运行。在这种分布异构环境中，通常存在多种硬件系统平台（例如 PC、工作站、小型机等），在这些硬件平台上又存在各种各样的系统软件（例如不同的操作系统、数据库、语言编译器等），以及多种风格的用户界面，这些硬件系统平台还可能采用不同的网络协议和网络架构连接。如何把这些系统集成起来并开发新的应用是一个非常现实而困难的问题。为了解决这个问题，出现了处于系统软件和应用软件之间的中间件。它使设计者集中设计与应用有关的部分，大大简化了设计和维护 工作。

15.1 中间件技术

中间件（middleware）是基础软件的一大类，属于可复用软件的范畴。顾名思义，中间件处在操作系统、网络和数据库之上，应用软件的下层（如图 15-1 所示），也有人认为它应该属于操作系统中的一部分。

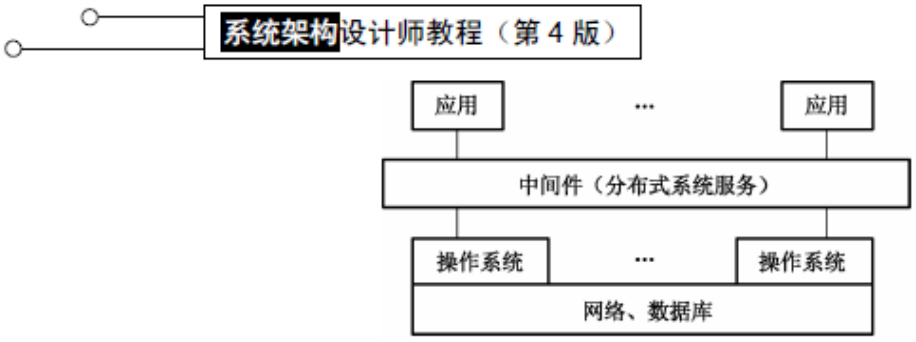


图 15-1　中间件图示

15.1.1 中间件的概念

中间件从诞生到现在，虽然仅有 10 多年时间，但发展极其迅速，是有史以来发展最快的软件产品，但在技术上还处于成长阶段，还没有统一的标准和模型，通常都是用 C++ 语言以面向对象的技术来实现的，但是它的特性已超出面向对象的表达能力，由于它属于可重用构件，目前趋向于用构件技术来实现。然而，中间件要涉及软件的所有标准、规范和技术，它有更多的内涵，因为它包括平台功能，自身具有自治性、自主性、隔离性、社会化、激发性、主动性、并发性、认识能力等特性，是近似于 Agent（代理）的结构。

目前很难给中间件一个严格的定义，国际上各家机构都有不同的定义，如 IDC 对中间件给出的定义是：中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。这些服务程序或软件具有标准的程序接口和协议。针对不同的操作系统和硬件平台，它们可以有符合接口和协议规范的多种实现。中间件为处于其上层的应用软件提供运行与开发的环境，帮助用户灵活、高效地开发和集成复杂的应用软件。中间件应具有如下的一些特点：

满足大量应用的需要；

运行于多种硬件和 OS 平台；

支持分布计算，提供跨网络、硬件和 OS 平台的透明性的应用或服务的交互；

支持标准的协议；

支持标准的接口。

世界著名的咨询机构 Standish Group 在一份研究报告中归纳了中间件的十大优越性：

缩短应用的开发周期；

节约应用的开发成本；

减少系统初期的建设成本；

降低应用开发的失败率；

保护已有的投资；

简化应用集成；

减少维护费用；

提高应用的开发质量；

保证技术进步的连续性；

增强应用的生命力。

具体来说，首先，中间件屏蔽了底层操作系统的复杂性，使程序开发人员面对一个简单而统一的开发环境，减少了程序设计的复杂性，将注意力集中在自己的业务上，不必再为程序在不同系统软件上的移植而重复工作，从而大大减少了技术上的负担。

中间件带给应用系统的，不只是开发的简便、开发周期的缩短，也有系统的维护、运行和管理的工作量的减少，还减少了计算机总体费用的投入。Standish 的调查报告显示，由于采用了中间件技术，应用系统的总建设费用可以减少 50%左右。在网络经济、电子商务大发展的今天，从中间件获得利益的不只是 IT 厂商，IT 用户也同样是赢家，并且是更有把握的赢家。

其次，中间件作为新层次的基础软件，其重要作用是将不同时期、在不同操作系统上开发的应用软件集成起来，彼此无缝地整体协调工作，这是操作系统、数据库管理系统本身做不了的。中间件的这一作用，使得在技术不断发展之后，人们以往在应用软件上的劳动成果仍然物有所用，节约了大量的人力、财力投入。

最后，由于标准接口对于可移植性和标准协议对于互操作性的重要性，中间件已成为许多标准化工作的主要部分。对于应用软件开发，中间件远比操作系统和网络服务更为重要，中间件提供的程序接口定义了一个相对稳定的高层应用环境，不管底层的计算机硬件和系统软件怎样更新换代，只要将中间件升级更新，并保持中间件对外的接口定义不变，应用软件

几乎不需任何修改，从而节省了企业在应用软件开发和维护中的重大投资。

15.1.2 中间件的分类

好比一个大型城市的交通系统，将网络看作市区马路，通过交通工具（如汽车）实现通信，每分钟将有数以万辆车在马路上行驶，如果没有相应的交通设施和管理规划，城市将会乱成一团，发生各种交通事故，中间件系统就相当于这些配套的交通设施。按照中间件在分布式系统中承担的职责不同，可以划分以下几类中间件产品。

（1）通信处理（消息）中间件。正如，安装红绿灯，设立交通管理机构，制定出交通规则，才能保证道路交通畅通一样，在分布式系统中，人们要建网和制定出通信协议，以保证系统能在不同平台之间通信，实现分布式系统中可靠的、高效的、实时的跨平台数据传输，这类中间件称为消息中间件，也是市面上销售额最大的中间件产品，目前主要产品有 BEA 的 eLink、IBM 的 MQSeries、TongLINK 等。实际上，一般的网络操作系统如 Windows 已包含了其部分功能。

（2）事务处理（交易）中间件。正如城市交通中要运行各种运载汽车，以此来完成日常的运载，同时随时监视汽车运行，在出现故障时及时排堵保畅。在分布式事务处理系统中，经常要处理大量事务，特别是 OLTP 中，每项事务常常要多台服务器上的程序按顺序协调完成，一旦中间发生某种故障，不但要完成恢复工作，而且要自动切换系统，达到系统永不停机，实现高可靠性运行。要使大量事务在多台应用服务器上能实时并发运行，并进行负载平衡的调度，实现与昂贵的可靠性机和大型计算机系统同等的功能，为了实现这个目标，要求中间件系统具有监视和调度整个系统的功能。BEA 的 Tuxedo 由此而著名，它
成为增长率最高的厂商。

（3）数据存取管理中间件。在分布式系统中，重要的数据都集中存放在数据服务器中，它们可以是关系型的、复合文档型、具有各种存放格式的多媒体型，或者是经过加密或压缩存放的，该中间件将为在网络上虚拟缓冲存取、格式转换、解压等带来方便。

（4）Web 服务器中间件。浏览器图形用户界面已成为公认规范，然而它的会话能力差、不擅长做数据写入、受 HTTP 协议的限制等，就必须进行修改和扩充，形成了 Web 服务器中间件，如 SilverStream 公司的产品。

（5）安全中间件。一些军事、政府和商务部门上网的最大障碍是安全保密问题，而且不能使用国外提供的安全措施（如防火墙、加密、认证等），必须用国产产品。产生不安全

因素是由操作系统引起的，但必须要用中间件去解决，以适应灵活多变的要求。

（6）跨平台和架构的中间件。当前开发大型应用软件通常采用基于架构和构件技术，在分布式系统中，还需要集成各节点上的不同系统平台上的构件或新老版本的构件，由此产生了架构中间件。功能最强的是 CORBA，可以跨任意平台，但是过于庞大；JavaBeans 较灵活简单，很适合于做浏览器，但运行效率有待改善；COM+模型主要适合 Windows 平台，在桌面系统已广泛使用。由于国内新建系统多基于 UNIX（包括 Linux）和 Windows，因此，针对这两个平台建立相应的中间件市场相对要大得多。

（7）专用平台中间件。为特定应用领域设计领域参考模式，建立相应架构，配置相应的构件库和中间件，为应用服务器开发和运行特定领域的关键任务（如电子商务、网站等）。

（8）网络中间件。它包括网管、接入、网络测试、虚拟社区、虚拟缓冲等，也是当前最热门的研发项目。

15.1.3 中间件产品介绍

本节介绍主流的中间件产品 IBM MQSeries 和 BEA Tuxedo。

1．IBM MQSeries

IBM 公司的 MQSeries 是 IBM 的消息处理中间件。MQSeries 提供一个具有工业标准、安全、可靠的消息传输系统，它用于控制和管理一个集成的系统，使得组成这个系统的多个分支应用（模块）之间通过传递消息完成整个工作流程。MQSeries 基本由一个信息传输系统和一个应用程序接口组成，其资源是消息和队列。

MQSeries 的关键功能之一是确保信息可靠传输，即使在网络通信不可靠或出现异常时也能保证信息的传输。MQSeries 的异步消息处理技术能够保证当网络或者通信应用程序本身处于“忙”状态或发生故障时，系统之间的信息不会丢失，也不会阻塞。这样的可靠性是非常关键的，否则大量的金钱和客户信誉就会面临极大的损害。

同时，MQSeries 是灵活的应用程序通信方案。MQSeries 支持所有的主要计算平台和通信模式，也能够支持先进的技术如（Internet 和 Java），拥有连接至主要产品（如 LotusNotes 和 SAP/R3 等）的接口。

2．BEA Tuxedo

BEA 公司的 Tuxedo 作为电子商务交易平台，属于交易中间件。它允许客户机和服务器参与一个涉及多个数据库协调更新的交易，并能够确保数据的完整性。BEA Tuxedo 一个特

色功能是能够保证对电子商务应用系统的不间断访问。它可以对系统构件进行持续的监视，查看是否有应用系统、交易、网络及硬件的故障。一旦出现故障，BEA Tuxedo 会从逻辑上把故障构件排除，然后进行必要的恢复性步骤。

BEA Tuxedo 根据系统的负载指示，自动开启和关闭应用服务，可以均衡所有可用系统的负载，以满足对应用系统的高强度使用需求。借助 DDR（数据依赖路由），BEA Tuxedo 可按照消息的上下文来选择消息路由。其交易队列功能，可使分布式应用系统以异步“少连接”方式协同工作。

BEA Tuxedo 的 LLE 安全机制可确保用户数据的保密性，应用/交易管理接口（ATMI）为 50 多种硬件平台和操作系统提供了一致的应用编程接口。BEA Tuxedo 基于网络的图形界面管理可以简化对电子商务的管理，为建立和部署电子商务应用系统提供了端到端的电子商务交易平台。

15.2 应用服务器技术

Web 应用开发大致经历了三个阶段。在第一阶段，大家都使用 Web 服务器提供的服务器扩展接口，使用 C 或者 Perl 等语言进行开发，例如 CGI、API 等。这种方式可以让开发者自由地处理各种不同的 Web 请求，动态地产生响应页面，实现各种复杂的 Web 系统要求。但是，这种开发方式的主要问题是对开发者的素质要求很高，往往需要懂得底层的编程方法，了解 HTTP 协议，此外，这种系统的调试也相当困难。

在第二阶段，大家开始使用一些服务器端的脚本语言进行开发，主要包括 ASP、PHP、Livewire 等。其实现方法实质上是在 Web 服务器端放入一个通用的脚本语言解释器，负责解释各种不同的脚本语言文件。这种方法的首要优点是简化了开发流程，使 Web 系统的开发不再是计算机专业人员的工作。此外，由于这些语言普遍采用在 HTML 中嵌入脚本的方式，方便实际开发中的美工和编程人员的分段配合。对于某些语言，由于提供了多种平台下的解释器，所以应用系统具有了一定意义上的跨平台性。但是，这种开发方式的主要问题是系统的可扩展性不够好，系统一旦比较繁忙，就缺乏有效的手段进行扩充。此外，从一个挑剔者的眼光来看，这种方式不利于各种提高性能的算法的实施，不能提供高可用性的效果，集成效果也会比较差。

为了解决这些问题，出现了一个新的 Web 应用开发方法，也就是应用服务器的方式。目前，应用服务器已经成为电子商务应用中一种非常关键的中间件技术。如今，各大主要软

件厂商纷纷将应用服务器作为其电子商务平台的基础，如 IBM 的 Websphere，Oracle 的 Internet 应用服务器，Sybase 的 Enterprise 应用服务器等。本节将阐述应用服务器的概念、相关技术及发展方向，并就目前主流的应用服务器产品进行简单的介绍。

15.2.1 应用服务器的概念

应用服务器是在当今 Internet 上企业级应用迅速发展、电子商务应用出现并快速膨胀的需求下产生的一种新技术，通过它能将一个企业的商务活动安全有效地实施到 Internet 上，实现电子商务。它并非一种传统意义上的软件，而是一个可以提供通过 Internet 来实施电子商务的平台。在分布式、多层结构及基于构件和服务器端程序设计的企业级应用开发中，它提供的是一个开发、部署、运行和管理、维护的平台。它可以提供软件“集群” 的功能，因而可以让多个不同的、异构服务器协同工作、相互备份，以满足企业级应用所需要的可用性、高性能、可靠性和可伸缩性等。

故而，从某种意义上说，应用服务器提供了一个“企业级应用的操作系统”。实现 J2EE 规范的应用服务器称为 J2EE 应用服务器。

现代社会商机稍纵即逝，电子商务应用要求能很快地开发出功能强大的系统。应用服务器可以帮助企业快速架构一个基 Internet 的电子商务系统，而且拥有极高的稳定性、可扩展性和安全性。它能够：

（1）更合理地分工企业级应用开发，加快应用的开发速度，减少应用的开发量。应用服务器将系统功能与业务功能分开，使得编程人员能够集中精力在业务功能上，在系统内建立/部署的构件越来越多，并且为分布式架构的时候，系统功能必将变得越来越复杂；而与此同时，对可靠性（负载均衡、容错和故障恢复）的需求也会越来越高。开发人员只关注编码业务方面的功能，对系统一级的功能并没有多少兴趣。因为系统级底层功能的实现，一般需要非常复杂的专业技能，因此对功能实现的合理分离可以允许技能的优化。

在应用服务器上开发采用的模块化方法，提供了大量的可重用模块。一个新的系统可以通过组合一些现成的框架和模块，再加上一定的开发来快速完成。而新开发出的代码又可作为今后重复利用的模块，这一点对于降低开发成本，提高开发速度是非常重要的。

另外，为了便于开发，有些应用服务器还提供开发版的服务器，以便进行各种调试工作。应用服务器一般还提供集成开发环境，将本地编辑、上传、项目管理和调试工具等集中在一起，使开发工作在一个界面内全部完成。还有一些开发环境同时提供后台系统的开发环境，

以便同时进行开发管理。此外，还有一些产品内置一些代码的自动生成器，数据库设计辅助工具等，例如 ORM（ObjectRelation Mapping，对象关系映射）等，这些都有效地提高了开发速度，减少了应用开发量。

（2）应用设计、开发、部署、运行、管理、维护的平台。应用服务器既是应用开发的平台，包括表示层、应用层和数据层的设计模式和编程环境；同时又是多层结构应用的部署、运行平台，对多层结构应用进行配置、启动、监控、调整，并在开发的不同阶段承担不同的职责。

设计：应用服务器完成底层通信、服务，并屏蔽掉复杂的底层技术细节，向用户提供结构简单、功能完善的编程接口，让用户可以专心于商务逻辑的设计。

开发：应用服务器提供了完全开放的编程语言和应用接口，用户可以用任何自己习惯的开发工具来工作。另外应用服务器自己也提供快速开发的工具和手段，帮助用户提高开发效率。

部署：应用服务器可以部署在任何硬件平台、任何操作系统上，而且可以分布在异构网络中，应用服务器帮助用户在复杂的网络环境中配置系统参数，使系统发挥最大的性能，拥有最好的稳定可靠性。

运行：应用服务器采用的是开放技术标准，它提供了一个完整的标准实现，即提供了系统的运行环境，任何基于同样标准的系统都能很好地运行于这个环境中。在运行中提供应用系统的名字解析、路由选择、负载平衡、事务控制等服务，并提供系统容错、修复、迁移、升级扩展等功能。

管理：应用服务器让用户通过图形化的界面方便地管理自己的资源，而且在系统运行时也能动态监控和管理。

（3）使得应用与底层平台无关，便于商业逻辑的实现与扩展。一个好的应用服务器通过提供对操作系统/数据库平台的广泛支持，或采用平台无关技术，如 J2EE，从而能够做到让应用独立于操作系统/数据库平台。显然，它可以确保企业应用具备很好的移植性并保护了企业在应用和开发技能上的投资。

另外一个方面，在激烈的市场竞争条件下，企业的商业逻辑不可能一成不变，而随着生产经营的拓展，首先需要解决的一个问题就是将按需地对现有的业务系统，方便地进行扩充和升级。应用服务器技术可以很好地解决这个问题，因为它采用了三层结构体系，应用服务器将业务流程单独作为一层，客户可以根据自己的商业逻辑来专心设计这一层。应用服务器能提供这种设计能力，当客户业务扩展时，只需专注于改进中间层的设计，原系统就能平滑

方便地升级。

（4）为企业应用提供现成的、稳定而强健的、灵活的、成熟的基础架构。在构建电子商务应用的竞争中，许多企业已经没有时间去从容地“千锤百炼”一个电子商务架构体系。通过应用服务器，立刻就可以拥有一个成熟的架构，包括基础平台、标准、应用开发工具和预制构件。

另外一方面，随着经济全球化的步伐加快，许多企业的业务服务于全球，计算机业务系统需要提供 24 小时不间断的服务，系统在大负荷量和长时间运转情况下的稳定性至关重要。应用服务器通过分布式体系来保障这一点，表现为：

当系统处理能力不够时，可以通过简单地增加硬件来解决；

动态调整不同主机间的负载可以最大地利用系统资源，同时提高单机的稳定性；

当系统中的某台机器出现故障时，它的工作可由其他机器来承担，不会影响系统整体的运行，即无单点故障。

希赛教育专家提示：目前市面上的应用服务器的解决方案基本都具备了这些作用，因而企业在选购应用服务器产品的时候，不能简单地判断优劣，而需要先充分了解自己的需求到底是什么，然后在各个主要技术问题上，确定适合自己的解决方案，最后寻找使用这些解决方案的产品来完成自己的系统。

15.2.2 主要的应用服务器

本节介绍五种主要的应用服务器产品。

1．BEA WebLogic

BEA WebLogic 作为新一代基于 Java 的 Web 应用服务器，是一款满足 Web 站点对性能和可靠性要求很高的产品。在提供传统的应用服务器功能的同时，还针对当今的 Internet 技术和 Java 技术提供了众多功能，它符合最新 Java 标准。安装 WebLogic 非常容易。

WebLogic EJB Deployer Tool 提供了对管理多个 EJB.jar 文件和配置 WebLogic Server 部署特性和资源的控制。Deployer Tool 支持两级 EJB 部署的合法性检测，它自动地检查特性和引用，以确保它们包含正确的值，并检验关键 EJB 所需的类是否符合 EJB1.1 规范。

名为 WebLogic Zero Administration Client（ZAC）发布向导的图形实用程序使用户可以创建、发布和管理包括应用程序、小程序或 Java 代码库的软件包。ZAC 使用户可以开发客户端 Java 应用并将这些应用打包分发。

2．IBM WebSphere

IBM 的 WebSphere 强调其在应用开发（WebSphere Studio 和 VisualAge for Java）、数据库（DB2）和消息服务（MQseries）的集成性。这些产品构成了该公司总体电子商务产品战略的基础。WebSphere 以对多种平台的支持和符合最新的 Java 标准，提供了开发电子商务应用的可靠平台。

WebSphere 安装简单易行。对 WebSphere 服务器及它运行的应用的控制是在 WebSphere 高级管理控制台中执行的。由于 WebSphere 可以运行多个服务器，因此，用户必须从控制台分别启动每一个服务器进程。如果必须重新引导系统的话，WebSphere 可以记住目前每个不同服务器的状态并自动地重新启动运行的服务器。

IBM 提供了像 WebSphereStudio 和 VisualAgeforJava 这类专为开发基于 Java 应用而设计的其他产品。WebSphere 的高级版和企业版在发送时附带了 IBM 的 DB2 数据库产品和 SecureWay 轻型目录访问协议服务器。企业版包括用于连接到外部数据库、CICS、IMS 或 MQSeries 应用的一个构件代理应用适配器。

3．SUN iPlanet

作为 SUN 与 Netscape 联盟产物的 iPlanet 公司生产的 iPlanet 应用服务器满足最新 J2EE 规范的要求，并通过了全套 J2EE 证书测试套件的测试。iPlanet 应用服务器的基本核心服务包括事务监控器、多负载平衡选项、对集群和故障转移全面的支持、集成的 XML 解析器和可扩展格式底稿语言转换（XLST）引擎，以及对国际化的全面支持。包括 Directory Server、Web Server 和用于 EAI（Enterprise Application Integration，企业应用集成）的另一些附件在内的其他 iPlanet 产品之间实现了紧密的集成。

IPlanet Application Deployment 工具是基于 Java 的程序，它可以指导用户完成一个应用的部署过程。iAS 提供了一款与 WebGainStudio、InspireJBuilder、IBMVisualAge 及 Java 企业版的 SUNForte 这类第三方工具集成在一起的独立产品 iPlanet Application Builder。

4．Oracle Internet ApplicationServer

毫无疑问，Oracle 公司的数据库产品是多种平台上的市场领先产品，凭借这种得天独厚的优势，Oracle 的 Internet Application Server（iAS）与其余 Oracle 产品实现了相互集成，例如，可以利用 Oracle iAS 向 Web 部署任何基于 OracleForms 应用的 Oracle Forms Service。

Oracle 利用一些扩展的 ApacheWeb Server 作为进入 Oracle iAS 的入口点。iAS 管理器是配置和管理应用的工具，提供了综合操作各种系统管理功能的统一界面。Oracle 为

Apache 开发了插件盒模块来处理 Java 应用程序、Perl 程序、PL/SQL 程序及 SSL 上的安全网页。插件盒模块是一个共享库，可以实现程序逻辑访问。插件盒模块中可运行一个或多个插件盒实例，包括 PL/SQL 插件盒、Jweb 插件盒、LiveHTML 插件盒、Perl 插件盒、C 插件盒、ODBC 插件盒等。

iAS 允许开发基于 CORBA 对象的应用，通信协议采用 IIOP。iAS 支持以下两种应用模式：CORBA 应用和 EJB 应用。这两种模式都允许不同的 CORBA 客户访问。iAS 企业版配置了 Oracle Portal。Oracle Portal 提供了部署企业信息用户所需的工具。

Oracle iAS 是该公司将应用推向 Web 战略的关键组成部分。Oracle 的客户可以比较容易地将他们的 Oracle Forms 和 Oracle Reports 放到 Web 上运行。但是，Oracle iAS 价格也比较昂贵。

5．Sybase Enterprise ApplicationServer

Sybase Enterprise Application Serve（ rEAServer ）将 Sybase 的 JaguarCTS 和 PowerDynamo 紧密集成并加以发展，是同时实现 Web 联机事务处理（WebOLTP）和动态信息发布的企业级应用服务器平台。它对各种工业标准提供广泛的支持，符合基于构件的多层架构，是支持所有主要构件模型的应用服务器产品，并且在它的最新版本中加强了对 PowerBuilder 构件和 EJB 的深层支持。这样，用户可以运用它提供的灵活的开发能力，充分利用多样化的计算环境，建立更加高效的企业 Web 应用系统。

EAServer 支持多种构件模型，同一应用中可以结合使用各种构件，支持标准脚本语言和任意客户类型，集成了 PowerSite 开发环境，使 Web 应用开发和提交方便快捷。除了优良的性能之外，EAServer 还支持多种数据库访问方式，给用户提供了可靠的安全性。

15.3 J2EE

J2EE 是针对 Web Service、业务对象、数据访问和消息报传送的一组规范。这组应用编程接口确定了 Web 应用与驻留它们的服务器之间的通信方式。J2EE 注重两件事，一是建立标准，使 Web 应用的部署与服务器无关；二是使服务器能控制构件的生命周期和其他资源，以便能够处理扩展、并发、事务处理管理和安全性问题。

J2EE 规范定义了以下几种构件：应用客户端构件、EJB 构件、Servlets 和 JSP、Applet 构件。J2EE 采用的是多层分布式应用模型，意味着应用逻辑将根据功能分成几个部分， 用户可以在相同或不同的服务器上安装不同应用构件组成的 J2EE 应用。这些层次可以参 见图

15-2。

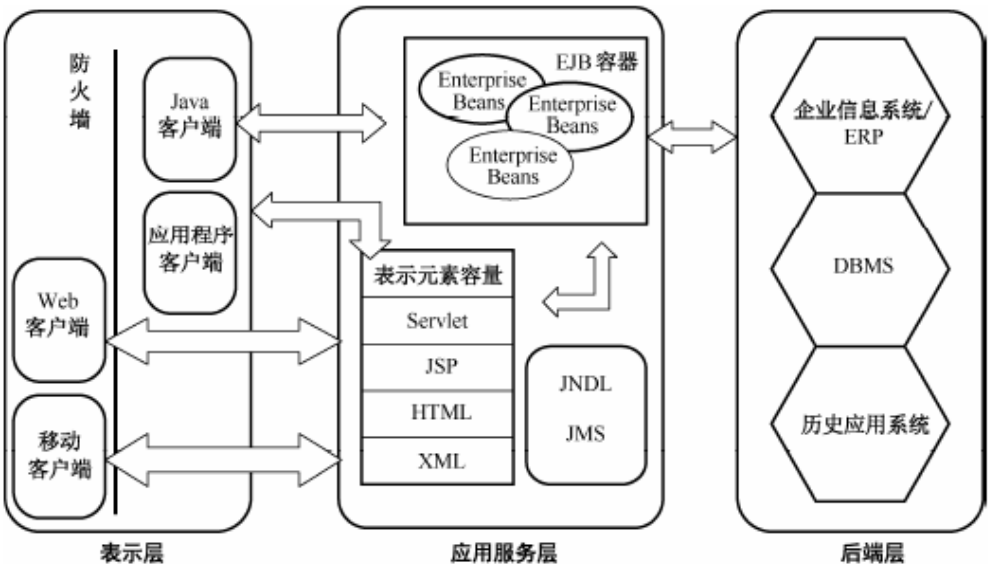


图 15-2　J2EE 多层分布式应用模型图

15.3.1 表示层

J2EE 客户端可以基于 Web，也可以基于 Java。在 HTML、Javascript、XML 等技术的帮助下，Web 浏览器可以支持强大、快速的用户界面。实际上，如果 HTML 足以捕获和显示应用所需的信息，则 HTML 为首选；如果 HTML 不足以达到此目的，则应该由客户端执行必要的捕获和操作。无论是 Applet 还是独立的 Java 程序，都可提供更丰富的图形用户界面。Applet 还可以与中层通信，从而进一步加强程序控制和系统灵活性。

分布式企业应用可以同时包括多种客户端，并且这些客户端都可以访问相同的业务逻辑。如图 15-3 所示：当客户端是 HTML 时，JSP/Servlet 组合将成为能实现业务目标的真正客户端。当客户端是 Java 程序或基于 COM 程序时，它可以直接访问业务逻辑

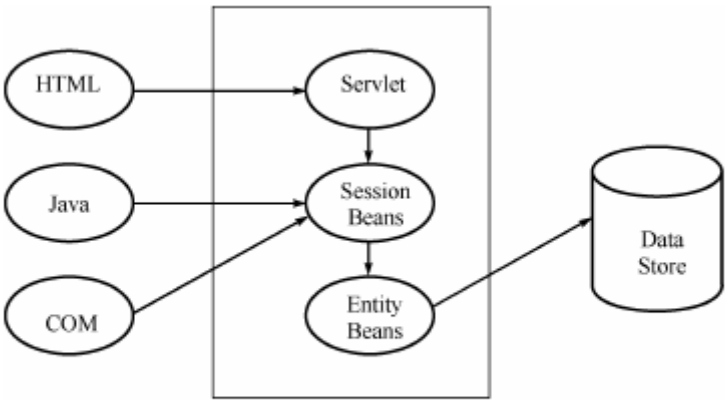


图 15-3　J2EE 多客户端应用图示

15.3.2 应用服务层

一般情况下，应用服务层包含表示层请求的表示逻辑和业务逻辑。表示层由显示 HTML 页面的 JSP 页面和 Servlets 实现。业务逻辑通过 RMI 对象和 EJB 实现。EJB 依靠容器来实现事务处理、生命周期和状态管理、资源池、安全等问题，容器是 EJB 运行的环境。

1．Servlet

Java Servlets 是指可以扩展 Web 服务器功能的程序。Servlet 从客户端接受请求，动态生成响应，然后将包含 HTML 或 XML 文档的请求发送给客户端。Servlet 类似于 CGI（公共网关接口），但 Servlet 使用 Java 类和流，更易于编写；由于 Servlet 可编译为 Java 字节码，在运行时，Servlet 例程驻留在内存中，每一个用户请求都生成一条新线程，故而它们的执行速度也更快。

2．JSP

JSP 页面是基于文本的 Servlet 开发方式。JSP 页面具有 Servlet 的所有优点，如果与 JavaBeans 类结合在一起，可以容易地将内容和显示逻辑分开。这使得无须了解 Java 代码就能更新页面的外观，更新 Java Beans 类的人也无须深入了解 Web 页面的设计。相对 CGI 而言，由于 CGI 依赖于平台，消耗资源更多，而且程序不能容易地访问参数数据等缺点，故而 JSP 页面和 Servlet 都比 CGI 应用广泛。

用户可以使用带 Java Beans 类的 JSP 页面定义 Web 模板，以便建立由外观相似的页面组成的 Web 站点，而 Java Beans 类负责组织数据。用户还可以借助标记和脚本将内容与应用逻辑捆绑在一起，或是嵌入一些 Java 小应用程序来实现一些简单的 Web 应用。

3．EJB

EJB 构件用于封装业务逻辑，使开发人员无须再担心数据访问、事务处理支持、安全性、高速缓存和迸发等琐碎任务的编程。在 EJB 规范中，它们由 EJB 容器负责。EJB 包含接口和类。客户端通过 EJB 的本地接口和远程接口访问 EJB 方法。本地接口提供的方法可用于生成、删除和查找 EJB，远程接口则提供业务方法。部署时，容器从这些接口生成类，这些类使客户端可以访问、生成、删除、查找和调用 EJB 上的业务方法。EJB 类为业务方法、生成方法和查找方法提供实施，如果 Bean 管理自己的存储，还得提供生成生命周期方法的实施。

EJB 共有三种类型：EntityBean（实体 Bean）、Session Bean（会话 Bean）和 Message Driven Bean（消息驱动 Bean），下面分别说明。

（1）实体 Bean。实体 Bean 表示数据库中的数据及作用于数据的方法。在关系型数据库中，表中的每一行就是一个 Bean 的实例。实体 Bean 是具有持久性的事务处理型 EJB，只要数据存在于数据库中，实体 Bean 就存在。

用容器管理的持久性访问关系数据库的 EJB，不需要为数据库访问使用任何 JDBC API，因为容器可以负责完成这项任务。但是，如果使用 Bean 管理的持久性或想访问关系数据库以外的企业信息系统，就需要提供相应的程序代码才能完成。但是如果 EJB 使用 Bean 管理的持久性访问数据库，用户必须借助于 JDBC API 实施 Bean 生命周期方法，这样才能加载和保存数据，并保持运行和持久数据库存储之间的一致性。

（2）会话 Bean。会话 Bean 代表与客户间的短暂对话。在执行数据库读写时，会话 Bean 可以请求 JDBC 调用，也可以使用实体 Bean 执行调用，这时会话 Bean 是实体 Bean 的客户端。会话 Bean 的字段中包含对话的状态，如果服务器或客户端出现故障，会话 Bean 将消失。

会话 Bean 可以有状态，也可以无状态。有状态会话 Bean 包含客户端方的对话状态，对话状态是会话 Bean 实例的字段值加上可以从会话 Bean 字段阅读的所有对象。有状态的会话 Bean 不表示持久数据库中的数据，但能够以客户端的名义访问和更新数据。

无状态会话 Bean 没有客户端的任何状态信息。它们一般不提供保留任何状态的服务器行为。无状态会话 Bean 需要的系统资源较少。提供通用服务或表示共享数据视图的业务对象适合作为无状态的会话 Bean。

（3）消息驱动 Bean。EJB2.0 规范中的消息驱动 Bean 能处理从 JMS 消息队列接收到的异步消息。JMS 将消息路由到消息驱动 Bean，由消息驱动 Bean 从池中选择某个实例处理消息。

消息驱动 Bean 在 EJB 容器中管理。由于它们不是由用户的应用直接调用的，因此不能借助 EJB 本地接口从应用进行访问。但是，用户的应用可以将消息发送到 Bean 所监听的 JMS 队列中，以此来实例化消息驱动 Bean。

4．JMS

JMS 是支持 Java 程序之间信息交换的 J2EE 机制。这也是 Java 支持异步通信的方法——发送者和接收者无须相互了解，因此可以独立操作。JMS 支持两种消息传播模式：

点到点（point to point）。基于消息队列，消息产生者将消息发送到队列中。消息消费者可以将自身与队列连接，以倾听消息。当消息到达队列时，客户可以从队列中取走，并给出响应。消息只能发送到一个队列，只能由一个消费者使用。消费者可以过滤消息，以便获得希望获得的消息。

出版和订阅（publish/subscribe）。消息生产者将消息发送到一个话题（topic），注册到此话题的消费者都能接收到这些消息。这种情况下，许多消费者都能接收到同样的消息。

5．JNDI

由于 J2EE 应用的构件可以独立运行，而且是在不同的设备上运行，因此客户端和应用服务器层代码必须以某种方式查找和参考其他代码和资源。客户端和应用代码使用 JNDI（Java Naming and Directory Interface，Java 命名和目录接口）查找用户定义对象（如 EJB）和环境实体（Environment Entities）。在 JDBC2.0 中，数据源可以绑定到 JNDI 上，并允许应用程序访问。

6．事务处理

J2EE 事务处理模型可以在部署过程中定义组成一个事务处理的方法之间的联系，以便事务处理中的所有方法可以作为一个整体存在。如果用户希望完成这一任务，因为事务处理是一系列步骤，要么全部执行成功，要么全部回滚。例如，EJB 中可能有一系列方法，其作用是将资金从一个账户转到另一个账户，方法是借记第一个账户和贷记第二个账户。用户可能希望将全部操作作为一个整体，这样，如果借记之后，贷记之前出现故障，借记将滚回。

事务处理的属性在应用构件的集成过程中确定。它可以将各种方法组合成应用构件间的事务处理，即用户可以在 J2EE 应用中方便地重新分配应用构件的事务处理属性，无须修改代码和重新编译。J2EE 事务处理 API（JTA）和 Java 事务处理服务（JTS）形成 J2EE 中事务处理支持的基础，而且更适合 EJB 和 JDBC2.0。JTS 是低级事务处理管理 API，主要作用是将 Java 映射到对象管理组（OMG）的对象事务处理服务。JTA 是高级 API，包括两个部分：

事务处理接口。该接口允许事务处理定界，通过分布式构件由进行全局事务处理登记来完成工作。这种方法可以令多组操作组成一个事务处理。

XA 资源接口。基于能处理分布式事务处理的 X/Open/XA 接口，有时也称为两步提交事务处理，需要多种资源之间的协调，如数据库或序列。分布式事务处理由两步提交协议协调，可跨越用 XA 兼容的 JDBC 驱动程序访问的多个数据库，如针对 Oracle/XA 的 BEA WebLogicDriver 等。

EJB 规范定义了 Bean 管理的事务处理和 Container 管理的事务处理。当 EJB 用 Container 管理的事务处理部署时，应用服务器将自动协调事务处理。如果 EJB 由 Bean 管理事务处理部署，EJB 参数必须提供事务处理代码。

基于 JMS 或 JDBC API 的应用代码可以启动事务处理，或参与先前启动的事务处理。一个事务处理联系与执行应用的应用服务器线程相关，所有事务处理操作都在参与当前事务处理的线程上执行。

多数情况下，用户无须担心用 JTA 编写明确事务处理的问题，因为此项工作由 JDBC 完成，EJB API 由 Container 处理，并由应用部署说明符配置。这样，用户就可以将精力集中在事务处理设计而非实施上。

15.4 .NET

Microsoft.NET 战略基于一组开放的互联网协议，推出了一系列的产品、技术和服务，吹响了互联网技术变革的号角。关于.NET，微软公司 CEO 鲍尔默这样描述：“Microsoft.NET 代表了一个集合、一个环境、一个可以作为平台支持下一代 Internet 的可编程结构。”这句话简单扼要地概括了.NET 的外部特性。

15.4.1 .NET 平台

.NET 首先是一个环境。这是一个理想化的未来互联网环境，微软的构想是一个“不再关注单个网站、单个设备与 Internet 相连的网络环境，而是要让所有的计算机群、相关设备和服务商协同工作”的网络计算环境。简而言之，互联网提供的服务，要能够完成更高程度的自动化处理。未来的互联网，应该以一个整体服务的形式展现在最终用户面前，用户只需要知道自己想要什么，而不需要一步步地在网上搜索、操作来达到自己的目的。

而要搭建这样一种互联网环境，首先需要解决的问题是针对现有 Internet 的缺陷来设

计和创造一种新一代 Internet 结构。这种结构不是物理网络层次上的拓扑结构，而是面向软件和应用层次的一种有别于浏览器只能静态浏览的可编程 Internet 软件结构。因此.NET 把自己定位为可以作为平台支持下一代 Internet 的可编程结构。

.NET 的最终目的就是让用户在任何地方、任何时间，以及利用任何设备都能访问他们所需要的信息、文件和程序。而用户不需要知道这些东西存在于什么地方，甚至连如何获得等具体细节都不需要知道。他们只需发出请求，然后只负责接收即可，而所有后台的复杂性是完全屏蔽起来的。故而，对于企业的 IT 人员来说，工作量将大大减少，他们也不需要管理复杂的平台及了解各种分布应用之间的工作是如何协调的。

.NET 包括 4 个重要特点：一是软件变服务，二是基于 XML 的共同语言，三是融合多种设备和平台，四是新一代的人机界面。这 4 个特点基本上覆盖了.NET 的技术特征。

（1）软件变服务。鲍尔默在谈到软件服务时说道：“今天的软件产品仅仅是一张光盘，用户购买软件，亲自安装、管理和维护。但是软件服务是来自 Internet 的服务，它替用户 安装、更新和跟踪这些软件，并让它们和用户一同在不同的机器间漫游。它为用户存储自 己的信息和参考资料。这些就是软件和软件服务各自不同的风格。”这段话概括了软件变服务的核心。

伴随着 ASP（应用服务提供）产业的兴起，软件正逐渐从产品形式向服务形式转化，这是 IT 行业的整体趋势。在.NET 中，最终的软件应用是以 Web 服务的形式出现并在 Internet 发布的。Web 服务是一种包装后的、可以在 Web 上发布的构件，.NET 通过 WSDL 协议来描述和发布这种 Web 服务信息，通过 DISCO 协议来查找相关的服务，通过 SOAP 协议进行相关的简单对象的传递和调用。

如图 15-4 所示，.NET 平台中 Orchestration 可视化编程工具用于产生基于 XML 的 XLANG 代码，它和 BizTalk 服务器、.NET 框架，以及 VisualStudio.NET 都曾是 Windows DNA 2000 战略的重要部分。

微软的.NET 战略意味着：微软公司及在微软平台上的开发者将会制造服务，而不是制造软件。在未来几年之内，微软将陆续发布有关.NET 的平台和工具，在 Internet 上开发 Web 服务。那时，工作在.NET 上的用户、开发人员和 IT 工作人员都不再购买软件、安装软件和维护软件。取而代之的是，他们将定制服务，软件会自动安装，所有的维护和升级也会通过互联网进行。

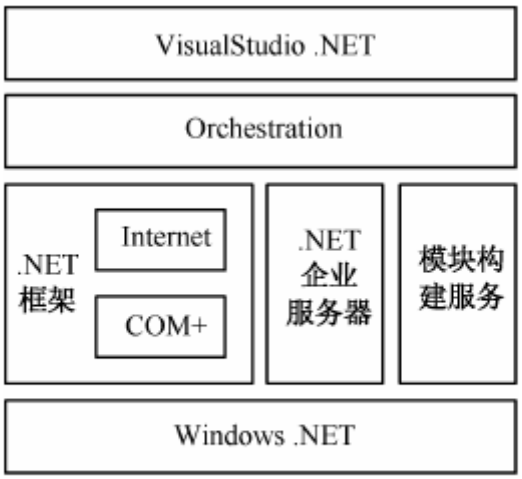


图 15-4　.NET 平台示意图

（2）基于 XML 的共同语言。XML 是从 SGML 语言演化而来的一种标记语言。作为元语言，它可以定义不同种类应用的数据交换语言。在.NET 架构中，XML 作为一种应用间无缝接合的手段，用于多种应用之间的数据采集与合并，用于不同应用之间的互操作和协同工作。具体而言，.NET 通过 XML 语言定义了简单对象访问协议（Simple Object Access Protocol，SOAP）、Web 服务描述语言（Web Services Description Language，WSDL）、Web 服务发现协议。SOAP 协议提供了在无中心分布环境中使用 XML 交换结构化有类型数据的简单轻量的机制。WSDL 协议定义了服务描述文档的结构，如类型、消息、端口类型、端口和服务本身。Web 服务发现协议定义了如何从资源或者资源集合中提取服务描述文档、相关服务发现算法等。

（3）融合多种设备和平台。随着 Internet 逐渐成为一个信息和数据的中心，各种设备和服务已经或正在接入和融入 Internet，并试图成为其中的一部分。.NET 谋求与各种 Internet 接入设备和平台的一体化，主要关注在无线设备和家庭网络设备及相关软件、平台方面。

（4）新一代的人机界面。新一代人机界面主要体现在“智能与互动”两个方面。.NET 包括通过自然语音、视觉、手写等多种模式的输入和表现方法；基于 XML 的可编辑复合信息架构——通用画布；个性化的信息代理服务；使机器能够更好地进行自动处理的智能标记等技术。

.NET 的平台及框架是基于微软软件工业基础的又一次升级和演化。然而，.NET 还是要尽力保证 Windows 系统及系列产品和.NET 能够融为一体，尽量在微软公司原有的软件资产基础上，使.NET 继续成为 Internet 的中心。

15.4.2 .NET 框架

Microsoft.NET 开发框架如图 15-5 所示。通用语言运行时及它所提供的一组基础类库是整个开发框架的基础；在开发技术方面，.NET 提供了全新的数据库访问技术 ADO.NET，以及网络应用开发技术 ASP.NET 和 Windows 编程技术 WinForms；在开发语言方面，.NET 提供了 VB、VC++、C#等多种语言支持；而 VisualStudio.NET 则是全面支持.NET 的开发工具。

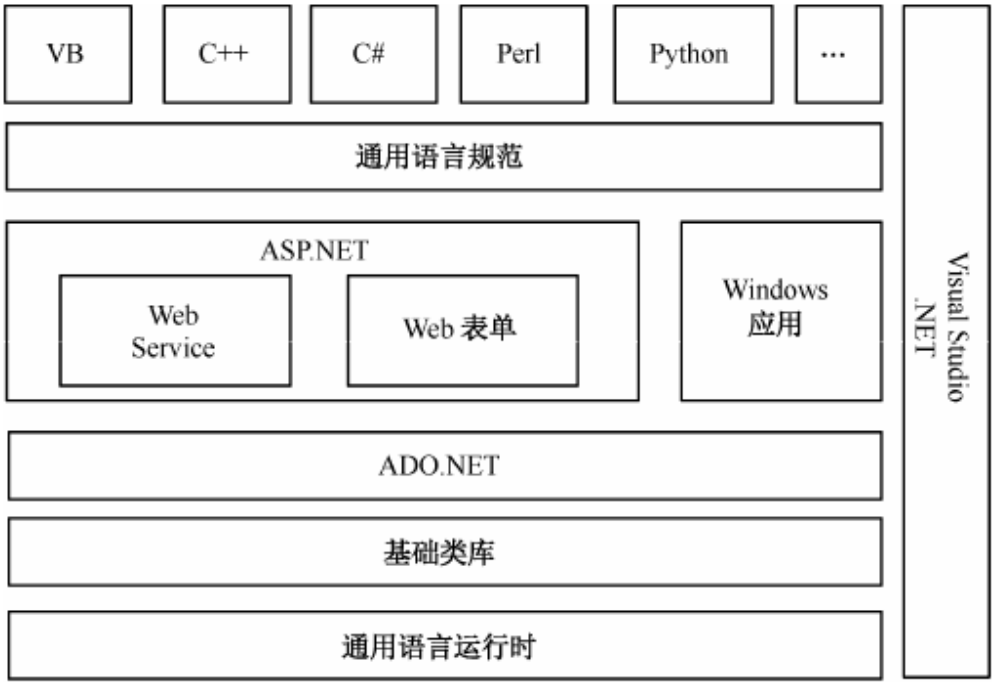


图 15-5 .NET 开发框架图示

1．通用语言运行时

Microsoft.NET 给开发人员带来了一种全新的开发框架，而通用语言运行时（Common Language Runtime，CLR）则处于这个框架的最底层，是这个框架的基础。读者也许对于所谓的 C 运行时、VB 运行时、Jave 虚拟机这些概念已经有所了解，而通用语言运行时则为多种语言提供了一种统一的运行环境。另外，它还提供了更多的功能和特性，例如，统一和简化的编程模型，用户不必迷惑于 Win32API 和 COM；避免了 DLL 的版本和更新问题，从而大大简化了应用程序的发布和升级；多种语言之间的交互，例如，甚至可以在 VB 中使用 C++编写的类；自动的内存和资源管理等。Microsoft.NET 正是基于通用语言运行时，实现了这些开发人员梦寐以求的功能。

基于通用语言运行时开发的代码称为受控代码，它的运行步骤大体如下：首先使用一种通用语言运行时支持的编程语言编写源代码，然后使用针对通用语言运行时的编译器生成独

立于机器的微软中间语言（Microsoft Intermediate Language，MIL），同时产生运行所需的元数据，在代码运行时再使用即时编译器（Just In Time Compiler，JITC）生成相应的机器代码来执行。

2．基础类库

当然对于开发者而言，他们除了关心通用语言运行时提供的那些新特性外，还关心它究竟给开发者提供了什么样的编程接口，这就是基础类库（Base Class Library）。这组基础类库包括了从输入输出到数据访问等各方面，提供了一个统一的面向对象的、层次化的、可扩展的编程接口。它使用一种点号分隔的方法，使得查找和使用类库非常容易。例如基础类库中的根，它的命名空间是 System，提供数据访问的类库的命名空间是 System.Data。在使用时，开发者只需在自己的应用中添加所需的基础类库的引用，然后就可以使用这个类库中的所有方法、属性等。跟传统的 Windows 编程相比，使用和扩展基础类库都非常容易，这使得开发者能够高效、快速地构建基于下一代互联网的网络应用。

3．ADO.NET

几乎所有的应用程序都需要访问从简单的文本文件到大型的关系型数据库等各种不同类型的数据。在 Microsoft.NET 中访问数据库的技术是 ADO.NET。ADO.NET 提供了一组用来连接到数据库、运行命令、返回记录集的类库，与从前的 ADO（ActiveX Data Object）相比，Connection 和 Command 对象很类似，而 ADO.NET 的革新主要体现在如下几个方面：

首先,ADO.NET 提供了对 XML 的强大支持，这也是 ADO.NET 的一个主要设计目标。在 ADO.NET 中通过 XMLReader，XMLWriter，XMLNavigator，XMLDocument 等可以方便地创建和使用 XML 数据，并且支持 W3C 的 XSLT、DTD、XDR 等标准。ADO.NET 对 XML 的支持也为 XML 成为 Microsoft.NET 中数据交换的统一格式提供了基础。

其次，ADO.NET 引入了 DataSet 的概念，这是一个驻于内存的数据缓冲区，它提供了数据的关系型视图。不管数据来源于一个关系型的数据库，还是来源于一个 XML 文档，人们都可以用一个统一的编程模型来创建和使用它。它替代了原有的 Recordset 的对象，提高了程序的交互性和可扩展性，尤其适合于分布式的应用场合。

另外，ADO.NET 中还引入了一些新的对象，例如 DataReader 可以用来高效率地读取数据，产生一个只读的记录集等。简而言之，ADO.NET 通过一系列新的对象和编程模型，并与 XML 紧密结合，使得在 Microsoft.NET 中的数据操作十分方便和高效。

4．ASP.NET

ASP.NET 是 Microsoft.NET 中的网络编程结构，它使得建造、运行和发布网络应用非常

方便和高效。可以从以下几个方面来了解 ASP.NET：

（1）Web 表单。ASP.NET WEB 表单的设计目的就是使开发者能够非常容易地创建 Web 表单，它把 VB 中的快速开发模型引入网络开发中，从而大大简化了网络应用的开发。具体来说：在 ASP.NET 中可以支持多种语言，不仅仅支持脚本语言，通用语言运行时支持的所有语言在 ASP.NET 中都可以使用；代码和内容分开，在现在的 ASP（Active Server Pages）开发中，内容和脚本交错，维护和升级很困难，将它们分开可以使得开发人员和设计人员能够更好地分工合作，提高开发效率；另外在 ASP.NET 中通过引入服务器端控件，将类似 VB 的快速开发应用到了网络开发中来，这样大大提高了构建网络表单效率，并且服务器端控件是可扩展的，开发者可以建造自己需要的服务器端控件。

（2）ASP.NET Web 服务。Web 服务是下一代可编程网络的核心，它实际上就是一个可命名的网络资源，可用来在 Internet 范围内方便表现和使用对象，就像使用今天的 COM 对象一样，不同的是使用和表现网络服务是通过 SOAP 甚至 HTTP 来实现的。在 ASP.NET 中，建造和使用网络服务都非常方便。

在 ASP.NET 中建造网络服务就是编写一个后缀为.ASMX 的文件，在这个文件中加入想要表现出来的方法即可，网络服务的建造者不需要了解 SOAP，XML 的细节，只需要把精力集中在自己的服务本身，这也为独立软件服务开发商提供了很好的机会；使用网络服务最简单的方式就是使用 HTTP 协议（HTTPGET 或 HTTPPOST），用户只需要直接访问网络服务（.ASMX 文件）的 URL 即可；当然用户还可以通过 SOAP 在自己的应用中更灵活地使用网络服务。

（3）ASP.NET 应用框架。ASP.NET 应用不再是解释脚本，而是编译运行，再加上灵活的缓冲技术，使其性能从根本上得到了提高；由于 ASP.NET 的应用框架基于通用语言运行时，故而发布一个网络应用，仅仅是一个复制文件的过程，即使是构件的发布也是如此，更新和删除网络应用，可以直接替换/删除文件；开发者可以将应用的配置信息存放到 XML 格式的文件中，管理员和开发者对应用程序的管理可以分开进行；提供了更多样的认证和安全管理方式；在可靠性等多方面都有很大提高。

5．WinForms

传统的基于 Windows 的应用（WinForms），它仍然是 Microsoft.NET 战略中不可或缺的一部分。在 Microsoft.NET 中开发传统的基于 Windows 的应用程序时，除了可以利用现有的技术例如 ActiveX 控件及丰富的 Windows 接口外，还可以基于通用语言运行时开发，可以使用 ADO.NET、网络服务等，这样也可以实现诸如避免 DLL 地狱、多语言支持等.NET 的

新特性。

6．开发语言

从上面的介绍中，已经知道 Microsoft.NET 开发框架支持多种语言，在目前的测试版中已经支持 VB，C++，C#和 Jscript 四种语言及它们之间的深层次交互。而且微软支持第三方生产针对 Microsoft.NET 的编译器和开发工具，这也就是说几乎所有市场上的编程语言都有可能应用于 Microsoft.NET 开发框架。这样开发者可以任意选择自己喜爱的语言，这种开放和交互的特性正是开发者所热爱的。

需要特别指出的是，微软在 Microsoft.NET 中推出了全新的 C#语言，这种全新的面向对象的语言使得开发者可以快速地构建从底层系统级到高层商业构件的不同应用。C#在保证了强大的功能和灵活性的同时，给 C 和 C++带来了类似于 VB 的快速开发，并且它还针对.NET 做了特别设计，例如，C#允许 XML 数据直接映射为它的数据类型等，这些特性结合起来使得 C#成为优秀的下一代网络编程语言。

与此同时，Microsoft.NET 对原有的 VB 和 C++也做了很大的改进，使得它们更加适应 Microsoft.NET 开发框架的需求。例如在 VisualBasic.NET 中增加了继承等面向对象的特性，结构化的出错处理等；可管理的 C++扩展，大大提高了利用 C++来开发 Microsoft.NET 应用的效率等。

Visual Studio.NET 和.NET 开发框架紧密结合，是构建互联网应用的优秀工具。VisualStudio.NET 通过提供一个统一的集成开发环境及工具，大大提高了开发者的效率；集成了多种语言支持；简化了服务器端的开发；提供了高效地创建和使用网络服务的方法等。.NET 框架的一个主要目的是使 COM 开发变得更加容易。COM 开发过程中最难的是 COM 基本结构的处理。因此，为了简化 COM 开发，.NET 框架实际上已自动处理了所有在开发人员看来是与 COM 紧密相关的任务，包括引用计算、接口描述及注册。必须认识到，这并不意味着.NET 框架构件不是 COM 构件。事实上，使用 VisualStudio 6.0 的 COM 开发人员可以调用.NET 框架构件，并且在他们看来，后者更像是拥有 iUnknown 数据的 COM 构件。相反，使用 VisualStudio.NET 的.NET 框架开发人员则将 COM 构件视作.NET 框架构件。

为了避免引起误解，这里需对这种关系加以特别说明：COM 开发人员必须手动去做大多数.NET 框架开发人员可以在运行时自动执行的事情。例如，必须手写 COM 构件的安全性模块，且无法自动管理模块占用的内存，而在安装 COM 构件时，注册条目必须放进 Windows 注册表中。对.NET 框架而言，运行时实现了这些功能的自动化。例如，构件本身

是自我描述型的，因而无须注册到 Windows 注册表中便能安装。

当把 COM 与 Microsoft 事务服务器（MTS）和分布式 COM（DCOM）结合在一起时，就变成了 COM+。COM+提供了一组面向中间层的服务。特别是 COM+提供了进程管理功能和数据库与对象连接池处理功能。在将来的版本中，它还将提供一种称为分区的功能——专门为应用程序服务提供商设计的更强大的进程隔离功能。COM+服务主要面向中间层应用程序开发，并主要为大型分布式应用程序提供可靠性和可扩展性。这些服务是对.NET 框架所提供服务的补充；通过.NET 框架类，可以直接访问这些服务。

7．其他特征

.NET 框架有几个要素值得一提。首先是它的安全系统和配置系统。这两个系统协同工作，有力地遏止了运行不安全代码的可能性，并大幅度减少了“DLL 地狱”对应用程序进行配置时所面临的挑战。

安全系统是一个高度细化、基于事实的系统，它赋予开发人员和管理员多种代码处理权限（而不仅仅是“on”或“off”）。将来，还会根据代码本身的核心要素来决定如何实施上述权限。

例如，当.NET 框架应用程序被下载到某一系统中时，它会申请一组权限（诸如对临时目录的写入权限）。运行时将收集有关应用程序的事实信息（诸如：它是从何处下载的、是否用了有效签名、甚至它访问系统的准确程度），并按管理策略决定是否允许应用程序运行。运行时甚至还可告之应用程序它无法授权申请的所有权限，并允许应用程序自行决定是否继续运行。

有这种安全系统做保障，许多应用程序配置问题便会迎刃而解。开发人员和管理员（最终是用户）所面临的最大挑战之一是版本的管理问题。如果在新装了某个应用程序之后，一切都陷于瘫痪状态，而在这之前系统一直运行得非常良好，那么最大的可能就是新安装的应用程序重写了一些共享库，并极有可能修正了现有应用程序正使用的程序错误。这种情况出现的频率很高，以致人们将它称为“DLL 地狱”。

.NET 框架拥有的几项高级功能可以彻底消除“DLL 地狱”现象。首先，它有一个非常强大的内部命名系统，能够有效地防止两个库因互相重名而被错当为对方的情况发生。除此之外，它还提供一项被称作“并行”配置的新功能。如果前例中新安装的应用程序确实重写了共享库，现有应用程序可对该库进行修复。等现有应用程序再次启动时，它会检查所有的共享文件。如果发现文件被更改，同时这些更改又是不兼容的，则它可以请求运行时提取一个它可以使用的版本。得益于强大的安全系统，运行时可以安全地执行该操作， 这样应用程序就完成了本身的修复工作。

总之，Microsoft.NET 开发框架在通用语言运行时的基础上，给开发者提供了完善的 基础类库、下一代的数据库访问技术 ADO.NET、网络开发技术 ASP.NET，开发者可以使 用多种语言及 VisualStudio.NET 来快速构建下一代的网络应用。随着相关的互联网标准及 技术的普及，可以预言将会有越来越多的开发者采用这种开发结构，并开发出丰富多样的下一代互联网应用。

15.5 企业应用集成

许多企业的信息系统在最初设计时没有考虑多个系统“协同工作”的需要。这主要是由于企业信息化建设者对信息系统由不熟悉到熟悉，从了解信息化的好处，到真正体会到好处需要一个长期的过程，这就客观上造成企业信息化建设缺乏一个整体规划，实际需要的时候才会想到。因而，企业的信息化往往是从单项业务系统开始的，不同系统的开发方式及对于开发规范的遵从程度都有所不同，这使得系统间存在很强的孤立性，再加上对企业外部的信息未予以足够的重视，致使各部门开发出的信息系统最终成为一个个信息孤岛，一个系统很难与其他系统交换信息。同时，大多数企业都有过去遗留下来的异构的系统、应用、商务流程及数据源构成的应用环境。应用环境的通信状况是混乱的，只有很少的接口文档，并且维护代价也非常昂贵。

据有关数据统计，一家典型的大型企业平均拥有 49 个应用系统，33%的 IT 预算是花在传统的集成上，而且普遍是通过“点对点”连接，如图 15-6 所示，众多的信息孤岛联系起来，以便让不同的系统之间交换信息。

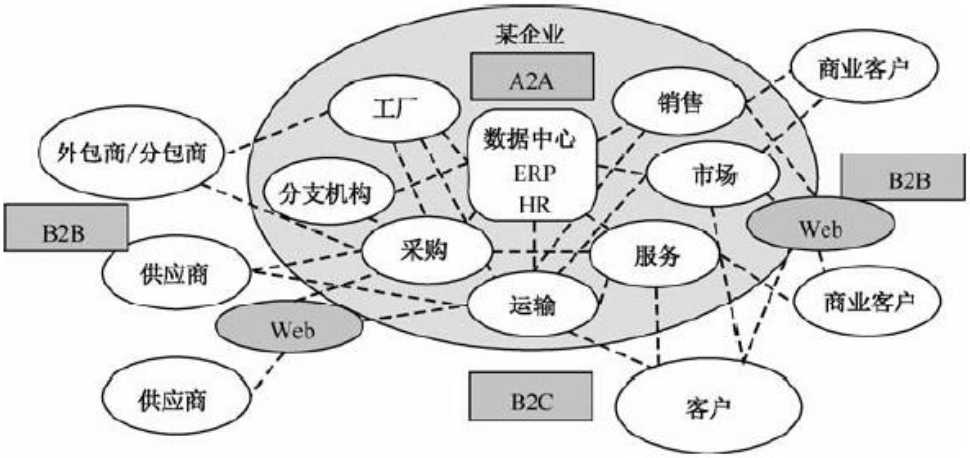


图 15-6　某企业应用系统状况图示

EAI（Enterprise ApplicationIntegration，企业应用集成），可以在一定程度上帮助人们解

决这一问题。EAI 是指通过将业务流程，应用软件、硬件和各种标准联合起来，对企业中完成不同业务功能的应用系统进行无缝集成，使它们像一个整体一样进行业务处理和信息共享，从而提高企业效率，为客户提供灵活的业务服务。

EAI 使人们可以从更高层次来看待企业内的信息资源，使新的信息和应用可以通过可插拔的方式和原有的资源在一个全新的信息集成共享平台上协同工作，共同发挥“1+1>2” 的集成效应。企业在借助 EAI 系统整合企业内部已有的各种信息系统的同时，也加速了数据的即时共享和提高了企业的信息反应能力。特别是，目前处在电子商务时代的企业不仅仅需要在企业内部系统之间进行集成，同时也需要对供应链中的不同环节进行集成。而 EAI 不仅是连接企业内应用的高效手段，它也是在企业之间建立信息沟通共享的一种科学而有效的方式，从而有效地降低供应链网络的整体拥有成本。

EAI 可以通过中间件技术来连接企业级各种应用，使异构应用系统之间能够相互“交流”与“协作”，如图 15-7 所示。通过 15.1 节的介绍，读者对中间件技术的特点有了一定的了解，这里就不再赘述。正因为这些特点，中间件技术能给企业带来的好处也就显而易见了。

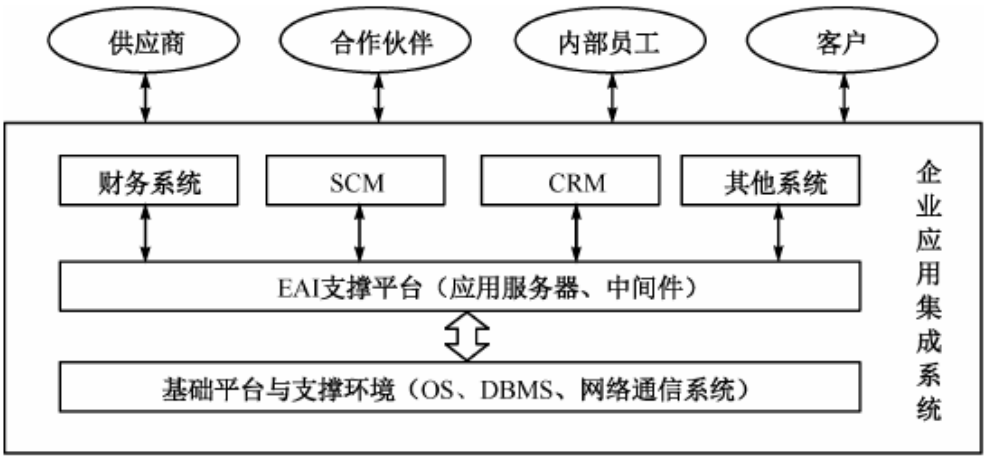


图 15-7　EAI 示意图

首先，中间件产品对各种硬件平台、操作系统、网络数据库产品及客户端实现了兼容和开放。

其次，中间件保持了平台的透明性，使开发者不必考虑操作系统的问题。

第三，中间件实现了对交易的一致性和完整性的保护，提高了系统的可靠性。

第四，中间件产品可以缩短开发周期 50%～75%，从而大大地降低了开发成本，提高了工作效率。

EAI 包括的内容很复杂，涉及结构、硬件、软件及流程等企业系统的各个层面，根据 EAI

集成的深度来划分可以分为应用集成、业务过程集成、数据集成。

1．应用集成

应用层次的集成主要为两个以上的应用中的数据和函数提供接近实时的集成。在网络环境中的跨平台应用程序之间的应用到应用（Application to Application，A2A）的集成。它涵盖了普通的代码（COBOL，C++，Java）撰写、应用程序接口、远端过程调用、分布式中间件如 TP 监控、分布式对象、CORBA、RMI、面向消息的中间件及 Web 服务等各种技术。应用层次的集成一般来说是通过处理多个应用系统之间的消息交换，实现系统间的集成。各个应用能够处于同步模式，即基于客户（请求程序）和服务器（响应程序）之间的请求响应交互机制。应用系统能够自己处理消息的转换，并且它将影响被集成系统的数据转换和有效性。但是，这需要对系统进行修改以建立发送和接收消息的接口。

2．业务过程集成

业务过程集成需要处理企业范围内的业务过程和把企业存在的应用系统整合到这些业务过程中。它是一个完全的企业应用集成实现策略，因为它使企业内的一个个分离系统变成了一个支持业务过程的连续系统，从而满足企业的整个业务过程需求。

当对业务过程进行集成的时候，企业必须在各种业务系统中定义、授权和管理各种业务信息的交换，以便改进操作、减少成本、提高响应速度。业务过程集成包括业务管理、进程模拟，以及综合任务、流程、组织和进出信息的工作流，还包括业务处理中每一步都需要的工具。业务过程集成至少包括以下两种形式的流程。

（1）交互式流程。交互式流程包含了跨两个系统之间的事务处理。这种流程是完整的且不间断的，它不包含任何需要人为参与的工作和间断的流程。由于交互式流程通常是在两个系统之间流转的，它不需要特别复杂的 EAI 处理。

（2）多步流程。作为业务流程的一部分，许多单独的事务处理根据事先定义的顺序在两个或者多个系统之间流转，这就涉及工作流和业务流程重组。多步流程有一系列的步骤并同多个系统相关，能在一定的时间内完成。这种流程可以是一对多、多对一或者多对多的关系。

3．数据集成

要完成应用集成和业务过程集成，必须首先解决数据和数据库的集成问题。为了处理多个数据库之间的数据移动，很多企业把数据级 EAI 作为他们实施 EAI 的切入点。当应用系统必须分享信息时，这种集成可以支持不同数据库之间的数据交换。目前有很多支持数据级 EAI 的工具，这使得数据级 EAI 实现起来相对容易，甚至不用修改应用系统的源程序。通

行的做法就是将历史数据批量导入新系统中和现行系统中的批量、实时数据处理，也称数据同步。

随着数据仓库的建立，越来越多的数据同步工作能够采用批量的方式来处理。这样可以掌握更多的信息，例如客户类型、客户交易历史和客户习惯的购买、交货方式都能够每日或者每周更新一次。关键数据、新客户的数据和可用库存增加的需求都能够进行批量实时的更新。很多企业也在寻找方法来进行批量数据的集成，缓解日益增长的数据给数据同步带来的压力。

当然，更深层次的数据集成，需要首先对数据进行标识并编成目录，另外还要确定元数据模型。这三步完成后，数据才能在数据库系统中分布和共享。但是，目前数据集成解决方案中最普遍的方法发生在企业内的数据库和数据源级别，即通过从一个数据源将数据移植到另外一个数据源来完成数据集成。下面举出数据集成的一些例子：

将订单从 ERP 系统更新到 CRM（Customer Relationship Management，客户关系管理）系统中，以便销售人员能够实时了解订单的情况。

从多个系统中同步和规范客户信息，使企业能够 360°全面审视客户。

将运作数据实时地保存在系统中，客户和分销商能通过商业智能网络访问企业的库存和订单信息。

每天一次或者多次地将 ERP 中的数据导入 SCM（Supply Chain Management，供应链管理）系统中，将有助于企业制订物料需求计划。

每天多次将运输的价格信息传输给各个下游分销商。

数据集成的一个最大的问题是商业逻辑常常只存在于主系统中，无法在数据库层次去响应商业流程的处理，因此限制了实时处理的能力。

在企业内部，EAI 通过建立底层结构来联系横贯整个企业的异构系统、应用、数据源等，完成在企业内部的 ERP、CRM、SCM、数据库、数据仓库，以及其他重要的内部系统之间无缝地共享和交换数据的需要。而在电子商务时代，企业不仅需要在内部的应用系统之间进行集成，还需要对供应链中的不同企业系统进行集成，以帮助企业创建一条畅通于企业的各个部门以及它的供应商、承运商、分销商、零售商和顾客之间的信息流，从而进行有效的数据和业务集成。

特别是随着信息技的普及和企业各种应用的迅速增加，越来越多的企业开始采用 EAI 解决方案将企业内部的应用软件与外部客户和供应商的应用软件进行链接，实现数据流和业务运作的自动化，从而达到业务的实时与快速。好的企业应用集成解决方案可以实现对于未

来业务的集成，维护和修改实现时间和成本的节约，从而提升企业的核心竞争力。

15.6 轻量级架构和重量级架构

MVC 模式是一种目前广泛流行的软件设计模式，随着 J2EE 的成熟，它正成为 J2EE 平台上推荐的一种设计模型，将业务处理与显示分离，将应用分为模型、视图及控制层，增加了应用的可扩展性。MVC 模式为搭建具有可伸缩性、灵活性、易维护性的 Web 系统提供了良好的机制。

J2EE 多层结构的出现促进了软件业的巨大改变，但是，J2EE 只是提出了一般意义上的框架设计，且其庞大的体系显得有些臃肿。轻量 级 Web 架构不仅保持了 J2EE 的优势，还简化了 Web 的开发。目前主流的轻量级架构是把 Struts、Spring 和 Hibernate 这三种在业内比较推崇的开源技术基于 MVC 模式相结合，这样在项目开发中不管是从效率上，费用上，还是易维护上都能达到很好的效果。下面将分别介绍这三种框架。

15.6.1 Struts 框架

Struts 是一个基于 SUN J2EE 平台的 MVC 框架，主要是采用 Servlet 和 JSP 技术来实现的。在 Struts 框架中，模型由实现业务逻辑的 JavaBean 或 EJB 构件构成，控制器由 ActionServlet 和 Action 来实现，视图由一组 JSP 文件构成，如图 15-8 所示。

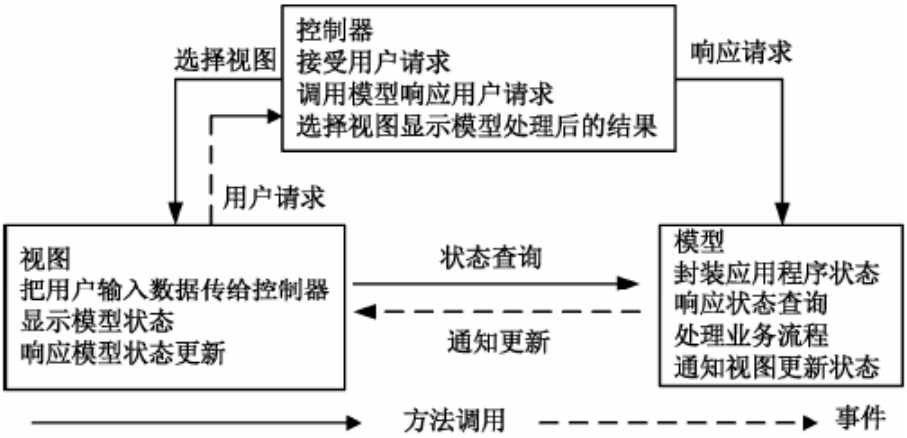


图 15-8 模型、视图和控制器的关系

Struts 把 Servlet、JSP、自定义标签和信息资源整合到一个统一的框架中，开发人员利用其进行开发时不用自己再编码实现全套 MVC 模式，极大地节省了时间。

Struts 将业务数据、页面显示、动作处理进行分离，这有利于对各部分的维护。Struts 采

用 Front Controller 模式来实现动作处理，使所有的动作请求都经过一个统一入口，然后进行分发。这方便了人们在入口中加入一些全局控制代码，如安全控制、日志管理、国际化编码等。通常情况下借助 Struts Validator 框架帮助完成 Web 层的验证工作，不用再去为每个 Web 页面写验证代码，只需通过配置即可实现。这也减少了开发量，由于验证代码的集中管理，也为维护带来便利。

Struts 的工作流程为：首先，JSPview 发起一个以.do 表示的请求；ActionForm 封装用户请求数据，同时提供验证数据的功能；ActionServlet 根据 struts-config.xml 文件来得到处理这个请求的 Action 对象，并将请求发送给这个 Action 对象；Action 对象调用 model 去处理这个请求，将结果返回给 ActionServlet；ActionServlet 决定将结果返回给对应的 view；view 得到结果，并将它显示给用户。这里需要提到的是，可以通过 Struts 提供的 ActionForm 封装 web form 中的元素，使重用 web 表单成为可能。

15.6.2 Spring 框架

Spring Framework 是由 Rod Johnson 创立的一个开放源码的应用框架。它是轻量级的 J2EE 应用程序框架，旨在简化 J2EE 的开发，降低 J2EE 项目实施的难度。这个框架包括声明性事务管理，通过 RMI 或 web services 远程访问业务逻辑，mail 支持工具，以及对于数据和数据库之间持久层的各种配置的支持。Spring 允许自由选择和组装各部分功能，还提供和其他软件集成的接口，如与 Hibernate、Struts 的集成。

Spring 核心本身是个容器,管理物件的生命周期、物件的组态、相依注入等，并可以控制物件在创建时是以原型（Pro-totype）或单例子（Singleton）的方式来创立。

Spring 的核心概念是控制反转（Inversionof Control，IoC），更具体而易懂的名词是依赖注入（Depen-dency Injection），使用 Spring，不必自己在程序码中维护物件的依赖关系，只需在构件中加以设定，Spring 核心容器会自动根据构件将依赖注入指定的物件。Spring 的目标是实现一个全方位的整合框架，在 Spring 框架下实现多个子框架的组合，这些子框架之间可以彼此独立，也可以使用其他的框架方案加以替代，Spring 成为企业级应用程序一站式的解决方案。其架构如图 15-9 所示。

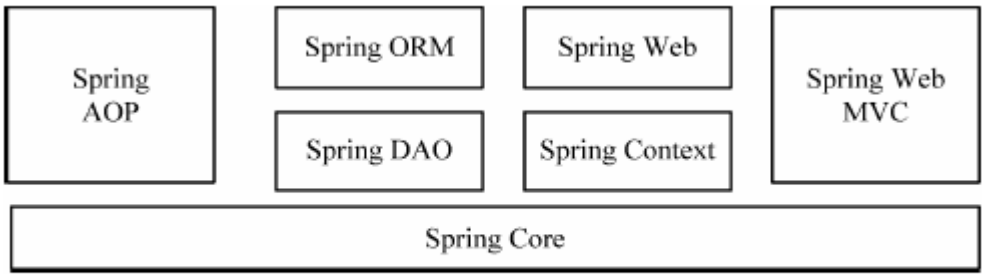


图 15-9　Spring 框架的 7 个模块

Spring 模块构建在核心容器之上。核心容器定义了创建、配置和管理 bean 的方式。Core 封装包的主要构件是 BeanFactory，它提供对 Factory 模式的经典实现来消除对程序性单例模式的需要，并真正地允许从程序逻辑中分离出依赖关系和配置。

DAO 提供了 JDBC 的抽象层，它可消除冗长的 JDBC 编码和解析数据库厂商特有的错误代码。并且，JDBC 封装包还提供了一种比编程性更好的声明性事务管理方法，不仅仅实现了特定接口，而且对所有的 POJOs（Plain Old Java Objects，普通的 Java 对象）都适用。

ORM 框架提供了对象关系映射工具，其中包括 JDO（Java Data Object，Java 持久对象）、Hibernate、Ibstis、JPA（Java Persistence API，Java 持久 API）。所有这些都遵从 Spring 的通用事务和 DAO 异常层次结构。

Context（上下文）是一个配置文件。Spring 的上下文包括企业服务，例如，EJB、JNDI、Remodeling Mail、Validation。

Web 包提供了基础的针对 Web 开发的集成特性，例如多方文件上传，利用 Servlet listeners 进行 IoC 容器初始化和应用上下文。当与 Web 或 Struts 一起使用 Spring 时，这个包使 Spring 可与其他框架结合。

通过策略接口，MVC 框架变为高度可配置的，容纳了大量视图技术。

Spring 的核心要点是支持不绑定到特定 J2EE 服务的可重用业务和数据访问对象。Spring 的 IoC 控件主要服务于利用类、对象和服务去组成一个企业级应用，通过规范的方式，将各种不同的控件整合成一个完整的应用。

15.6.3 Hibernate 框架

Hibernate 是一种对象和关系之间映射的框架，是 Java 应用和关系数据库之间的桥梁。它可以将数据库资源映射为一个或者多个 POJO。将面向数据库资源的各种业务操作以 POLO 的属性和方法的形式实现，使人们摆脱烦琐的 JDBC 代码，将精力更多地集中在编写

数据表示和业务逻辑上。Hibernate 的基本实现框架如图 15-10 所示。

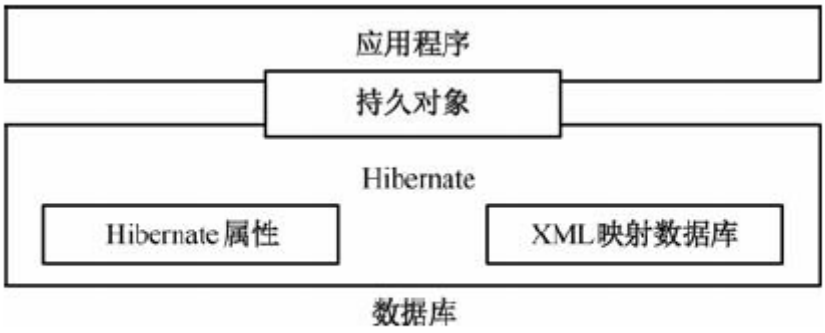


图 15-10 Hibernate 的基本实现框架

Hibernate 是一个工具，而不是一个 J2EE 的服务器。可以在各种流行的服务器中使用 Hibernate，利用 Hibernate 来作为持久化的处理技术，或者在桌面程序中直接利用 Hibernate 来完成数据库操作，还可以基于 Hibernate 成熟的持久化技术框架来扩展平台软件的功能，例如基于 Hibernate 来完成对 EJB3.0 标准的实现。Hibernate 在支持集成方面提供了对 JMX 标准的支持，实现了封装 Hibernate 全部功能的 MBean 接口。Hibernate 的作用如图 15-11 所示。

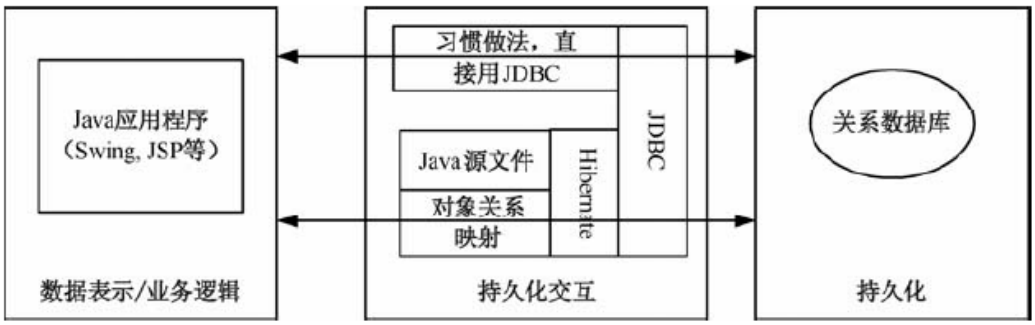


图 15-11 Hibernate 的作用

在 Hibernate 中对象/关系映射机制的核心是一个 XML 文件，通常命名为*.hbm.xml。这个映射文件描述了数据库模式是怎么与一组 Java 类绑定在一起的。Hibernate 提供工具从已有的数据库模式和 Java 代码生成*.hbm.xml 文件。一旦有了*.hbm.xml 文件，就可以生成 Java 代码，或数据库模式，或者两者兼得。

Hibernate 只是一个将持久化类与数据库表映射的工具，Hibernate 只需要将每个持久化实例对应于数据库表中的一个数据行即可。

15.6.4 基于 Struts、Spring 和 Hibernate 的轻量级架构

基于 Struts、Spring 和 Hibernate 框架，可以构造出 Web 轻量级架构。如图 15-12 所示，该系统逻辑上分成三层。

（1）表示层。由 Struts 实现，主要完成如下任务：管理用户请求和响应；提供一个控制器代理以调用业务逻辑和各层的处理；处理从其他层抛给 StrutsAction 的异常；为显示提供数据模型；借助 Struts Validator 框架帮助完成 Web 层的验证工作，通常情况下，不用再去为每个 Web 页面写验证代码，只需通过配置即可实现。Struts 里的控制器也就是 ActionServlet 会解析核心配置文件 struts- config.xml。其中用户登录验证的配置文件如下：

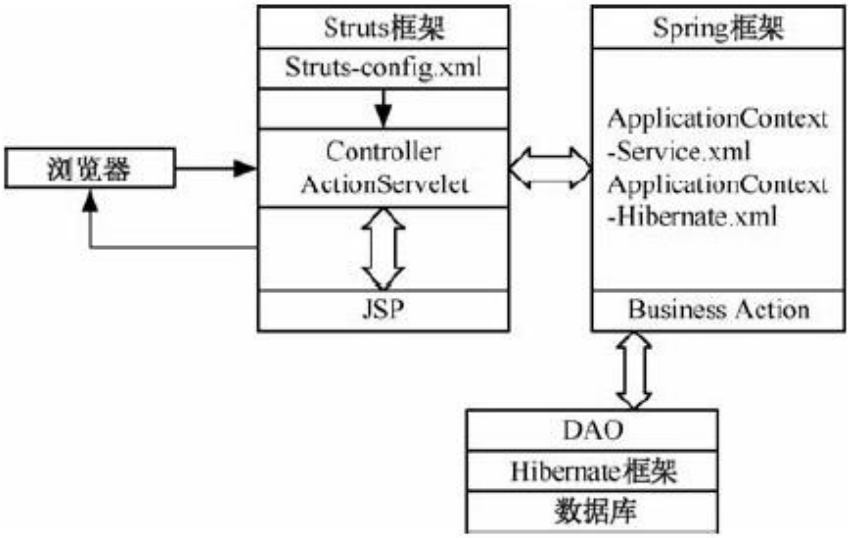


图 15-12　基于 Struts、Spring 和 Hibernate 的轻量级架构

```
<s truts -config>
<form-beans>
<form-bean name="loginForm"type="com.authorise.LoginForm">
<form-property name="name"type="java.lang.String" />
</form-bean> </form-beans> <action-mappings>
<actionpath="/login"name="loginForm"
type="com.authorise.LoginAction"scope="reques t" input="/Login.jsp"validate="false"/>
</action-mappings>
</s truts -config>
```

（2）持久层。由 Hibernate 实现。它通过一个面向对象的查询语言（Hibernate Query

Language，HQL）或正则表达式的 API 来检索对象的相关信息。HQL 类似于 SQL，只是把 SQL 里的表和列用对象和它的字段代替。Hibernate 还负责存储、更新、删除数据库记录。同时 Hibernate 支持大部分主流数据库，且支持父表/子表关系、事务处理、继承和多态。

（3）业务层。由 Spring 来实现。使用 Spring 的优点是：利用延时注入思想组装代码，提高了系统扩展性和灵活性，实现插件式编程。利用 AOP 思想，集中处理业务逻辑，减少重复代码，构建了较理想的解决方案。利用其对 Hibernate 的会话工厂（Session Factory）、事务管理的封装，可以更简洁地应用 Hibernate。

15.6.5 轻量级架构和重量级架构的探讨

重量级的开发倒并不是指 EJB 或者是 JNDI，很大意义上，重量级的开发都是需要依赖一个非常庞大的容器系统进行开发，在 EJB 的开发中，所有开发的内容基本都需要放置在一个容器系统中进行运行这些容器因为基本针对大型企业应用，所以体积庞大，占用资源过多，在开发的过程中效率很低。因为使用大型容器作为开发环境的话，很大一部分时间都用在了配置、运行这样的过程上，有时候改动一个小小的部分，需要等很长的时间才能看到结果。如果做单元测试也比较麻烦，虽然现在有很多针对容器的单元测试框架，但是还是没有很好地解决配置的等待问题，所以在开发者这里，EJB 逐渐失去了吸引力，因为感觉实在是太笨重了。

轻量级框架的优势很大程度上是因为加速了开发的速度，不用部署一个很庞大的容器系统就可以实现以前需要容器才能实现的功能，可以使用 Spring 代替 EJB 中的无状态的会话 Bean，可以使用 Hibernate 代替 EJB 中的实体 Bean，而且可以直接写一个应用程序运行已经完成的系统，马上可以看到结果，做单元测试非常简单，不需要做太多的工作就可以构建系统，这些特性对于开发人员来说非常有吸引力。

关于轻量级和重量级之间的论战已经由来已久了，最终也没有出现一个很好的结果，重量级框架在大规模运行的时候会表现出非常优异的性能，劣势主要是开发效率较低，轻量级框架正好相反，开发的时候非常迅速，但是在大规模运行的时候，性能与重量级框架相比还是有差异的。

但是随着最近一些框架标准的成熟，可以有新的选择，因为不管是轻量级还是重量级框架，基本解决的是两个问题，一个是事务控制，另一个是持久化控制，在 JPA 标准发布以后，看到一个很好的解决方式，持久化的开发可以和任何框架没有关系，直接使用 JPA 的

标准注解即可，所以开发持久化部分的时候可以使用 JPA 进行注解，开发时期用 Hibernate 作为 JPA 的实现进行开发测试，需要上线运行的时候就可以直接部署到 EJB 的实体 Bean 上，在 EJB 3.0 之后，已经很好进行移植部署了。关于事务控制，现在所有的实现方式都比较简单，针对方法进行注解事务类型即可，开发的时候可以用一个转换器将这些注解转化为 Spring 的映射，快速地进行开发，在上线运行的时候，直接使用 EJB 的会话 Bean 进行部署就可以解决，这些方式实现起来并不困难，可以很好地解决“重量级”和“轻量级”之间的矛盾。

第 16 章：安全性和保密性设计

随着科技进步、社会发展，尤其是以计算机为代表的信息技术飞速发展，各种信息呈爆炸式发展。计算机及信息技术的应用领域在不断扩展，计算机在政府、企业、民生等各个领域中都得到越来越广泛的应用。与此同时，网络攻击和入侵事件与日俱增，重要机构的信息系统遭黑客袭击的事件时有发生。攻击者可以从容地对那些缺乏足够安全保护的信息系统进行攻击和入侵，如进行拒绝服务攻击、从事非授权的访问、肆意窃取和篡改重要的数据信息、安装后门监听程序以便随时获得内部信息、传播计算机病毒、摧毁主机等。攻击和入侵事件给这些机构和组织带来了巨大的经济损失和形象的损害，甚至威胁到国家的安全。

信息化时代，人们对信息系统的安全需求越来越迫切。

信息安全，具体地说就是保证信息的保密性、完整性、真实性、占有性。

保密性是指系统中的信息必须按照该信息拥有者的要求保证一定的秘密性，不会被未经许可的第三方非法获取。系统必须阻止一切对秘密信息的非授权访问或泄露。

完整性是指系统中的信息应当安全、准确、有效，要求数据不能被非法改动或删除。完整性是信息安全的最基本要求。为了实现完整性，可以借助本章讲述的数字签名、加密等措施，从而有力地保护数据的完整。

真实性是指对信息的发送者身份的确认或系统中有关主体的身份确认，这样可以保证信息的可信度。信息的真实性可以通过数字签名、公钥加密等方式来实现。

占有性是指要保护信息赖以存储的节点、介质、载体等不被盗用或窃取。保护信息占有性的方法有使用版权、专利、商业秘密性，提供物理的或逻辑的存取限制方法，维护和检查有关窃取文件的记录等。

16.1 加密和解密

加密技术源远流长，自从古代有了信息的传递和存储，就有了加密技术的运用。此后，很长一段时间里，加密及解密技术在军事、政治、外交、金融等特殊领域里被普遍采用，并经过长时间的研究和发展，形成了比较完备的一门学科——密码学。

密码学是研究加密方法、秘密通信的原理，以及解密方法、破译密码的方法的一门科学。

加密和解密的过程大致如下：首先，信息的发送方准备好要发送信息的原始形式，叫作明文。然后对明文经过一系列变换后形成信息的另一种不能直接体现明文含义的形式，叫作密文。由明文转换为密文的过程叫作加密。在加密时所采用的一组规则或方法称为加密算法。接收者在收到密文后，再把密文还原成明文，以获得信息的具体内容，这个过程叫作解密。解密时也要运用一系列与加密算法相对应的方法或规则，这种方法或规则叫作解密算法。在加密、解密过程中，由通信双方掌握的参数信息控制具体的加密和解密过程，这个参数叫作密钥。密钥分为加密密钥和解密密钥，分别用于加密过程和解密过程。

在加密和解密的过程中，如果采用的加密密钥与解密密钥相同，或者从一个很容易计算出另一个，则这种方法叫作对称密钥密码体制，也叫作单钥密码体制。反之，如果加密和解密的密钥并不相同，或者从一个很难计算出另外一个，就叫作不对称密钥密码系统或者公开密钥密码体制，也叫作双钥密码体制。

16.1.1 对称密钥加密算法

在过去很长一段时间里，人们一直都采用对称密钥密码体制来对信息进行加密和解密，直到现在，对称密钥密码体制也仍然是一种非常重要的常用加密方法。

对称密钥密码体制中，加密和解密过程中所使用的是同一个密钥，或者即使加密密钥和解密密钥不同，但是很容易地由一个计算出另外一个。显然，在这种密码体制中，密钥成为整个秘密通信的核心，整个加密系统的安全性完全以密钥的保密为基础。如果密钥暴露，则整个密码体制就完全失去了保密的效果。所以说，密钥的保密是对称密钥加密体制安全保密的关键，必须妥善保存并经由可靠的渠道传递。

对称密钥加密算法有多种，例如，DES（Data Encryption Standard，数据加密标准）、 IDEA（International Data Encryption Algorithm，国际数据加密算法）、Skipjack、3DES、GDES、 New DES、Lucifer、FEAL N、LOKI 91、RC4、RC5 等。

1．DES 算法

DES 算法是 1977 年美国政府公布的一种加密算法，由于算法实现简单，加密效果好，在很长时间里在全世界范围都被广泛运用。它通过对数据进行非常复杂的迭代和置换进行加密，使得企图破译者从加密后的密文中无法获得任何有效信息。对这种加密方法，如果用穷举的方法进行攻击的话，由一台一秒钟能够进行 10 000 次破译的计算机来计算，则要经过 200 多年才能够破解，可见 DES 算法具有很好的保密效果。另外，DES 算法实现起来并不复杂，不但在软件中可以容易地实现，而且早已经在芯片上实现了，使用起来非常方便。

DES 算法的过程，简单来说，就是把要加密的明文分成 64 位的数据段作为输入，再使用根据 64 位密钥变化生成的 52 个子密钥，对输入的数据段依次进行初始置换、16 轮迭

代、逆初始置换，然后得到 64 位密文。

DES 的解密过程与加密过程几乎相同，只是子密钥的使用顺序不一样。加密时依次使用的部分参数 K1 K2 K3 …K16，在解密时则按照 K16 K15 K14 …K1 顺序使用。其他算法完全一样，这也是 DES 容易使用的一个方面。

2．IDEA 算法

IDEA 在加密运算中所处理的数据段大小也是 64 位，但是所用的密钥长度为 128 位，而且采用更加复杂的加密算法，目的是保证它不会被轻易破译。IDEA 是一种加密强度很高的加密算法，迄今为止还没有出现对该算法的有效攻击。假如一台计算机一秒钟可以产生和运行 10 亿个密钥，则要猜出 IDEA 密钥需要花费 1013 年的时间，可见 IDEA 的加密强度非常高。另外，IDEA 实现非常方便，既可以通过软件实现，也可以通过硬件实现。

IDEA 算法对数据的处理是以 64 位为单位的，在加密前把要加密的明文按每 64 位作为一个数据段进行分割然后分别加密。

IDEA 的解密过程与加密过程基本相同，所不同的就是解密子密钥的产生方式与加密子密钥的产生方式不一样，解密的其他运算过程同加密一样，也是把 64 位数据段分成 4 个 16 位的数据段，然后经过八轮迭代变换和一轮输出变换，就可以得到对应的明文结果。

16.1.2 不对称密钥加密算法

对称密钥加密方法是加密、解密使用同样的密钥，由发送者和接收者同时保存，在加密和解密时使用相同的密钥。采用这种方法的主要问题是密钥的生成、导入、存储、管理、分发等过程比较复杂，特别是随着用户的增加，密钥的需求量成倍增加。而在较大规模的信息

系统中，大量密钥的分配与管理是一个难以解决的问题。

例如，系统中有 n 个用户，其中每两个用户之间需要建立密码通信，则系统中每个用户须掌握(n-1)/2 个密钥，而系统中所需的密钥总数为 n*(n-1)/2 个。对 10 个用户的情况，每个用户必须有 9 个密钥，系统中密钥的总数为 45 个。对 100 个用户来说，每个用户必须有 99 个密钥，系统中密钥的总数为 4950 个。这还仅仅考虑用户之间的通信只使用一种会话密钥的情况，如果不同的会话需要变换不同的密钥，则密钥总数就更多了。如此庞大数量的密钥生成、管理、分发是一个难以处理的问题。

与对称密钥加密方法不同，不对称密钥加密技术在对信息进行加密和解密时，需要分别采用两个不同的密钥，因此也称为双钥加密方法。它在运算中，先产生一对密钥，其中之一是保密密钥，由用户自己保存，不能向外界泄漏，简称私钥；另一个为公开密钥，可对外公开，甚至可在公共目录中列示，简称公钥，因此也称公开密钥加密方法。

只有使用私钥才能解密用公钥加密的数据，同时使用私钥加密的数据只能用公钥解密。在通信过程中，如果发送者要向接收者发送保密信息，则需要先用接收者的公开密钥对信息进行加密，然后发送给该接收者，接收方用其私钥能够顺利解密。而其他人即使收到加密的密文也无法正确解读，从而达到保密通信的目的。

公开密钥加密方法中，要想达到良好的加密效果，算法上必须做到：在计算上产生密钥非常容易；已知公钥的情况下对明文加密在计算上很容易实现；已知私钥的情况下对密文解密在计算上很容易实现；尽管用于加密和解密的两个密钥在数学上是相关的，但是在已知公钥的情况下，要想求得私钥在计算上不可行；已知公钥和密文的情况下，要想求得明文在计算上不可行。只有做到以上几点，才能有效地防止攻击者对算法的破译。

不对称密钥加密算法有多种，例如，RSA、背包密码、McEliece、Diffe Hellman、Rabin、Ong Fiat Shamir、零知识证明的算法、椭圆曲线、EIGamal 等。这里主要介绍 RSA 的加密原理。

在众多的公钥加密算法中，以 1977 年由 Ron Rivest、Adi Shamir 和 Leonard Adleman 提出的以他们的名字命名的 RSA 加密算法最为著名。而且它是第一个既能用于数据加密也能用于数字签名的算法。RSA 从提出到现在已经二十多年，经历了各种攻击的考验，逐渐为人们所接受，被普遍认为是目前优秀的公钥加密方法之一。由于它易于理解和操作，因而获得了广泛的应用。但 RSA 的安全性一直未能得到理论上的证明。

RSA 的安全性依赖于大数的分解，即求得两个大数（例如，大于 100 位的十进制数）的乘积非常容易，但是要把一个大数分解为两个素数却非常困难。

在 RSA 加密体制中，每个用户有公开密钥 PK = (N，e)和私人密钥 SK = (N，d)，其中，N 为两个大素数的乘积，为了保密性更好，一般都取两个 100 位以上的大素数相乘得到 N。e 和 d 是根据一定运算法则计算得到的，虽然 N、e、d 之间存在一定的计算关系，但是攻击者根据 N、e 无法求解 d，从而实现不对称加密。

1．RSA 加密过程

首先把需要加密的明文按比特位分成等长的数据段，使得每个数据段对应的十进制数小于 N，即数据段的长度小于 log2N。然后依次对每个明文数据段 m 做加密运算可以得到密文 c：c = me mod N。

相应的，解密时对密文数据段做解密运算就可以得到明文 m：m = ce mod N。

2．RSA 数字签名

RSA 加密算法不仅可以用于信息的加密，而且还可以用于发送者的身份验证或数字签名。例如，用户 B 要向 A 发送一个信息 m，而且要让 A 确信该信息就是 B 本人发出的。为此，B 用自己的私钥 SK = (N，d)对信息加密得到密文 c：c = md mod N，然后把 c 发送给 A。A 收到密文后，使用 B 的公钥 PK = (N，e)对密文进行解密得到明文 m：m = ce mod N。这样，经过验证，A 可以确认信息 m 确实是 B 发出的，因为只有 B 本人才有与该公钥对应的私钥，其他人即使知道公钥，也无法猜出或计算出 B 的私钥来冒充他发送加密信息。

16.2 数字签名与数字水印

散列函数是一种公开的数学函数。散列函数运算的输入信息也可叫作报文。散列函数运算后所得到的结果叫作散列码或者叫作消息摘要。散列函数具有如下一些特点：

（1）不同内容的报文具有不同的散列码，而一旦原始报文有任何改变，哪怕改变一位信息，则通过散列函数计算后得到的散列码也将完全不同。这样，这个散列码就好比是这个报文所特有的“指纹”。

（2）散列函数是单向的，即求解某一个报文的散列码非常容易，但是根据散列码来倒推原始报文是非常困难的。

（3）对于任何一个报文，无法预知它的散列码。

（4）散列码具有固定的长度，不管原始报文的长度如何，通过散列函数运算后的散列码都具有一样的长度。例如，MD5（Message Digest Algorithm 5，消息摘要算法第 5 个版本）

散列算法的散列码长度为 128 位，并且不管是对一部百科全书，还是对某个人的工资进行 MD5 散列运算，得到的散列码长度都是 128 位。

由于散列函数具有这些特征，因此散列函数可以用来检测报文的可靠性。接收者对收到的报文用与发送者相同的散列函数进行运算，如果得到与发送者相同的散列码，则可以认为报文没有被篡改，否则，报文就是不可信的。

常见的散列函数有 MD5、SHA、HMAC 等。MD5 是一种非常著名的散列算法，已经成为国际标准，具有很好的安全性能。MD5 算法在对输入的报文进行计算时，是以 512 位为单位进行处理的，结果生成一个 128 位长的消息摘要；SHA、HMAC 等算法都是对任意长度的报文以 512 位为单位进行处理，最后得出一个 160 位的消息摘要。

16.2.1 数字签名

对于计算机系统中传送、存储的重要文件、数据、信息等，一般需要有某种方式来确认其真实性，即接收者能够确认自己得到的信息确实是由该信息所声称的发送者发出的，而不是由非法入侵者伪造、冒充发出的，并且还要能够保证信息在传送、存储中没有被恶意篡改，这样这份信息才能真实地反映发送方的意图。另外，对于发送方来说，如果发出一份信息，还必须有一定的措施阻止其否认自己发出信息的行为，即不可否认性。

只有做到以上几点，一个信息传送、存储系统才能够安全、可靠，其上所传送、存储的信息才是真实的、值得相信的。

举例来说，互有贸易往来的买卖双方之间通过计算机系统进行贸易，卖方通过计算机系统给买方发出一张电子报价单，买方收到后，擅自更改了收到的单价，并声称是卖方发出的，而且据此下订单，这就是篡改信息。显然安全的系统应该能够阻止这种行为。

要实现上述安全的系统，就离不开数字签名技术。

数字签名主要由两个算法组成：签名算法和验证算法。通过使用签名算法签名一个消息，所得到的签名能够通过一个验证算法来验证签名的真实性和有效性。

所以数字签名技术的大致过程就是：信息的发送方对信息利用自己的私钥进行签名，接着发送方把这个签名和信息一起发送给接收方。接收方收到信息后利用发送方的公钥来对其中的数字签名进行验证，确认其合法性。

目前已经有大量的数字签名算法，例如，RSA 数字签名算法、EI Gamal、Fiat-Shamir、Guillon-Oucsquerrter、DSS（Digital Signature Standard，数字签名标准）、DSA（Digital Signature

Algorithm，数字签名算法）、椭圆曲线等。

1．RSA 结合 MD5 数字签名

前面讲过，RSA 公钥加密技术本身就可以用来实现数字签名。但是仅仅使用公钥加密算法进行数字签名的运算量比较大，尤其是要传送的信息量比较大时，速度会更加慢。显然，直接用这种方法进行数字签名并不是很好的选择。

而散列算法（例如，MD5 算法）就有很好的特性，它能对每一个不同长度的信息产生相互不同的、独特的、简短的消息摘要。这个消息摘要可以看作这个信息特有的“指纹”，因而非常适合用作数字签名。

通过散列算法对原始数据进行散列，再对散列码进行公钥加密就可以很好地实现数字签名。它的特点是：它代表了文件的特征，具有唯一性。只要文件发生哪怕一位数据的改变，或者签名者有任何差别，数字签名的值也将随之而发生改变；不同的文件和签名者得到的是不同的数字签名。

RSA 结合 MD5 数字签名的主要过程是：信息的发送方通过对信息进行散列运算生成一个消息摘要，接着发送方用自己的私钥对这个消息摘要进行加密，就形成发送方的数字签名。然后，把这个数字签名作为信息的附件和信息一起发送给信息的接收方。接收方收到信息后，首先对收到的信息进行与发送者相同的散列运算得到一个消息摘要，接着再用发送方的公钥来对信息中附加的数字签名进行解密得到发送方计算出的散列码。如果两个散列码相同，那么接收方就能确认该信息和数字签名是由发送方发出的。通过数字签名能够实现对原始信息完整性的鉴别和发送方发送信息的不可抵赖性。

下面结合一个例子，看一看 RSA 结合 MD5 数字签名的具体步骤：

（1）信息发送者 A 要向 B 发送一份信息，A 先按双方约定的散列算法对该信息进行散列运算，得到一个该信息特有的消息摘要 H，从前面所述可以知道，只要改动信息中任何一位，重新计算出的消息摘要值就会与原先的值不相符。这样就保证了信息的不可更改性。

（2）接着把该消息摘要用 A 自己的私钥加密，得到 A 对该信息的数字签名 S。

（3）然后 A 把信息原文与数字签名 S 一起发送给 B。

（4）当 B 收到后，先用 A 的公钥对数字签名 S 解密得到 A 的消息摘要 H。

（5）再用同样的散列算法对收到的信息进行散列运算，得到消息摘要 H'。

（6）比较 H 与 H'，如相等则说明信息确实来自它所声称的发送者 A。

在传输过程中，如有攻击者对文件进行了篡改，但他并不知道发送方的私人密钥，因此，接收方解密得到的数字签名 H 与经过计算后的数字签名 H'必然不同。这就提供了一个安全

的确认发送方身份的办法。

当然，以上例子中，对传送的信息是以明文出现的，不具有保密意义。实际应用中，还需要对信息本身运用适当的保密措施。

RSA 用于数字签名的一个重要的特点是能够证实信息发送方的身份及电子文件的可靠性和完整性，它对于发送方和被发送的信息都是独一无二的，具有可验证性和不可否认的权威性特点；另一个重要的特点是它通过在计算机之间交换数字证书就可以确定当事者就是他们所宣称的人。

2．数字签名标准

DSS 是美国国家标准与技术学会的数字签名标准，自 1991 年提出以来又经过广泛的修改。DSS 为计算和验证数字签名指定了一个数字签名算法——DSA。DSA 是 El Gamal 数字签名算法的一个改进版本，它通过选择较小规格的参数减少数字签名的数据量，从而减少了存储空间和传输带宽。

DSS 中指定 SHA 作为其散列算法，它对原始信息进行运算后产生 160 位的消息摘要，然后 DSS 把这一消息摘要与一个用作这个特殊签名的随机数作为输入送到数字签名算法中，经过运算生成数字签名。

该数字签名函数还依赖于发送方的私钥 SK 和一个对许多通信方都公开的由重要的公钥集合组成的全局公钥。

接收方在收到消息摘要和签名后将其作为验证函数的输入。验证函数还依赖于全局公钥和与发送方的私钥相匹配的公钥 PK，这样只有发送方用其自己的私钥才能产生有效的签名。

数字签名作为一项重要的鉴别技术，近年来越来越受到人们的重视，在政府、军事、金融、安全等领域得到广泛的运用。通过数字签名可以有效地保证数据的完整性，防止第三方伪造或发送方的抵赖。

2004 年 8 月 28 日，十届全国人大常委会第十一次会议表决通过了电子签名法。这部法律规定，可靠的电子签名与手写签名或者盖章具有同等的法律效力，并于 2005 年 4 月 1 日起施行。这部法律将对我国电子商务、电子政务等计算机信息系统的发展起到极其重要的促进作用。

16.2.2 数字信封

数字信封是公钥密码体制在实际中的一个应用，是用加密技术来保证只有规定的特定收

信人才能阅读通信的内容。

在数字信封中，信息发送方采用对称密钥来加密信息内容，然后将此对称密钥用接收方的公开密钥来加密（这部分称数字信封），之后，将它和加密后的信息一起发送给接收方，接收方先用相应的私有密钥打开数字信封，得到对称密钥，然后使用对称密钥解开加密信息。这种技术的安全性相当高。

16.3 数字证书与密钥管理

过去，人们总是依赖于对于加密算法和密钥的保密来增加保密的强度和效果。随着现代密码学的发展，大部分的加密算法都已经公开了。一些典型的算法（例如，DES、IDEA、 RSA等）更是成了国际标准，被广泛接纳。人们可以从多种途径来获取算法的细节，也已经有很多采用这些算法的软件、硬件设备可以利用。

因此，在现代密码系统中，算法本身的保密已经不重要了，对于数据的保密在很大程度上，甚至完全依赖于对密钥的保密。只要密钥能够保密，即使加密算法公开，甚至加密设备丢失，也不会对加密系统的坚固性和正常使用产生多大影响。相反，如果密钥丢失，则非法用户可以窃取机密数据，而合法用户却面对密文如读天书，无法提取有效的信息。与其如此，还不如不加密呢！因此，在密码系统中，如何高效地分配密钥、安全地管理密钥对于保证数据安全至关重要。

16.3.1 密钥分配中心

鉴于密钥的举足轻重的地位，密钥必须通过安全的通路进行分配。例如，可以派非常可靠的信使，以十分安全的方式，物理地携带密钥，人工送达需要进行通信的各方。这种方法虽然可靠，但是效率比较低，在过去信息技术不发达的时候用得比较多。而对于一个用户比较多、相互之间通信比较频繁的信息系统来说，则显然不适合采用这种人工方法，而必须采用计算机自动分配密钥。

一个方法是在一个信息系统中任意两个用户之间自己协商来选择不同的密钥，如图16-1所示。

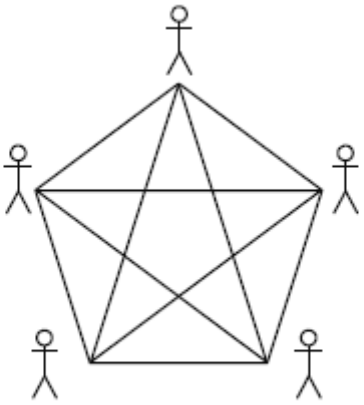


图 16-1　用户自己协商密钥的密钥管理方式

显然，对于有 N 个用户的这种通信系统中，每个用户都要保存（N−1）个密钥，系统中总共要保存 N∗(N−1)/2 个密钥。在用户数量较少时，这样来分配密钥还是比较简单、易用的，但是一旦用户数量多起来，系统中要保存的密钥会急剧增多，让每个用户自己高效、安全地管理数量庞大的密钥实际上是不可能的。

此外有一种非常有效的密钥自动分配方案是密钥分配中心（Key Distribution Center，密匙分配中心）技术。

在 KDC 方案中，每一个用户都只保存自己的私钥 SK 和 KDC 的公钥 PKKDC，而在通信时再经由 KDC 获得其他用户的公钥 PK 或者仅仅在某一次通信中可以使用的对称密钥加密算法的临时密钥 K。

假设有两个用户 A、B 都是 KDC 的注册用户，他们分别拥有私钥 SKA 和 SKB，相应的公钥分别是 PKA 和 PKB。现在 A 想要与 B 进行会话，假如采用对称密钥加密算法来加密这次会话，那么密钥的分配过程如图 16-2 所示。

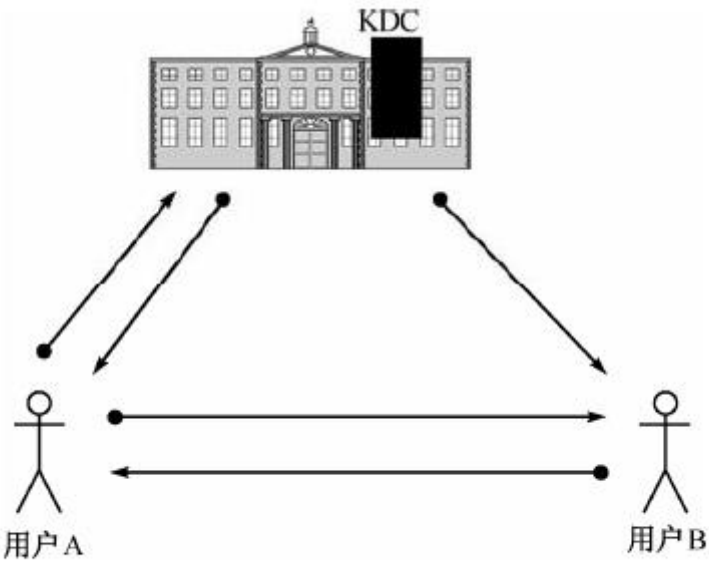


图 16-2　密钥分配中心密钥管理方式

首先用户 A 向 KDC 发送用自己的私钥 SKA 加密的请求 SKA(A，B)，说明自己想要与 B 进行会话。

KDC 收到这个请求后，根据某种算法生成一个可以供 A、B 双方进行秘密会话的对称密码算法的密钥 K，然后向 A 返回一个应答 PK(K, PKB(A, K))。这个应答信息是用 A 的公钥 PKA 加密的，当然只有用户 A 自己才能够正确解读，可以从中提取与 B 会话的密钥 K。

同时，该信息中还有一部分内容 PKB(A，K)，表明用户 A 欲与 B 进行会话，并且密钥是 K，这是用 B 的公钥 PKB 加密的。用户 A 把这一部分信息发送给 B，B 收到后从中解密出会话密钥。

至此，完成一次密钥的自动分配过程。此后，A、B 双方就可以利用密钥 K 进行加密通信了。

16.3.2　数字证书和公开密钥基础设施

公钥加密算法的密钥分配和对称密钥加密算法中密钥的分配要求有着很大的区别。在对称密钥加密体制中，要求将密钥从一方传送到另一方，并且保证只有通信的双方知道密钥，而不让其他任何一方知道密钥。

而在公钥加密体制中，则要求通信各方的私钥只有通信的一方知道，而其他任何一方都不能知道，同时每一方的公钥需要公开，其他任何一方都可以查看和提取。

在公钥加密体制中，私钥的分配相对容易，但是，公钥的发布和获取就需要采取合适的方法来进行，否则很容易留下安全漏洞。　　一种简单的发布公钥的方法是公开宣布。通

信系统中的每一方都独自保管好自己的私钥，而把自己的公钥公开地公布给其他所有各方，以使其他人能够得到他的公钥，从而可以与他进行加密通信。这实现起来非常简单，似乎也没有什么问题。但是，却有一个致命的漏洞，就是任何一个非法入侵者也可以冒充是这个通信系统中的一方，向这个通信系统中公布一个冒充的公钥。此后系统中与该用户的通信实际上就是与该非法冒充者进行通信。

数字签名和公钥加密都是基于不对称加密技术，因此也存在这样的问题：如何保证公开密钥的持有者是真实的；大规模信息系统环境下公开密钥如何产生、分发和管理。

要解决上述问题，就要用到数字证书和 PKI（Public Key Infrastructure，公开密钥基础设施）。

1．数字证书

数字证书提供了一个在公钥和拥有相应私钥的实体之间建立关系的机制。目前最常用的数字证书格式是由国际标准 ITU-T X.509 v3 版本定义的。

数字证书中采用公钥体制，即利用一对互相匹配的密钥进行加密、解密。每个用户自己保存私钥，用它进行解密和签名；同时设定一个公钥，并由本人公开，为一组用户所共享，用于加密和验证签名。

数字证书是用户在系统中作为确认身份的证据。在通信的各个环节中，参与通信的各方通过验证对方数字证书，从而确认对方身份的真实性和有效性，从而解决相互间的信任问题。

数字证书的内容一般包括：唯一标识证书所有者的名称、唯一标识证书签发者的名称、证书所有者的公开密钥、证书签发者的数字签名、证书的有效期及证书的序列号等。

2．公开密钥基础设施

PKI 在信息系统中的作用就相当于作为公共设施在社会生活中的作用，其目标是向广大的信息系统用户和应用程序提供公开密钥的管理服务。PKI 是指由数字证书、证书颁发机构（Certificate Authority，CA），以及对电子交易、通信等所涉及的各方的合法性进行检查和验证的其他注册机构组成的一套系统。为了使用户在不可靠的网络环境中获得真实可靠的公开密钥，PKI 引入公认可信的第三方；同时 PKI 中采用数字证书机制来避免在线查询集中存放的公开密钥产生的性能瓶颈。可信的第三方是 PKI 的核心部件，系统中任意两个实体之间都是通过公认可信的第三方建立安全联系的。数字证书中第三方的数字签名，使用户可以离线地确认一个公开密钥的真实性。

除了数字证书的有效期，证书撤销列表（Certificate Revocation List，CRL）是另一种数字证书有效期控制机制。当数字证书中认可的事实发生变化时，数字证书发布者必须使用某种

机制来撤销以前发出、但现在失效的证书。证书发布者定期发布 CRL，列出所有曾发布但当前已被撤销的证书号，证书的使用者依据 CRL 即可验证某证书是否已被撤销。

（1）PKI 的结构模型。PKI 中有三类实体：管理实体、端实体和证书库。管理实体是 PKI 的核心，是 PKI 服务的提供者；端实体是 PKI 的用户，是 PKI 服务的使用者；证书库是一个分布式数据库，用于证书和 CRL 的存放和检索。

CA 和注册机构（RegisteAuthority，RA）是两种管理实体。CA 是 PKI 框架中唯一能够发布和撤销证书的实体，维护证书的生命周期；RA 负责处理用户请求，在验证了请求的有效性后，代替用户向 CA 提交。RA 可以单独实现，也可以合并在 CA 中实现。作为管理实体，CA 和 RA 以证书方式向端实体提供公开密钥的分发服务。

持有者和验证者是两种端实体。持有者是证书的拥有者，是证书所声明的事实上的主体。持有者向管理实体申请并获得证书，也可以在需要时请求撤销或更新证书。持有者使用证书声明自己的身份，从而获得相应的权力。验证者确认持有者所提供的证书的有效性和对方是否为该证书的真正拥有者，只有在成功鉴别之后才可与对方进行更进一步的交互。

证书库可以用 Web、FTP 或目录等来实现。由于证书库中存取的对象是证书和 CRL，其完整性由数字签名保证，因此对证书库的操作可在无特殊安全保护的通道上传输。

不同的实体间通过 PKI 操作完成证书的请求、确认、发布、撤销、更新和获取等过程。PKI 操作分成存取操作和管理操作两类。其中，存取操作包括管理实体或端实体，把证书和 CRL 存放到证书库、从证书库中读取证书和 CRL；管理操作则是管理实体与端实体之间或管理实体与管理实体之间的交互，是为了完成证书的各项管理任务和建立证书链。

（2）PKI 层次模型。PKI 框架可以分为三个层次。最低层是传输层，向上提供 PKI 报文的可靠传输，它可以是传输层协议或应用层协议。中间层是密码学服务层，向上提供加密、解密、数字签名、消息摘要等基本密码学服务，可由 RSA、MD5 等方法实现。最高层是证书服务层，使用前面两层提供的加密和传输服务，向用户提供证书的请求、签发、发布、撤销和更新等服务。

PKI 的三类实体对这三层服务的使用各不相同。证书库不需要特殊的安全交互措施，所以仅使用传输层服务来分发证书和 CRL；管理实体和端实体使用证书服务层构造 PKI 证书，使用密码学服务层来鉴别和保护交互信息，使用传输层服务传送信息。

（3）X.509 数字证书。ISO/ITU、ANSI、IETF 等组织制定的 X.509 标准，对数字证书的格式进行了专门定义，该标准是为了保证使用数字证书的系统间的互操作性而制定的。理论上为一种应用创建的 X.509 证书可以用于其他任何符合 X.509 标准的应用。但实际上，不

同的公司对 X.509 证书进行了不同的扩展，并不是所有的证书都彼此兼容。

X.509 证书具有如下一些突出的特点：

① 支持多种算法。X.509 证书独立于算法，CA 可以根据需要选择证书的签名和摘要算法，以及端实体所拥有密钥对的类型。摘要算法有 MD2、MD5 和 SHA-1，证书签名算法有 RSA 和 DSA，密钥对类型有 RSA 密钥、DSA 签名密钥、D-H 密钥交换密钥、KEA 密钥和 ECDSA 密钥。

② 支持多种命名机制。X.509 证书除了使用 X.500 名字机制标识持证者和验证者，还支持 E-mail 地址、IP 地址、DNS 名和 URI。

③ 可以限制证书（公开密钥）的用途。CA 能够规定证书的使用范围，如签名、不可否认、密钥加密、数据加密、密钥协商、证书签发和 CRL 签发等。

④ 定义证书遵循的策略。每个 CA 都定义了一定的安全策略，规范证书的操作过程。这些策略包括：CA 的命名空间、身份验证、撤销机制、法律责任和收费等。

⑤ 控制信任关系的传递。建立 CA 体系，跨域认证，使得每个 CA 除负责本域的证书管理任务外，还要维护与其他 CA 间的信任关系。X.509 证书定义若干字段用于控制信任关系的传递，CA 能够将自己管理域的安全策略体现在信任关系中。

可见，X.509 证书适用于大规模信息系统环境，它的灵活性和扩展性能够满足各种应用系统不同类型的安全要求。

X.509 有不同的版本，例如，X.509v3 是比较新的版本，它是在原有版本 X.509 v 的基础上进行功能的扩充。每一版本都包含下列数据项：

① 版本号。用来区分 X.509 的不同版本号。

② 序列号。由 CA 给每一个证书分配唯一的数字型编号，由同一 CA 发放的每个证书的序列号是唯一的。

③ 签名算法识别符。用来指定 CA 签发证书时所使用的公开密钥算法和 HASH 算法，须向国际标准组织注册。

④ 发行者名称。建立和签署证书的 CA 名称。

⑤ 有效期。证书有效的时间包括两个日期：证书开始生效的日期、证书失效的日期和时间。在所指定的这两个时间之间有效。

⑥ 主体名称。证书持有人的姓名、服务处所等信息。

⑦ 主体的公开密钥信息。包括主体的公开密钥、使用这一公开密钥的算法的标识符及相应的参数。

⑧ 发行者唯一识别符。这一数据项是可选的，当 CA 名称重新用于其他实体时，则用这一识别符来唯一标识发行者。

⑨ 主体唯一标识符。这一数据项也是可选的，当主体的名称重新用于其他实体时，则用这一识别符来唯一识别主体。

⑩ 扩充域。其中包括一个或多个扩充的数据项。

⑪签名。CA 用自己的私钥对上述各数据项的散列值进行数字签名的结果。

16.4 安全协议

Internet 是 IT 领域中发展的重大成就，它的迅速发展和全面普及给人们的生产、生活带来了很大的帮助。

但是，Internet 在当初是为了让更多的人来使用网络、共享资源，并且容易扩充、容易治理等而设计的，因此它是一个全面开放的系统，而没有在安全方面作充分的考虑。加上日益增加的庞大的用户、各种不同的动机等因素，使得 Internet 上的安全事件层出不穷。

在 Internet 安全中，网络通信的安全是一个非常重要的环节，因此有必要研究在网络上安全传输数据的方法。

16.4.1 IPSec 协议简述

在 TCP/IP 协议中，对 IP 数据包没有提供任何安全保护，攻击者可以通过网络嗅探、IP 欺骗、连接截获等方法来攻击正常的 TCP/IP 通信。因此，通信过程中会存在以下危险：数据并非来自合法的发送者、数据在传输过程中被非法篡改、信息内容已被人窃取等。

为了确保在 IP 网络上进行安全保密的通信，IETF 制定了一套开放标准的网络安全协议 IPSec（IP Security）。该协议把密码技术应用在网络层，以向信息的发送方和接收方提供源地址验证、数据传输的完整性、存取控制、保密性等安全服务，保护通信免遭窃听、抵御网络攻击，而且更高层的应用层协议也可以直接或间接地使用这些安全服务，为其上层协议如 TCP、UDP 等提供透明的安全保护服务，在 Internet 这样不安全的网络中为通信提供安全保证。

在 IPv6 中，IPSec 协议是一个必备的组成部分，被强制实施；在 IPv4 中，它是一个可选的扩展协议。

由于 Internet 等网络具有公共特性，因此在通信过程中难以确认传输媒介是安全的，

所以要进行安全的通信，则通信数据必须经过加密。IPSec 协议对数据的加密以数据包而不是整个数据流为单位，这不仅非常灵活，也有助于进一步提高 IP 数据包的安全性。

IPSec 协议的基本工作原理是：发送方在发送数据前对数据实施加密，然后把密文数据发送到网络中去，开始传输。在整个传输过程中，数据都是以密文方式传输的，直到数据到达目的节点，才由接收方对密文进行解密，提取明文信息。

IPSec 协议对网络层的通信使用了加密技术，它不是加密数据包的头部和尾部信息（如源地址、目的地址、端口号、CRC 校验值等），而是对数据包中的数据进行加密。由于加密过程发生在 IP 层，因此可在不改变 HTTP 等上层应用协议的情况下进行网络协议的安全加密，为通信提供透明的安全传输服务。

IPSec 协议中使用端到端的工作模式，掌握加密、解密方法的只有数据的发送方和接收方，两者各自负责相应的数据加密、解密处理，而网络中其他节点只负责转发数据，无须支持 IPSec，从而可以实现加密通信与传输媒介无关，保证机密数据在公共网络环境下的适应性和安全性。因此，IPSec 可以应用到非常广泛的环境中，能为局域网、拨号用户、远程站点、Internet 之上的通信提供强有力的保护，而且还能用来筛选特定数据流，还可以用于不同局域网之间通过互联网的安全互联。

IPSec 协议不是一个单独的协议，它包括应用于 IP 层上网络数据安全的一整套协议，主要包括 AH（Authentication Header，IP 认证头部协议）、ESP（Encapsulating Security Payload，封装安全负载协议）、IKE（Internet Key Exchange，Internet 密钥交换协议）和用于网络认证及加密的一些算法等。

AH 提供数据的完整性和认证，但不包括保密性；而 ESP 原则上只提供保密性，但也可在 ESP Header 中选择适当的算法及模式来实现数据的完整性和认证。AH 和 ESP 可分开使用也可一起使用。IKE 则提供加密算法、密钥等的协商。

1．安全关联和安全策略

安全关联（Security Association，SA）是指提供通信安全服务的发送方和接收方之间的一种单向关系。安全关联是构成 IPSec 的基础，它是进行通信的双方经协商建立起来的一种协定。安全关联可以用一个 32 位的安全参数索引（Security Parameter Index，SPI）来唯一标识，一个 SPI 值决定一个特定的 SA，它通常放在 AH 或 ESP 头中；安全关联是单向的，如果要对两台主机 A 与 B 实现双向安全，则需要两个安全关联，每个方向一个：（A，B）、（B，A）。安全关联的内容包含了 IP 数据包是否加密、认证，以及加密、认证采用的算法、密钥等相关信息。所有的 SA 记录都存放在安全关联数据库中，按散列方式存取。

安全策略（Security Policy）定义了两个 IPSec 系统之间的安全通信特征，并决定在该通信中为数据包提供的安全服务。一个 IPSec 系统的所有安全策略都存放在安全策略数据库中，根据选择符（包括源地址、目的地址、协议、端口等）进行检索。安全策略通常与 SA 合作，共同作用于通信的数据包。

2．AH

AH 协议先将数据进行校验和加密，然后封装为 IP 包，从而实现无连接通信的数据完整性、数据源认证和防止重放攻击。AH 能完成除数据加密外的所有的 ESP 所能提供的功能。在认证机制上，它所覆盖的范围比 ESP 的广，包括对 IP 头中一些选项的认证。

为了应用 IPSec 协议，IP 数据包的格式要有所改变，即在 IP 头和被保护的数据之间插入一个 AH 头，如图 16-3 所示。

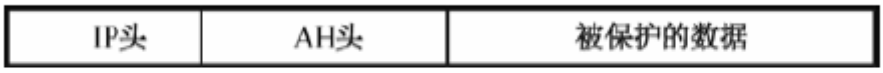


图 16-3　用 AH 保护的 IP 数据包格式示意图

AH 头的格式如图 16-4 所示，包括：下一报头、有效载荷长度、保留位、安全参数索引、序列号、认证数据。

图 16-4　AH 头的格式

AH 使用的典型的认证算法是一种迭代型的消息摘要算法。AH 中采用 MD5 算法，可以提供完整性服务。从前面的讲述可以知道 MD5 可以对任意长度的信息进行散列运算产生一个唯一的 128 位消息摘要。由于消息摘要是唯一的，所以对信息的任何修改都将得到另一个不同的消息摘要，因此能防止消息被篡改，从而保证了数据的完整性。AH 也可以采用 SHA 算法提供更强的抗攻击能力，SHA 是在 MD5 的基础上，增加了分组处理的迭代次数和复杂性，产生一个 160 位的消息摘要。接收者在收到数据后可以通过检验数据包中的单向递增的序列号来确定数据包的合法性，防止重放攻击。

3．ESP

ESP 通过对数据包的数据进行加密来提供传输信息的保密性，从而实现了数据完整性、

数据源认证、数据保密性的安全服务。ESP 是一个通用的、可扩展的安全机制，其加密认证算法主要由 SA 的相应数据项决定。接收者也可以通过在收到数据后检验数据包中的单向递增的序列号来确定数据包的合法性，防止重放攻击。

在应用中，需要在 IP 数据包的头和被保护的数据之间插入一个 ESP 头，在被保护的数据后附加一个 ESP 尾，如图 16-5 所示。

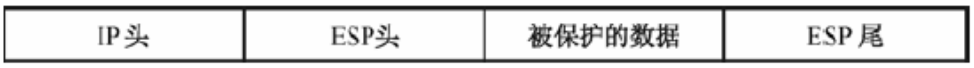


图 16-5　用 ESP 保护的 IP 数据包示意图

ESP 头的格式如图 16-6 所示，包括：安全参数索引（标识用于处理数据包的安全关联）、序列号（用于防止重放攻击）、有效荷载数据。ESP 头的所有字段都是不加密的，因为在解密数据包时需要先读取头部字段。

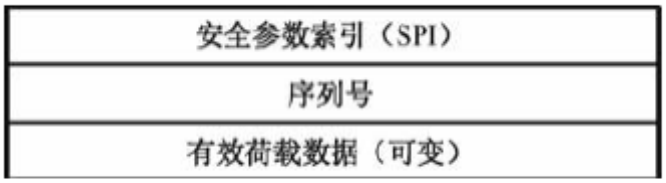


图 16-6　ESP 头的格式

ESP 尾的格式如图 16-7 所示，包括：填充项（某些加密算法要求被加密数据的长度是密钥长度的整数倍，若受保护的数据的长度不满足这个要求，就需要在后面追加一些填充项）、填充项长度（指明填充项的长度）、下一个头部、认证数据（数据完整性的检验结果）。

图 16-7　ESP 尾的格式

ESP 在提供加密功能的同时，还可以提供认证功能。对于发出的数据包，首先进行加密处理；而对于收到的数据包，则先进行认证处理。

ESP 支持多种加密算法。DES 是 ESP 中默认的加密算法，它采用 64 位的密钥，对明文进行加密，加密、解密使用同一个密钥，该算法简单高效。此外还可以选择采用 3DES、AES、RC5、RC6、Blowfish 等算法。

4．IP 密钥交换

IKE 是一个混合协议，它使用了 Internet 安全关联和密钥管理协议（Internet Security Association and Key Management Protocol，ISAKMP）、密钥确定协议 Oakley 和描述支持匿名和快速密钥刷新的密钥交换的 SKEME 协议。IKE 除了实现通信双方的密钥交换，还使用 ISAKMP 实现 IPSec 的安全关联。

ISAKMP 协议是 IKE 的核心组成部分，它定义了包括协商、建立、修改、删除安全关联的过程和数据格式。ISAKMP 的工作分为两个阶段：第一阶段，通信双方协商并建立一个安全的通道，并对该通道进行验证，为第二阶段的进一步通信提供安全服务；第二阶段，为 IPSec 建立起具体的 IPSec 安全关联，用于保护通信双方的数据传输安全。在 IKE 的协商过程中，使用了 Diffie-Hellman 机制、Oakley 的密钥交换模式和 SKEME 的共享和密钥更新技术。

5．IPSec 的工作模式

IPSec 的工作模式有两种：传输模式和隧道模式。

传输模式首先将要传送的数据使用 IPSec 加密封装起来，再把相关的 IPSec 头插入 IP 头和被保护的数据之间封装起来。因为 IP 头没有加密，接收端收到封装的数据包时直接处理 IP 头，然后从 IPSec 头读取 SPI 值得到相对的 SA，再利用 SA 所定的解密参数解出所加密的数据。

传输模式的 IPSec 头直接加在欲传送的数据前，由于加密的部分较少，没有额外的处理，因此比较节省带宽和 CPU 负载，通信和处理效率较高。

在传输模式中，解密者就是目的地址端的使用者。

隧道模式首先使用 SA 的相关信息将 IP 的数据包全部加密，接下来在前面加上 ESP Header，然后把它们作为数据为它们再加上一个新的 IP 头。接收端收到 ESP 封包后，使用 ESP Header 内容中的 SPI 值提供的 SA，然后解出 ESP Header 后的装载数据，就可以取回原始的 IP 头与封包。

隧道模式可以在两个终端之间建立一个安全的隧道，经由这两个终端之间的通信均在这个隧道中进行，因此安全性较高。

两种模式的 IP 数据包的格式如图 16-8 所示。

图 16-8　不同传输模式下的 IP 包

16.4.2 SSL 协议

SSL 是用于安全传输数据的一种通信协议。它采用公钥加密技术、对称密钥加密技术等保护两个应用之间的信息传输的机密性和完整性。但是，SSL 也有一个不足，就是它本身不能保证传输信息的不可否认性。

SSL 协议包括服务器认证、客户认证、SSL 链路上的数据完整性、SSL 链路上的数据保密性等几个方面，通过在浏览器和 Web 服务器之间建立一条安全的通道来保证 Internet 数据传递的安全性。目前，利用公钥加密的 SSL 技术，已经成为 Internet 上进行保密通信的工业标准。SSL 协议常常用于增强 Web 服务的安全性。

在 TCP/IP 协议中，SSL 协议建立在传输层即 TCP 之上、应用层之下。SSL 协议有一个突出的优点，就是它与应用层协议相独立，高层的应用层协议如 HTTP 等可以透明地建立在 SSL 协议之上进行工作。

通过 SSL 协议建立的传输通道具有如下的基本安全性：

（1）通道是保密的，经过握手确定密钥之后，所有的消息被加密。SSL 协议在应用层协议工作之前就已经完成了加密算法、密钥的协商、服务器认证等工作，而此后的所有应用层所传送的数据都是经过加密的，因此 SSL 协议具有很好的保密性。

（2）通道是被认证的，通信中的服务器端总是被认证，客户端可选认证。在基于 SSL 协议的通信过程中，服务器端认证是必须进行的，所以，即使在一次会话过程中不进行客户端认证，该会话的确认性也能够有很好的保证。

（3）通道是可靠的，用 MAC 对传送的消息进行完整性检查，保证通道上数据的完整性。基于 SSL 协议的通信过程，因为传递的消息中包括消息完整性检查数据（即 MAC 数据），因此，可以保证该通信是可靠的。

SSL 协议由 SSL 记录协议、SSL 握手协议、SSL 密码变更说明协议、SSL 警告协议等组

成。其架构如图 16-9 所示。

图 16-9　SSL 协议

1．SSL 记录协议

在 SSL 记录协议中，所有要传输的数据都被封装在记录中，记录是由纪录头和长度不为 0 的记录数据组成的。所有的 SSL 通信，包括握手消息、安全空白记录、应用数据等都需要使用 SSL 记录。

2．SSL 协议记录头格式

SSL 协议记录头格式如图 16-10 所示。

SSL 协议记录头包括的数据有记录头长度、记录数据长度、记录数据中是否有粘贴数据等。SSL 协议记录头长度既可以是 2 字节、也可以是 3 字节长。当记录头的最高位为 1 时，表示不含有粘贴数据，记录头长度为 2 字节，记录数据最大长度为 32 767 字节；当记录头的最高位为 0 时，则含有粘贴数据，记录头长度为 3 字节，记录数据最大长度为 16 383 字节。当记录头的最高位为 0 时，次高位有特殊的含义。当次高位为 1 时，表示所传输的记录是普通记录；当次高位为 0 时，表示所传输的记录是安全空白记录。

图 16-10　SSL 协议记录头格式

记录头中数据长度编码不包括数据头所占用的字节长度。记录头长度为 2 字节时记录长度的计算方法为：

记录长度=((byte[0]&0x7f) <<8))|byte[1]

记录头长度为 3 字节时记录长度的计算方法为：

记录长度=((byte[0]&0x3f) <<8))|byte[1]

以上计算式中，byte[0]、byte[1]也分别表示所传输的第一、二个字节。

另外，粘贴数据的长度为传输的第三个字节。

3．SSL 记录数据的格式

SSL 记录数据包含三个部分：MAC 数据和实际数据和粘贴数据。

MAC 数据用于数据完整性检查。计算 MAC 所用的散列函数由握手协议中的消息确定。若使用 MD5 算法，则 MAC 数据长度是 16 字节。MAC 数据的产生方式为：

MAC 数据= HASH（密钥、实际数据、粘贴数据、序号）

其中，当会话的客户端发送数据时，密钥是客户的写密钥（服务器用读密钥来验证 MAC 数据）；而当会话的客户端接收数据时，密钥是客户的读密钥（服务器用写密钥来验证 MAC 数据）。序号是一个可以被发送和接收双方递增的计数器。每个通信方都会建立一个计数器，分别属于发送者和接收者。计数器有 32 位，计数值循环使用，每发送一个记录计数值递增一次，序号的初始值为 0。

4．SSL 握手协议

SSL 握手协议建立在 SSL 记录协议之上，用于在实际的数据传输开始前，通信双方进行身份认证、协商加密算法、交换加密密钥等。SSL 握手的过程可以分为两个阶段，第一阶段用于建立秘密的通信信道，第二阶段用于客户验证。

在 SSL 协议中，同时使用了对称密钥加密算法和公钥加密算法，这是为了综合利用对称密钥加密算法的高速度和公钥加密算法的安全性的优点。SSL 协议使用公钥加密算法使服务器端身份在客户端得到验证，并且传递用于会话中对数据加密的对称密钥。然后再利用对称密钥在通信过程中对收到和发送的数据进行比较快速的加密，从而减小系统开销，保证通信效率。

SSL 支持各种加密算法。在“握手”过程中，使用 RSA 公开密钥系统。密钥交换后，可以使用多种密码，例如，RC2、RC4、IDEA、DES、3DES 及 MD5 信息摘要算法等。

SSL 协议可以非常有效地保护通信过程。但是，如果某种攻击是利用 SSL 协议通信进行的，那么，这种攻击也会受到 SSL 协议的保护，从而使得攻击更加隐蔽，难于被发现。当然，这种攻击也能够很好地穿透防火墙、躲过入侵检测系统的检查。

另外，SSL 在通信过程中，要进行许多加密、解密的操作，这些计算的复杂性随着密码的强度不同而不同，但是高强度的计算会增加服务器负载、增加网络带宽，从而使服务器性

能下降，吞吐量也下降。

16.4.3 PGP 协议

在信息时代里，电子邮件已经成为人们生活中的一部分，同时电子邮件的安全问题也就日益显得突出。一般来说，电子邮件在网络上的传输是不加密的。这种不加保护的邮件在网络上传输，第三者就会轻易获得通信过程中传送的信息。此外，为了防止冒名顶替，收信人需要确认邮件没有被第三者篡改，确实是发送者本人发出的，这就需要使用数字签名的一些技术。从前面的讲述可以知道，RSA 公钥密码体系非常适合用来满足上述要求。但是要直接使用 RSA 加密电子邮件，还有一些不方便的地方。

PGP（Pretty Good Privacy）是美国人 PhilZimmermann 于 1995 年提出的一套电子邮件加密方案。它可以用来对邮件加密以防止非授权者阅读，还能对邮件加上数字签名而使收信人可以确认邮件确实是由发送方发出的。

PGP 并不是新的加密算法或协议，它综合采用了多种加密算法，例如，对邮件内容加密采用 IDEA 算法、对于加密信息采用 RSA 公钥加密算法，还采用了用于数字签名的消息摘要算法，加密前进行压缩处理等技术手段进行邮件加密的一套软件。通过组合使用这些加密方法，把 RSA 公钥加密体系的良好加密效果和对称密钥加密体系的高速度结合起来，并且通过在数字签名和密钥认证管理机制中的巧妙设计，使得 PGP 成为一个优秀的强有力的数据加密程序。

由于 PGP 功能强大、处理迅速、使用简便，而且它的源代码是免费的，因此，PGP 在 IT 等多个行业得到了广泛的应用，迅速普及。如今，PGP 除了用于通常的电子邮件加密，还可以用来加密重要文件，用 PGP 代替 UUencode 生成 RADIX64 格式（就是 MIME 的 BASE64 格式）的编码文件，以保证它们在网络上的安全传输，或为文件做数字签名，以防止篡改和伪造。

1．PGP 加密的原理

假设一个用户 A 想要发送一个加密的邮件给另一个用户 B。那么加密的过原理如图 16-11 所示。

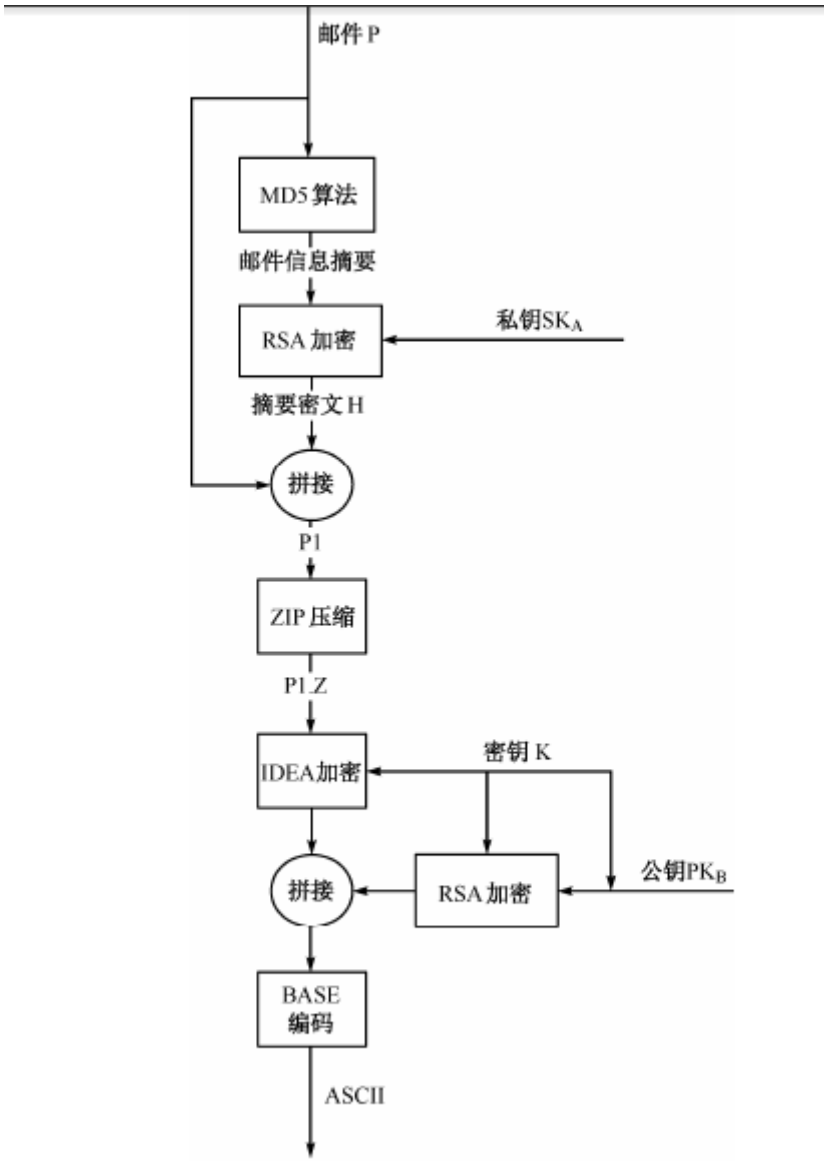


图 16-11　PGP 加密的原理

首先，用户 A 对要发送的邮件 P 运用 MD5 散列算法进行计算，生成一个 128 位的消息摘要，有了这个消息摘要就可以检验邮件信息是否完整、有没有被篡改。然后再通过 RSA 算法，运用 A 的私钥 SKA 对消息摘要进行加密，生成消息摘要的密文 H。邮件 P 与经过加密的邮件消息摘要 H 共同构成新的报文 P1，接着对 P1 进行 ZIP 压缩，成为压缩的报文 P1.Z。再对 P1.Z 采用 IDEA 算法加密，这次加密使用一个一次性的密钥 K，并且 K 必须经过 RSA 算法使用通信的另一方 B 的公开密钥 PKB 加密，与加密后的报文 P2 一起，再经过 BASE64 编码，得到一系列 ASCII 码，作为邮件内容发送到网络上。

用户 B 接收到 A 发来的加密的邮件后，执行解密过程：与加密过程相反，首先对邮件内容进行 BASE64 解码，再利用自己的秘密密钥 SKB，通过 RSA 算法解出 IDEA 的密钥 K。再用此密钥恢复出 P1.Z，对 P1.Z 进行解压缩后还原出 P1。接着把明文 P 和邮件信息

摘要的密文 H 分离开来，并用 A 的公开密钥 PKA 解密 H 得到真正的邮件消息摘要。然后 B 自己也运用 MD5 算法对邮件明文 P 进行运算，生成一个 128 位的消息摘要。比较这两个摘要是否一致，如果一致，则表明 P 是 A 发来的邮件。

通过上述通信过程可以看出，PGP 既可以保证邮件不被第三方窃取，又可以防止发信人抵赖和信件被途中篡改。

由于 RSA 算法的计算量太大、速度太慢，对邮件正文这种大量数据不适合用它来加密。所以 PGP 实际上用来加密邮件正文的不是 RSA 本身，而是采用的 IDEA 加密算法。 IDEA 的加密和解密使用同一个密钥，它的主要缺点就是在公共网络环境中很难进行安全的密钥的传递，不适合 Internet 上邮件加密的需要。但 IDEA 的加密、解密速度比 RSA 快得多，所以 PGP 使用一个随机生成的密钥（每次加密都不同）运用 IDEA 算法对明文加密，然后用 RSA 算法对 IDEA 密钥加密。这样收件人同样使用 RSA 算法解密出这个随机的 IDEA 密钥，再用 IDEA 算法解密邮件本身。这样的链式加密就做到了既具有 RSA 算法的保密效果，又具有 IDEA 算法的快捷方便。这里，PGP 在每次加密邮件时所使用的 IDEA 密钥是一个随机数，而且为了增强随机性，PGP 是从用户敲击键盘的时间间隔上取得随机数种子来产生密钥的，从而更加增强了它的加密效果。

PGP 中使用 PKZIP 算法来压缩加密前的明文。这对电子邮件而言，一方面压缩后再加密得到的密文有可能比明文更短，这就节省了网络传输的时间；另一方面，明文经过压缩，实际上相当于多经过一次变换，信息更加杂乱无章，对非法攻击的抵御能力更强。

PGP 还可以只签名而不加密，这可以用于公开发表声明。声明人为了证实自己的身份，可以用自己的私钥签名。这样就可以让公众用其公开的公钥来验证该签名，从而确认声明人的身份。

2．PGP 的密钥管理机制

在 PGP 加密通信过程中，密钥无疑起着最为关键的作用。一个成熟的加密体系必然要有一个成熟的密钥管理机制与之相配套。PGP 对于密钥管理也提出了一套分配、使用、管理的方案。

公钥加密体制本身就是为了解决对称密钥加密体制中的密钥分配难以保密的问题而提出的。例如，攻击者常用的手段之一就是“监听”，如果密钥是通过网络传送就很容易被拦截。PGP 中采用公钥来加密，而公钥本来就要公开，所以不存在被监听的问题。但是公钥在发布过程中仍然存在安全隐患。例如，公钥被非法篡改，这就是公钥密码体系中的一大安全隐患，因为这很难被普通用户发现。

举例来说，假如用户 A 要向用户 B 发一封加密的邮件，那么 A 必须拥有 B 的公钥。于是 A 从公共目录中查到了 B 的公钥，并用它加密了邮件然后发给了 B。这是一个正常的过程。

但是，在这个过程中可能出现攻击：A 和 B 都不知道，另一个用户 C 用他自己假冒 B 的名字生成的密钥当中的公钥替换了 B 的公钥！

那么 A 用来发信的公钥就不是 B 的而是 C 的公钥。然而一切看来都很正常，因为 A 拿到的公钥的用户名是 B。于是 C 就可以用他手中的私钥来解密 A 发给 B 的邮件，甚至他还可以用 B 真正的公钥来转发 A 发给 B 的信，这样 A 和 B 都不会发现什么异常，而他们的通信却全部泄漏了。甚至 C 如果想改动 A 发给 B 的邮件也毫无问题。

而且，C 还可以伪造 B 的签名给 A 或其他人发送信息，因为 A 和其他人手中的公钥是 C 伪造的，A 和其他人可以正常解密这份伪造的签名，因而以为真是来自 B 的信息。

要防止这种情况，必须防止任何人伪造其他人的公钥。例如，通信双方直接见面并交换密钥，就可以避免得到伪造的公钥。然而当双方相隔遥远或不方便直接见面时，就难以直接交换密钥。这种情况下，PGP 是通过一种公钥介绍机制来解决这个问题的。

继续上面的例子：如果 A 和 B 有一个共同的朋友 D，而 D 知道他手中 B 的公钥是正确的（这里假设 D 已经认证过 B 的公钥）。这样 D 就可以用他自己的私钥在 B 的公钥上签名，表示他担保这个公钥是 B 的真正的公钥，并把它发送给 A。然后 A 用 D 的公钥来验证 D 发给 A 的 B 的公钥，同样 D 也可以向 B 担保 A 的公钥。这样 D 就成为了 A 和 B 之间的公钥介绍人。

这样 B 或 D 就可以放心地把 D 签过名的 B 的公钥列示到公共目录中，供 A 读取，没有人能够伪造 B 的公钥而不被 A 发现。这就是 PGP 从不安全的 Internet 上传递公钥的安全手段。

不过，如何确认 D 的公钥的安全可靠性呢？对这种情况 PGP 建议通过一个大家普遍信任的人或权威部门担当认证机构角色。每个由权威认证机构签字的公钥都被认为是真实的，这样大家只要有一份认证机构的公钥就行了。由于认证机构广泛提供公钥服务，因而其公钥流传广泛，假冒其公钥是很困难的，所以认证其公钥也非常方便。

在 PGP 中使用密钥，要注意在使用任何一个公钥之前，一定要首先认证它。无论什么情况下，都不要直接信任一个从公共渠道得来的公钥。而要使用可信的人介绍的公钥，或者自己与对方亲自认证。

由于 PGP 能够实现数字签名、不可否认、防止篡改、防止破译等功能，所以自从 PGP

推出以来，就受到人们的普遍欢迎。目前，PGP 几乎成为最流行的公钥加密软件。随着人们通信的增加和安全意识的增强，PGP 将会得到更加广泛的应用。

16.5 计算机病毒与防治

计算机技术和网络技术的飞速发展，为人们的工作、学习、生活带来了极大的方便。计算机已经成为人们不可缺少的现代化工具。但是计算机病毒的出现带给人们不安和忧虑，同时向人们提出了挑战。

16.5.1 计算机病毒概述

计算机病毒（Computer Virus）的概念最早是由美国计算机病毒研究专家 F.Cohen 博士提出的。对于计算机病毒的定义，不同的国家、不同的专家从不同的角度给出的定义也不尽相同。根据《中华人民共和国计算机信息系统安全保护条例》第 28 条规定：“计算机病毒，是指编制或者在计算机程序中插入的破坏计算机功能或者毁坏数据，影响计算机使用，并能自我复制的一组计算机指令或者程序代码。”此定义在我国具有法律效力和权威性。

和生物病毒一样，计算机病毒的复制能力使得计算机病毒可以很快地蔓延，又常常难以根除。它们能把自身附在宿主系统或文件中，当系统被运行或文件从一个用户传送到另一个用户时，它们就随同系统运行或文件传输一起蔓延开来。

在病毒的生命周期中，病毒一般会经历潜伏阶段、传染阶段、触发阶段和发作阶段 4 个阶段。多数病毒是基于某种特定的方式进行工作的，因此也依赖于某个特定的操作系统或某个特定的硬件平台。因此，攻击者经常利用某个特定系统的细节和弱点来设计病毒程序。

1．计算机病毒的特征

计算机病毒多种多样，但是它们都具有共同的特征，即传染性、非授权性、潜伏性和破坏性。

计算机病毒的传染性是指病毒具有把自身复制到其他系统或文件等宿主中去的能力，这是病毒的基本特征。非授权性是指病毒程序的执行不需要得到用户的同意，对用户来说是未知的。潜伏性是病毒生存的必要条件，即病毒潜伏在系统中而不被人们所发觉。破坏性是指病毒在一定条件下可以自动触发，并对计算机实施破坏，是病毒的表现特征。病毒的非授权性、潜伏性使得病毒的行为是不可预见的，也增加了病毒检测的困难。病毒破坏性的触发条件越多，则传染性越强，但同时其潜伏性降低。一个病毒必须具备传染性，但不一定需要拥

有其他属性。

2．计算机病毒的分类

计算机病毒按不同的分类标准，有许多不同分类：

按照操作系统分，可分为攻击 DOS 系统的病毒、攻击 Windows 系统的病毒、攻击 Unix/Linux 系统的病毒、攻击 OS/2 系统的病毒、攻击 Macintosh 系统的病毒、攻击手机的病毒、其他操作系统上的病毒。

按照链接方式分，计算机病毒可分为源码型病毒、嵌入型病毒、Shell 病毒、宏病毒、脚本型病毒、操作系统型病毒。

按照破坏情况分，计算机病毒可分为良性病毒和恶性病毒。按传播媒介来分，计算机病毒可分为单机病毒和网络病毒。

3．计算机病毒的组成

病毒程序一般由传染模块、触发模块、破坏模块和主控模块组成，相应地完成病毒的传染、触发和破坏等任务。也有少数病毒不具备所有的模块。

（1）传染模块。传染模块是病毒进行扩散传播的部分，负责把计算机病毒从一个系统或文件传播到更多的系统或文件中去。每个病毒都有一个自我识别的标记，叫作传染标记或病毒签名。病毒程序传染系统或文件时，要把传染标记写入系统或文件中某个特定区域，例如，宿主程序、注册表、物理磁道等，作为该系统或文件已被传染的标记，以防止重复传染，增强病毒的潜伏效果。传染模块的主要功能有：寻找一个可传染的系统或文件；检查该系统或文件中是否有传染标记，判断该系统或文件是否已经被传染；如果没有传染标记，则进行传染操作，将病毒代码植入宿主系统或文件中，完成一次传染。

（2）触发模块。病毒触发模块主要检查预定触发条件是否满足，如果满足，则调用相应传染或破坏模块，进行传染和破坏动作。病毒的触发条件有多种形式，如日期、时间、键盘、发现特定程序、发现网络连接、发现系统漏洞、传染的次数、特定中断调用的次数等。依据触发条件的情况，可以控制病毒传染和破坏动作的频率，使病毒在隐蔽的状态下，进行传染和破坏动作。

（3）破坏模块。破坏模块负责实施病毒的破坏动作。这些破坏动作可能是破坏程序及数据、降低系统的性能、干扰系统的运行，还有些病毒甚至可以破坏计算机硬件。也有少数病毒的破坏模块并没有明显的恶意破坏行为，仅在被传染的系统设备上表现出特定的现象，该模块有时又称为表现模块。

（4）主控模块。主控模块在总体上控制病毒程序的运行。染毒程序运行时，首先运行

的是病毒的主控模块。

16.5.2 网络环境下的病毒发展新趋势

在互联网给人们的工作、生活带来方便的同时，也给大量新病毒的产生和发展带来了“方便”。在互联网高度发达的今天，计算机病毒的数量急剧增多，传播途径也更加多样，传染速度也更加快捷。除了以往通过相互复制文件、系统之间交叉传染等方式外，目前的计算机病毒更多地通过网页、电子邮件、局域网共享、系统漏洞等方式在网络上进行自动传播。有些流行病毒，常常借助于网络在两、三天内迅速传遍全国，传遍全世界。

例如，E-mail 病毒就是目前最为流行 Internet 病毒种类之一，它通过在电子邮件附件中添加危险的病毒执行程序，在邮件正文中添加诱惑性的文字，诱使收件人执行附件病毒程序，以达到其激活的目的。目前此类病毒多数针对微软的 Outlook 和 Outlook Express 程序，并以使用 Windows 的地址簿联系人发送病毒邮件形式传播。不但危害个人，而且可能导致 Internet 邮件服务器由于收发大量的病毒附加的邮件被大量占用网络资源，直至邮件服务器崩溃。

此外，由于 Internet 即时通信被广泛运用，借助于 MSN、QQ、OICQ 等传播病毒也成为近年来病毒流行的一个趋势，病毒很可能通过这些软件自动发送有害信息实现自动传播。

目前，通过局域网共享传播的病毒也有很多。在广泛使用的 Windows 系统中，管理共享、特殊端口等默认是打开的，有些版本的系统中，甚至具有可写权限。这样，病毒在搜索到局域网共享资源后，便可以直接传染目标计算机相应文件夹中的文件或者将病毒写入相应系统中，以便得到在目标计算机中执行的机会。也正是因为这个原因，很多病毒在传染一台计算机后会造成某个局域网普遍传染的情况，并且在不断开网络连接的情况下很难将病毒清除干净。

在当前的网络环境下，最常见的恶意程序是木马程序，也称后门程序。它们大多利用系统漏洞或者空闲的端口，通过在系统中安装相应的木马程序，并通过互联网使用专门的软件监视宿主计算机，以这样的方式获得宿主系统文件访问授权，以及收集宿主系统的信息，如用户个人资料、银行账号与密码、网游账号等。

由于网络传输速度很快，所以网络环境下的病毒传染速度也非常快，一度轰动的“冲击波”、“震荡波”等病毒都是在相应的系统漏洞被发现后几天时间里，就迅速在全世界大范围蔓延开来。可以说，今天的计算机病毒的品种和传播速度，超过了以往任何时候。

16.5.3 计算机病毒的检测与清除

本节简单介绍计算机病毒的检测方法，以及如何清除病毒。

1．特征码检测

所谓特征码查毒法，就是在获取病毒样本后，提取出其特征码，（例如，杨基病毒的特征码是 16 进制的“F4 7A 2C 00”，快乐时光病毒中的“Fun Time”字符串等），然后通过该特征码对目标文件或内存等进行扫描。如果发现这种特征码，就说明感染了这种病毒，然后针对性地清除病毒。

特征码技术是最早被采用，而且被许多反病毒软件一直沿用至今的病毒检测方法。特征码检测方法检测病毒，方法简单、准确、快速，可识别病毒的名称，误报警率低。

但是，特征码技术只能诊断已知的计算机病毒，其响应速度永远滞后于病毒，而且不能检查未知病毒和变形病毒，不能对付隐蔽性病毒。

随着计算机病毒的发展，不断出现的新的病毒，甚至有些病毒具有自动变形功能，例如，“卡死脖”病毒，采用传统病毒特征码搜索技术的杀毒软件常常难以应付这些变形病毒。为此，人们提出了广谱特征码过滤技术，该技术在一定程度上可以弥补以上缺陷。

2．校验和检测

先计算正常文件的内容和正常的系统扇区数据的校验和，将该校验和写入数据库中保存。检测时，检查文件现在内容的校验和与原来保存的校验和是否一致，从而可以发现文件或扇区是否被感染，这种方法称校验和检测。

校验和检测技术的优点是：方法简单、能发现未知病毒、被查文件的细微变化也能发现。但是，它不能识别病毒种类。而且，由于病毒感染并非是文件内容改变的唯一原因，文件内容的改变有可能是正常程序引起的，所以校验和检测技术受到种种限制，同时这种方法也会影响文件的运行速度。另外，校验和不能检测新的文件，如从网络传输来的文件、磁盘和光盘拷入的文件、备份文件和压缩文档中的文件等。

3．行为监测

随着近年来病毒与反病毒斗争的不断升级、新病毒产生的速度不断加快，传统反病毒技术滞后于病毒的特点越来越不能适应防病毒的需要，更需要采用通用反病毒技术来保护计算机的安全。现阶段中被广泛研究和采用的通用病毒检测技术有病毒行为监测技术、启发式扫描技术和虚拟机技术。

通过研究发现，病毒不论伪装得如何巧妙，它们总是存在着一些和正常程序不同的行为，

而这些行为在正常应用程序中却十分罕见，这就是病毒的行为特性。

常见的病毒行为特性有：对可执行文件进行写操作、写磁盘引导区、病毒程序与宿主程序的切换、程序自己重定位、通过搜索函数索引表来获取 API 函数地址等。

利用这些特征，就可以对病毒实施监视，在病毒程序体进行活动时发出报警。采用这种行为特性检测方法不仅可以检测出已知病毒，而且可以检测出新出现的未知病毒，无论该病毒是什么种类，或是否变形。但是，行为监测技术也可能误报警，而且不能识别病毒名称。

4．启发式扫描

在特征码扫描技术的基础上，利用对病毒代码的分析，获得一些统计的、静态的启发性知识，可以用于静态的启发性扫描技术（Heuristic Scanning）。

启发式扫描主要分析文件中的指令序列，根据统计知识，判断该文件可能被感染或者没有被感染，从而有可能找到未知的病毒。因此，启发式扫描技术是一种概率方法，遵循概率理论的规律。早期的启发式扫描软件采用代码反编译技术作为它的实现基础。这类病毒检测软件在内部保存数万种病毒行为代码的跳转表，每个表项对应一类病毒行为的必用代码序列，如病毒格式化磁盘必须用到的代码等。启发式病毒扫描软件利用代码反编译技术，反编译出被检测文件的代码，然后在这些表格的支持下，使用“静态代码分析法”和 “代码相似比较法”等有效手段，就能有效地查出已知病毒的变种，以及判定文件是否含有未知病毒。

由于病毒代码千变万化，具体实现启发式病毒扫描技术是相当复杂的。通常这类病毒检测软件要能够识别并探测许多可疑的程序代码指令序列，如格式化磁盘类操作、搜索和定位各种可执行程序的操作、实现驻留内存的操作、子程序调用中只执行入栈操作、远距离（如超过文件长度的三分之二）跳往文件头的指令等。一般来说，仅仅一项可疑的功能操作不足以触发病毒报警。但如果同时具有多项可疑操作，目标程序就很可能是病毒程序。

5．虚拟机

自动变形病毒，也称为多态性病毒或多型（形）性病毒。自动变形病毒每次感染宿主时都自动改变自身的程序代码和特征码，这类病毒的代表有“幽灵”病毒等。

一般而言，自动变形病毒采用以下几种操作来不断变换自己：采用等价代码对原有代码进行替换；改变与执行次序无关的指令的次序；增加许多垃圾指令；对原有病毒代码进行压缩或加密等。因为自动变形病毒对其代码不断进行变换，而且每次传染使用不同的密钥。将染毒文件的病毒代码相互比较，也难以找出相同的可作为病毒特征的稳定特征码，因此用传统检测方法根本无法检测出这类病毒。但是，自动变形病毒也有一个共同的规律：即无论病毒如何变化，每一个自动变形病毒在其自身执行时都要对自身进行还原。

为了检测自动变形病毒，出现了一种新的病毒检测方法——“虚拟机技术”。该技术用软件方法让病毒在一个虚拟的环境中，仿真一部分系统指令和功能调用，对病毒代码作解释执行，而且仿真运行不对系统产生实际的影响，即可获得程序运行的后果，并在此基础上对程序运行分析，进而判断是否存在病毒。不管病毒使用什么样的加密、隐形等伪装手段，只要在虚拟机所营造的虚拟环境下，病毒都会随着运行过程自动褪去伪装（实际上是被虚拟机动态还原）。正是基于上述设计原理，虚拟机在处理加密、变换、变形病毒方面具有很强的优越性。

虚拟机检测方法，实际上是用软件实现了模拟人工反编译、智能动态跟踪、分析代码运行的过程，其效率更高，也更准确。使得反病毒从单纯的静态分析进入了动态和静态分析相结合的新时期，极大地提高了对已知病毒和未知病毒的检测水平。在今后相当长的一段时间内，虚拟机技术还会有很大的发展。

6．病毒的清除

将病毒代码从宿主中去除，使之恢复为可正常运行的系统或程序，称为病毒清除。大多数情况下，采用反病毒软件或采用手工处理方式可以恢复受感染的文件或系统。

不是所有染毒文件都可以消毒，也不是所有染毒的宿主都能够被有效恢复。依据病毒的种类及其破坏行为的不同，感染病毒后，如果宿主数据没有被删除，常常可以恢复；如果宿主数据被病毒删除或覆盖、或者宿主数据的逻辑关系被病毒破坏，常常不能恢复。

16.5.4 计算机病毒的预防

“防重于治”，对于计算机病毒也是如此。在日常使用计算机的过程中，同时做好预防工作，可以很大程度上避免被病毒感染，减少不必要的物力、数据损失。

要预防计算机病毒，最好的方法就是不与外界交换文件，但这是不可能的。人们在工作中，要经常与外界进行各种数据交换。而大量与外界交换信息，就给病毒的感染与传播创造了条件。

为了保护计算机不受病毒破坏，至少必须做到：

（1）一定要在计算机中安装反病毒软件。

（2）不要轻易使用来历不明的或者没有经过确认的软件；对从网络上下载的程序和文档应十分小心，在执行文件或打开文档之前，要检查是否有病毒；从外部取得的介质及其中的文件，应检查病毒后再使用；压缩后的文件应解压缩后检查病毒。

（3）电子邮件的附件应该先检查病毒后再开启，并在发送邮件之前检查病毒；不要运行来历不明的 E-mail 附件，尤其是在邮件正文中以诱惑性的文字建议执行的附件程序。

（4）定期使用反病毒软件扫描系统。

（5）确保所使用的反病毒软件的扫描引擎和病毒代码库为最新的，因为旧的扫描引擎和病毒代码库不会检查到新出现的病毒。

（6）为防止引导型病毒对系统的破坏，应该在系统安装完成后立即制作系统应急启动盘，以便万一硬盘分区表遭到破坏时，能从应急盘启动，并用备份的引导区、分区表等直接进行恢复。

（7）对于一些重要的文件，要定期进行备份，以便万一系统遭受病毒破坏时能够从备份恢复。

（8）利用安全扫描工具定时扫描系统和主机。若发现漏洞，及时寻找解决方案，从而减少被病毒和蠕虫感染的机会。

（9）使用反病毒软件时，最好先查毒，找到了带毒文件后，再确定是否进行杀毒操作。因为查毒不是危险操作，它可能产生误报，但绝不会对系统造成任何损坏；而杀毒是危险操作，有的操作可能把文件破坏。

（10）建立本单位的计算机病毒防治管理制度；并对计算机用户进行反病毒培训。

16.6 身份认证与访问控制

访问控制是通过某种途径限制和允许对资源的访问能力及范围的一种方法。它是针对越权使用系统资源的保护措施，通过限制对文件等资源的访问，防止非法用户的侵入或者合法用户的不当操作造成的破坏，从而保证信息系统资源的合法使用。

访问控制技术可以通过对计算机系统的控制，自动、有效地防止对系统资源进行非法访问或者不当地使用，检测出一部分安全侵害，同时可以支持应用和数据的安全需求。

访问控制技术并不能取代身份认证，它是建立在身份认证的基础之上的。

访问控制技术包括如下几方面的内容：

（1）用户标识与认证。用户标识与认证是一种基于用户的访问控制技术，它是防止未经授权的用户进入系统的一种常规技术措施。用户标识用于向系统声明用户的身份。用户标识一般应当具有唯一性，其最常见的形式就是用户 ID。系统必须采用一定的策略来维护所有的用户标识。验证用户标识的有效性、真实性，通常有三种类型的认证方式：一是用户个

人掌握的秘密信息，例如，口令、密钥、PIN 码等；二是用户个人所拥有的带有认证信息的特定物品，例如，磁卡、IC 卡等；三是用户个人的特定生理、生物学特征，例如，声音、指纹等。在同一种系统中可以单独采用一种认证方法，也可以联合采用多种认证方法。

（2）逻辑访问控制。逻辑访问控制是基于系统的访问控制技术，用来控制特定的用户对特定资源的访问。通常，把用户分成不同的组，再对组授予不同的访问权限来实现对用户的逻辑访问控制，防止用户访问他所不需要访问的资源、或者进行与工作无关的访问。

（3）审计与跟踪。审计与跟踪系统的一个或多个运行记录，在事件发生后对事件进行调查，分析其时间、原因、活动内容、引发的相关事件、涉及的用户等。

（4）公共访问控制。如果一个应用系统是面向公众开放，允许公众进行访问时，面临的主要威胁是来自外部的匿名攻击，必须采取访问控制等措施以保护系统数据的完整性和敏感信息的保密性。

16.6.1 身份认证技术

身份认证是对系统的用户进行有效性、真实性验证。

1．口令认证方式

使用口令认证方式，用户必须具有一个唯一的系统标识，并且保证口令在系统的使用和存储过程中是安全的，同时口令在传输过程中不能被窃取、替换。另外特别要注意的是在认证前，用户必须确认认证者的真实身份，以防止把口令发给冒充的认证者。

使用口令的单向身份认证过程一般是：请求认证者和认证者之间建立安全连接、并确认认证者身份等；然后请求认证者向认证者发送认证请求，认证请求中必须包括请求认证者的 ID 和口令；认证者接受 ID 和口令，在用户数据库中找出请求认证的 ID 和口令；查找是否有此用户并比较两口令是否相同；最后向请求认证者发回认证结果。如果请求认证者的 ID 在认证者的用户数据库中，并且请求认证者发送的口令与数据库中相应的口令相同，则允许请求认证者通过认证。

2．基于公钥签名的认证方式

公开密钥签名算法的身份认证方式，是通过请求认证者与认证者（对于双向身份认证而言，双方互为请求认证者和认证者）之间对于一个随机数做数字签名与验证数字签名来实现的。这种方式中认证双方的个人秘密信息不用在网络上传送，从而减少了口令等秘密信息泄漏的风险。

采用数字签名技术认证与口令认证方式有一个很大的不同：口令认证通常在正式数据交换开始之前进行。认证一旦通过，双方即建立安全通道进行通信，此后的通信被认为是安全的，不再进行身份认证；而数字签名认证在每一次的请求和响应中进行，即接收信息的一方先从接收到的信息中验证发送者的身份信息，验证通过后才对收到的信息进行相应处理。

使用公钥加密算法进行身份认证要求：请求认证者必须具有私钥实现数字签名的功能；认证者必须具有使用公钥验证数字签名的功能；认证者必须具有产生随机数的功能，而且随机数的质量必须达到一定要求。

使用公钥加密算法进行身份认证的方式，对用于数字签名的私钥由参与通信的认证者自己保密，而用于验证数字签名的公钥则需要采用可靠的方式进行安全分发。一般可以采用公钥数据库方式或者使用认证机构签发数字证书的方式（认证机构与数字证书的内容参见前文 PKI 部分）。

如果使用公钥数据库的方式管理公钥，则请求认证者 ID 就包含在认证请求中发给认证者，认证者使用该 ID 从公钥数据库中获得请求认证者的公钥。

如果使用认证机构签发数字证书的方式管理公钥，则请求认证者的数字证书包含在认证请求中发给认证者，认证者验证请求认证者的数字证书后，从数字证书中获取请求认证者的公钥。

3．持卡认证方式

持卡认证方式最早采用磁卡。磁卡中最重要的部分是磁道，不仅存储数据，而且还存储用户的身份信息。目前所用的卡是 IC 卡，与磁卡相比，它除了存储容量大之外，还可一卡多用，同时具有可靠性高，寿命长，读写机构简单可靠，造价便宜，维护方便，容易推广等诸多优点。正由于上述优点，使得 IC 卡在世界各地广泛使用。IC 卡上一般分为不加密的公共区、加密的数据区等，有些还有自己的操作系统和微处理器。IC 卡已被广泛应用于身份认证领域。

一般 IC 卡与用户的个人 PIN 一起使用。在脱机系统中，PIN 以加密的形式存在卡中，识别设备读出 IC 卡中的身份信息，然后将其中的 PIN 解密，与用户输入的 PIN 比较，以决定 IC 卡持有者是否合法。在联机系统中，PIN 可不存在 IC 卡上，而存在主机系统中，鉴别时，系统将用户输入的 PIN 与主机的 PIN 比较，而由此认证其身份的合法性。

4．基于人体生物特征的认证方式

这种方式是指通过计算机，利用人体固有的生理特征或行为特征进行个人身份鉴定。与传统的身份鉴别手段相比，基于生物特征的认证技术具有突出的优点：一是不会遗忘或丢失；二是防伪性能好，无法伪造；三是随时随地可用。能够用来鉴别身份的生物特征一般具有广

泛性（每个人都应该具有这种特性）、唯一性（每个人拥有的特征应各不相同）、稳定性（所选择的特征应该不随时间变化而发生变化）和可采集性（所选择的特征应该便于采集、测量）。目前，可用于身份鉴别的生物特征主要有指纹、笔迹、脸像、红外温、视网膜、手形、掌纹等。

由于生物特征识别的设备比其他身份认证的设备要复杂，所以一般用在非常重要的机密场合，如军事等。生物特征识别主要采用模式识别技术。身份识别系统工作方式分为识别模式和鉴定模式，其性能指标主要有错误拒绝率和错误接受率等。在选择这种认证方式时需要对这些参数作认真的考虑。

5．动态口令技术（一次性口令技术）

一般情况下，所使用的计算机口令都是静态的，也就是说在一定的时间内是相对不变的，而且可重复使用。这种口令很容易被系统中的嗅探程序所劫持，而且很容易受到基于字典的暴力攻击。

针对这种静态口令认证方式的缺陷，人们提出了利用散列函数产生一次性口令的方法，即用户每次登录系统时使用的口令都是变化的。一次性口令是动态变化的密码，其变化来源于产生密码的运算因子。一次性口令的产生因子一般都采用双运算因子：一是用户的私钥，它代表用户身份的识别码，是固定不变的。二是变动因子，正是变动因子的不断变化，才能够产生动态的一次性口令。

动态口令技术认证方式中要用到动态口令密码卡，这是一种便于携带的智能化硬件产品。这种密码卡内置的构件和程序能通过密码卡内的密钥加上其他因子动态地计算出新的口令。

当密码卡持有者将这个口令输入计算机时，系统中的认证服务器会根据相同的算法和动态因子计算出对应于该密码卡的认证口令，并把这个口令与密码卡产生的口令比对，进行身份认证。

6．PPP 中的认证协议

点到点协议（Point-to-Point Protocol，PPP）提供了一种在点到点链路上封装网络层协议信息的标准方法。PPP 也定义了可扩展的链路控制协议。链路控制协议使用验证协议磋商机制，在链路层上传输网络层协议前验证链路的对端。

PPP 包含如下几个部分：在串行链路上封装数据报的方法；建立、配置和测试数据链路连接的链路控制协议（Link Control Protocol，LCP）；建立和配置不同网络层协议的一组网络控制协议（Network Control Protocol，NCP）。

PPP 协议定义了两种验证协议：密码验证协议（Password Authentication Protocol，PAP）和

挑战—握手验证协议（Challenge-Handshake Authentication Protocol，CHAP），此外还有扩展认证协议（Extensible Authentication Protocol，EAP）。

一个典型的 PPP 链路建立过程分为三个阶段：创建阶段、认证阶段和网络层协商阶段。

（1）创建阶段。在这个阶段，将对基本的通信方式进行选择。链路两端设备通过 LCP 向对方发送配置信息，建立链路。在链路创建阶段，只是对验证协议进行选择，具体的用户验证过程在认证阶段实现。

（2）认证阶段。在这个阶段，客户端会将自己的身份发送给远端的接入服务器。该阶段使用一种安全的验证方式避免第三方窃取数据或冒充远程客户接管与客户端的连接。认证成功，则转到网络层协商阶段。如果认证失败，则链路终止。

（3）网络层协商阶段。认证阶段完成之后，PPP 将调用在链路创建阶段选定的各种 NCP 协商高层协议问题，例如，在该阶段 IP 控制协议可以向拨入用户分配动态地址。这样，经过三个阶段以后，一条完整的 PPP 链路就建立起来了。

最常用的认证协议有 PAP 和 CHAP，此外还有 EAP。

（1）PAP。PAP 是一种简单的明文验证方式。网络接入服务器要求用户提供用户名和口令，PAP 以明文方式返回用户信息，并且对回送或者重复验证和错误攻击没有保护措施。很明显，这种验证方式的安全性较差，第三方可以很容易地获取被传送的用户名和口令，并利用这些信息与网络接入服务器建立连接获取网络接入服务器提供的资源。所以，一旦用户密码被第三方窃取，PAP 无法提供避免受到第三方攻击的保障措施。

（2）CHAP。CHAP 是一种加密的验证方式，能够避免建立连接时传送用户的明文密码。网络接入服务器向远程用户发送一个挑战口令，其中包括会话 ID 和一个任意生成的挑战字串。远程客户端使用 MD5 散列算法返回用户名和加密的挑战口令、会话 ID 及用户口令。

CHAP 对 PAP 进行了改进，不再直接通过链路发送明文口令，而是使用挑战口令以散列算法对口令进行加密。因为服务器端存有客户的明文口令，所以服务器可以重复客户端进行的散列操作，并将结果与用户返回的口令进行对照。

CHAP 为每一次验证任意生成一个挑战字串来防止受到攻击。在整个连接过程中，CHAP 将不定时地随机向客户端重复发送挑战口令，从而避免非法入侵者冒充远程客户进行攻击。

HAP 验证方式具有如下的优点：

① 通过可变的挑战口令和随机地、重复地发挑战口令，CHAP 防止了重放攻击。

② 该认证方法依赖于认证者和对端共享的密钥，密钥不是通过链路发送的。

③ 虽然该认证是单向的，但是在两个方向都进行 CHAP 协商，同一密钥可以很容易地

实现交互认证。

④ 由于 CHAP 可以用在许多不同的系统认证中，因此可以用用户名作为索引，以便在一张大型密钥表中查找正确的密钥。这样也可以在一个系统中支持多个用户名—密钥对，在会话中随时改变密钥。

CHAP 在设计上的要求：

① CHAP 算法要求密钥长度必须至少是 1 字节，至少应该不易让人猜出，密钥最好至少是散列算法所选用的散列码的长度，如此可以保证密钥不易受到穷举搜索攻击。所选用的散列算法，必须保证从已知挑战口令和响应值来确定密钥在计算上是不可行的。

② 每一个挑战口令应该是唯一的，否则在同一密钥下，重复挑战口令将使攻击者能够用以前截获的响应值应答挑战口令。由于希望同一密钥可以用于地理上分散的不同服务器的认证，因此挑战口令应该做到全局临时唯一。

③ 每一个挑战口令也应该是不可预计的，否则攻击者可以欺骗对方，让对方响应一个预计的挑战口令，然后用该响应冒充对端欺骗认证者。虽然 CHAP 不能防止实时地主动搭线窃听攻击，但是只要能产生不可预计的挑战口令就可以防范大多数的主动攻击。

（3）EAP。EAP 是一个用于 PPP 认证的通用协议，可以支持多种认证方法。EAP 并不在链路控制阶段而是在认证阶段指定认证方法，这样认证方就可以在得到更多的信息以后再决定使用什么认证方法。这种机制还允许 PPP 认证方简单地把收到的认证信息传给后方的认证服务器，由后方的认证服务器来真正实现各种认证方法。

EAP 的认证过程是：在链路阶段完成以后，认证方向对端发送一个或多个请求报文。在请求报文中有一个类型字用来指明认证方所请求的信息类型，例如，可以是对端的 ID、MD5 的挑战口令、一次性密码及通用密码卡等。MD5 的挑战口令对应于 CHAP 认证协议的挑战口令。典型情况下，认证方首先发送一个 ID 请求报文随后再发送其他的请求报文。对端对每一个请求报文响应一个应答报文。和请求报文一样，应答报文中也包含一个类型字段，对应于所回应的请求报文中的类型字段。认证方再通过发送一个成功或者失败的报文来结束认证过程。

EAP 具有突出的优点：它可以支持多种认证机制，而不需要在建立连接阶段指定；某些设备，例如，网络接入服务器，不需要关心每一个请求信息的真正含义，而是作为一个代理把认证报文直接传给后端的认证服务器，设备只需关心认证结果是成功还是失败，然后结束认证阶段。

当然 EAP 也有一些缺点：它需要在 LCP 中增加一个新的认证协议，这样现有的 PPP

要想使用 EAP 就必须进行修改。同时，使用 EAP 也和现有的在 LCP 协商阶段指定认证方法的模型不一致。

7．RADIUS 协议

RADIUS（Remote AuthenticationDial-in User Service）协议是由朗讯公司提出的客户/ 服务器方式的安全认证协议，它能在拨号网络中提供注册、验证功能，现已成为 Internet 的正式协议标准，是当前流行的 AAA（Authentication、Authorization、Accountion）协议。

RADIUS 协议可以把拨号和认证这两种功能放在两个分离的服务器——网络接入服务器（NAS）和后台认证服务器（RADIUS 服务器）上。在 RADIUS 服务器上存放有用户名和它们相应的认证信息的一个大数据库，来提供认证用户名和密码及向用户发送配置服务的详细信息等。

RADIUS 具有非常突出的特点：

① RADIUS 协议使用 UDP 进行传输，它使用 1812 号端口进行认证，以及认证通过后对用户授权，使用 1813 号端口对用户计费。

② 支持多种认证方法，RADIUS 能支持 PAP、CHAP、UNIX Login 及其他认证方法；

③ 支持认证转接（AuthenticationForwarding），一个 RADIUS 服务器可以作为另一个 RADIUS 服务器的客户端向它要求认证，这叫作认证转接。

④ 协议扩展性好，通过协议中变长的属性串能够进一步扩展 RADIUS 协议。

⑤ 认证信息都加密传输，安全性高。RADIUS 服务器和接入服务器之间传递的认证信息用一个事先设置的口令进行加密，防止敏感信息泄露，因此安全性高。

RADIUS 的认证过程如下：

① 接入服务器从用户那里获取用户名和口令（PAP 口令或 CHAP 口令），把它同用户的一些其他信息（如主叫号码、接入号码、占用的端口等）组成 RADIUS 认证请求数据包发送给 RADIUS 服务器，请求认证。

② RADIUS 服务器收到认证请求包后，首先查看接入服务器是否已经登记，然后根据请求中用户名、口令等信息验证用户是否合法。如果用户非法，则向接入服务器发送访问拒绝包；如果用户合法，那么 RADIUS 服务器会将用户的配置信息，例如，用户类型、IP 地址、连接协议、端口信息、ACL 授权等信息，一起组成访问接受包发送回接入服务器。

③ 接入服务器收到访问接受/拒绝包时，首先要判断包中的签名是否正确，如果不正确将认为收到了一个非法的包。验证签名的正确性后，如果收到了访问接受包，那么接入服务器会接受用户的上网请求，并用收到的授权信息对用户进行配置、授权，限制用户对资源的

访问；如果收到的是访问拒绝包则拒绝该用户的上网请求。

④ 当用户成功登录后，接入服务器会向 RADIUS 服务器发送一个连接开始的记账信息包，其中包括用户使用的连接种类、协议和其他自定义的用户记账的信息；当用户断开连接后，接入服务器再向 RADIUS 服务器发送一个连接结束的记账信息包，通知 RADIUS 服务器停止对该用户记账。RADIUS 服务器根据收到的记账信息包按照该用户的设置为用户记账。

16.6.2 访问控制技术

访问控制是在身份认证的基础上，根据不同身份的用户对用户的访问请求加以限制。身份认证关心的是“你是谁，你是否拥有你所声明的身份”这个问题；而访问控制则关心“你能做什么，不能做什么”的问题。

在访问控制过程中，一般把发出访问、存取请求的一方，例如，用户、程序、进程等叫作主体；而把被访问的对象和资源，例如，文件、数据库、设备、内存区域等叫作客体。另外还有一套定义主体与客体之间相互关系，确定不同主体对不同客体的访问能力与权限的规则，叫作访问规则。一个完整的访问控制体系就是由上述三方面共同构成的。

1．访问控制策略

访问控制策略一般可以划分为三类：自主访问控制（Discretionary Access Control， DAC），强制访问控制（Mandarory Access Control，MAC），基于角色的访问控制（Roal Based Access Control，RBAC）。其中 DAC、MAC 是属于传统的访问控制策略，而 RBAC 则是后来出现的一种访问控制策略，被认为具有很大的优势，具有很好的发展前景。

（1）DAC。自主访问控制是目前计算机系统中实现最多的访问控制机制，它使主体可以自主地进行配置以决定其他的主体可以采取什么样的方式来访问其所拥有的一些资源，即一个拥有一定权限范围的主体可以直接或者间接地把权限授予其他的主体。

常见的操作系统如 Windows、UNIX 等都是采用自主访问控制策略来实施访问控制的。其常见的方式是由某个用户（一般为某个文件或资源的拥有者或超级管理员）采用某种方式指定不同类型、不同分组的其他用户对其名下的资源的访问许可和访问方式。

自主访问控制策略中，由用户自己决定其他用户对系统中某些资源的访问权限，这样虽然方便，但是却很难保证这种类型的授权对于整个系统来说是安全的。首先，用户往往不知道或者难以确定其他的用户是否适合具有对某些资源的访问权限；其次，如果不是所有的用户都有很强的安全意识，可能随意授权，那么这对于系统安全就是一个潜在的威胁；再次，

由用户自己决定访问权限的分配，不利于系统管理员实施统一的全局访问控制；另外，许多组织中往往希望对于信息系统采取的授权与控制结构能够与该组织的行政结构一致。总之，自主访问控制策略容易使系统失控，容易给非法入侵者留下可乘之机。所以，自主访问控制策略的安全性不是很高。

随着网络规模的扩大，用户对访问控制服务的质量也提出了更高的要求，采用自主访问控制策略已经很难满足一个安全性要求比较高的系统的需要。

（2）MAC。强制访问控制是系统统一采用某种访问权限的授予和撤销的策略，而且强制所有主体都必须服从这种访问权限的分配。

MAC 一般用在安全级别层次比较多的军事、安全等特殊应用领域中。它预先为系统中接受的所有主体、客体根据可以信任的程度、所处的岗位和承担的任务、信息的敏感程度、时间发展的阶段等划分成若干级别，例如，信息可以分为绝密、机密、秘密和无密级等不同的级别。然后再根据主体和客体的级别标记来决定访问模式，任何用户对任何客体的访问请求都由这种安全级别的划分及相应的权限配置来控制。

强制访问控制由于过于强调系统的安全性能，虽然能够很好地控制系统的安全，但是它管理起来比较麻烦，工作量很大，也不够灵活。

（3）RBAC。DAC 和 MAC 访问控制策略都各有其特点，但是也各有它们的不足。而基于角色的访问控制则可以在克服以上两者的缺点的同时，提供一个良好的安全的系统环境，因而是面向企业的系统中一种十分有效的访问控制策略。

DAC 系统中，有一种常见的情况，就是在一个组织中，最终用户能够使用某些资源，但是它并不是该资源的拥有者，资源的拥有者是这个组织或组织中的所有用户。这时，就应该基于用户的职务来进行访问权限的设置和分配，而不应该基于资源的拥有者来进行。

例如，在图书馆中，应该根据某一个用户是流通人员、文献编目人员，还是分馆的管理员等不同的角色来分配和设置权限。如果是文献编目人员，那么他对系统中流通的图书这种资源就只能有查看的权限，而对未进行典藏的图书等资源就有比较高的访问权限；如果是分馆的管理员，那么他相应地就具有对该分馆的读者、文献等资源有较高的访问权限，而对其他用户则没有。也就是说，用户具有什么样的访问权限，不直接取决于用户自己，而是取决于他所属的角色，有什么样的角色就有什么样的权限。

角色的种类和访问权限由系统管理员来定义，每一个成员属于哪种类型的角色也由系统管理员来规定，即只有系统管理员才有权定义和分配角色，而且对于用户来说只能服从系统中的这一系列规定，而不能有自主的配置，因此这是一种非自主型访问控制策略。

2．访问许可的授权对访问许可的授权有三种类型：

（1）等级型。把对客体的存取控制权限的修改能力划分成不同的等级，拥有高级别修改能力的主体可以把这种权限分配给比其级别低的主体。依此类推，从而将访问许可的授权关系组成一个树型结构。

例如，超级管理员可以作为这个等级树的根，具有修改所有客体的存取控制表的能力，且可以向任意一个主体分配这种修改权。系统管理员把用户根据部门划分成多个子集，并对部门领导授予相应存取控制权限的修改权和对修改权的分配权。部门领导又可以把自己所拥有的权力按照同样的方法向下授权。

这种方式的优点是树型结构与实际组织机构类似，并且可以由领导根据日常实际工作需要进行授权来对各级用户进行控制与管理。但这种方式也有一个缺点，就是对同一个客体来说，可能存在多个主体有能力修改其存取控制权限。

（2）拥有型。这种类型对每一个客体都有一个拥有者（一般情况下就是该客体的创建者），拥有者具有对所拥有的客体的全部的控制权，并且可以任意修改其拥有的客体的访问控制表，并可对其他主体授予或撤销对其客体的任何一种访问权限。但是拥有者无权将其对客体的访问控制权的分配权授予其他主体。

在 UNIX 系统中就是用这种方式来进行授权控制的。

（3）自由型。自由型的特点是一个客体的拥有者可以对任何主体授予对他所拥有的客体的访问权限，同时还可以把这种分配权授予其他主体而不受任何限制。这样，获得了这种授权的主体就可以把这种分配权授予更多的主体而不受该客体拥有者的限制。这样，一旦访问控制的分配权被授予出去，就很难控制对客体的访问了。显然，这样做安全性比较差，一般的系统中很少采用这种方式。

对于系统中如何更好地进行访问控制，可以参考本书下册中的相关讲解。

16.7 网络安全体系

ISO 的 OSI/RM 是著名的网络架构模型，但是，OSI/RM 并没有在安全性方面作专门的设计，因此该模型本身的安全性是很弱的。为了改善网络的安全状况，提高网络安全强度，ISO 又在 OSI/RM 的基础上提出了一套 OSI 安全架构，用以强化网络的安全性。

16.7.1 OSI 安全架构

OSI 安全架构是一个面向对象的、多层次的结构，它认为安全的网络应用是由安全的服务实现的，而安全服务又是由安全机制来实现的。

1．OSI 安全服务

针对网络系统的技术和环境，OSI 安全架构中对网络安全提出了 5 类安全服务，即对象认证服务、访问控制服务、数据保密性服务、数据完整性服务、禁止否认服务。

（1）对象认证服务。对象认证服务又可分为对等实体认证和信源认证，用于识别对等实体或信源的身份，并对身份的真实性、有效性进行证实。其中，对等实体认证用来验证在某一通信过程中的一对关联实体中双方的声称是一致的，确认对等实体中没有假冒的身份。信源认证可以验证所接收到的信息是否确实具有它所声称的来源。

（2）访问控制服务。访问控制服务防止越权使用通信网络中的资源。访问控制服务可以分为自主访问控制、强制访问控制、基于角色的访问控制。由于 DAC、MAC 固有的弱点，以及 RBAC 的突出优势，所以 RBAC 一出现就成为在设计中最受欢迎的一种访问控制方法。访问控制的具体内容前面已有讲述，此处不再赘述。

（3）数据保密性服务。数据保密性服务是针对信息泄漏而采取的防御措施，包括信息保密、选择段保密、业务流保密等内容。数据保密性服务是通过对网络中传输的数据进行加密来实现的。

（4）数据完整性服务。数据完整性服务包括防止非法篡改信息，如修改、删除、插入、复制等。

（5）禁止否认服务。禁止否认服务可以防止信息的发送者在事后否认自己曾经进行过的操作，即通过证实所有发生过的操作防止抵赖。具体的可以分为防止发送抵赖、防止递交抵赖和进行公证等几个方面。

2．OSI 安全机制

为了实现前面所述的 OSI 5 种安全服务，OSI 安全架构建议采用如下 8 种安全机制：加密机制、数字签名机制、访问控制机制、数据完整性机制、鉴别交换机制、流量填充机制、路由验证机制、公正机制。

（1）加密机制。加密机制即通过各种加密算法对网络中传输的信息进行加密，它是对信息进行保护的最常用措施。加密算法有许多种，大致分为对称密钥加密与公开密钥加密两大类，其中有些（例如，DES 等）加密算法已经可以通过硬件实现，具有很高的效率。

（2）数字签名机制。数字签名机制是采用私钥进行数字签名，同时采用公开密钥加密算法对数字签名进行验证的方法。用来帮助信息的接收者确认收到的信息是否是由它所声称的发送方发出的，并且还能检验信息是否被篡改、实现禁止否认等服务。

（3）访问控制机制。访问控制机制可根据系统中事先设计好的一系列访问规则判断主体对客体的访问是否合法，如果合法则继续进行访问操作，否则拒绝访问。访问控制机制是安全保护的最基本方法，是网络安全的前沿屏障。

（4）数据完整性机制。数据完整性机制包括数据单元的完整性和数据单元序列的完整性两个方面。它保证数据在传输、使用过程中始终是完整、正确的。数据完整性机制与数据加密机制密切相关。

（5）鉴别交换机制。鉴别交换机制以交换信息的方式来确认实体的身份，一般用于同级别的通信实体之间的认证。要实现鉴别交换常常用到如下技术。

① 口令：由发送方提交，由接收方检测。

② 加密：将交换的信息加密，使得只有合法用户才可以解读。

③ 实体的特征或所有权：例如，指纹识别、身份卡识别等。

（6）业务流填充机制。业务流填充机制是设法使加密装置在没有有效数据传输时，还按照一定的方式连续地向通信线路上发送伪随机序列，并且这里发出的伪随机序列也是经过加密处理的。这样，非法监听者就无法区分所监听到的信息中哪些是有效的，哪些是无效的，从而可以防止非法攻击者监听数据，分析流量、流向等，达到保护通信安全的目的。

（7）路由控制机制。在一个大型的网络里，从源节点到目的节点之间往往有多种路由，其中有一些是安全的，而另一些可能是不安全的。在这种源节点到目的节点之间传送敏感数据时，就需要选择特定的安全的路由，使之只在安全的路径中传送，从而保证数据通信的安全。

（8）公证机制。在一个复杂的信息系统中，一定有许多用户、资源等实体。由于各种原因，很难保证每个用户都是诚实的，每个资源都是可靠的，同时，也可能由于系统故障等原因造成信息延迟、丢失等。这些很可能会引起责任纠纷或争议。而公证机构是系统中通信的各方都信任的权威机构，通信的各方之间进行通信前，都与这个机构交换信息，从而借助于这个可以信赖的第三方保证通信是可信的，即使出现争议，也能通过公证机构进行仲裁。

3．OSI 安全服务与安全机制之间的关系

OSI 安全服务与安全机制之间不是一一对应的关系。有的服务需要借助多种机制来实线，同时，有些机制可以提供多种服务。一般来说，OSI 安全服务与安全机制之间具有如 表 16-1

所示的关系，在设计中可以参考选用这些安全机制从而提供相应的安全服务。

表 16-1　OSI 安全服务与安全机制之间的关系

安全服务 / 安全机制	对象认证	访问控制	数据保密性	数据完整性	防止否认
加密	√		√	√	
数字签名	√	√		√	√
访问控制		√			
数据完整性				√	√
鉴别交换	√				
业务流填充			√		
路由控制			√		
公证					√

16.7.2 VPN 在网络安全中的应用

虚拟专用网络（Virtual Private Network，VPN）是指利用不安全的公共网络如 Internet 等作为传输媒介，通过一系列的安全技术处理，实现类似专用网络的安全性能，保证重要信息的安全传输的一种网络技术。

1．VPN 技术的优点

VPN 技术具有非常突出的优点，主要包括：

（1）网络通信安全。VPN 采用安全隧道等技术提供安全的端到端的连接服务，位于 VPN 两端的用户在 Internet 上通信时，其所传输的信息都是经过 RSA 不对称加密算法加密处理的，它的密钥则是通过 Diffie-Hellman 算法计算得出的，可以充分地保证数据通信的安全。

（2）方便的扩充性。利用 VPN 技术实现企业内部专用网络，以及异地业务人员的远程接入等，具有方便灵活的可扩性。首先是重构非常方便，只需要调整配置等就可以重构网络；其次是扩充网络方便，只需要配置几个节点，不需要对已经建好的网络作工程上的调整。

（3）方便的管理。利用 VPN 组网，可以把大量的网络管理工作放到互联网络服务提供商一端来统一实现，从而减轻了企业内部网络管理的负担。同时 VPN 也提供信息传输、路由等方面的智能特性及与其他网络设备相独立的特性，也给用户提供了网络管理的灵活的手段。

（4）节约成本显著。利用已有的无处不在的 Internet 组建企业内部专用网络，可以节省大量的投资成本及后续的运营维护成本。以前，要实现两个远程网络的互联，主要是采用专线连接方式。这种方式成本太高。而 VPN 则是在 Internet 基础上建立的安全性较好的虚

拟专用网，因此成本比较低，而且可以把一部分运行维护工作放到服务商端，又可以节约一部分维护成本。

2．VPN 的原理

实现 VPN 需要用到一系列关键的安全技术，包括：

（1）安全隧道技术。即把传输的信息经过加密和协议封装处理后再嵌套装入另一种协议的数据包中送入网络中，像普通数据包一样进行传输。经过这样的处理，只有源端和目标端的用户对加密封装的信息能进行提取和处理，而对于其他用户而言，这些信息只是无意义的垃圾。

（2）用户认证技术。在连接开始之前先确认用户的身份，然后系统根据用户的身份进行相应的授权和资源访问控制。

（3）访问控制技术。由 VPN 服务的提供者与最终网络信息资源的提供者共同协商确定用户对资源的访问权限，以此实现基于用户的访问控制，实现对信息资源的保护。

VPN 系统的结构如图 16-12 所示。

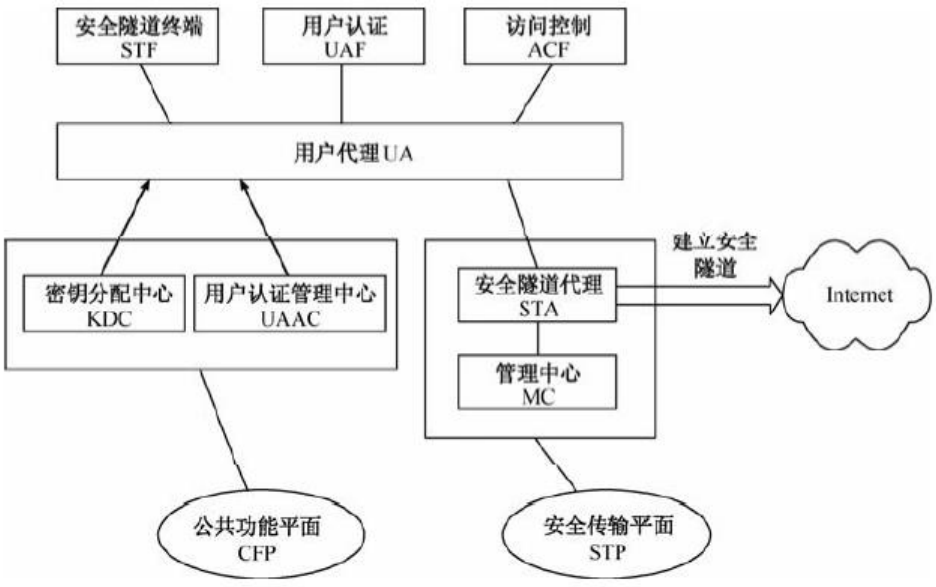


图 16-12　VPN 系统的结构

在图 16-12 中，安全隧道代理和管理中心组成安全传输平面（Secure Transmission Plane，STP），实现在 Internet 上安全传输和相应的系统管理功能。用户认证管理中心和密钥分配中心组成公共功能平面（Common Function Plane，CFP），它是安全传输平面的辅助平面，主要向用户代理提供相对独立的用户身份认证与管理、密钥的分配与管理功能。

建立 VPN 通信时，VPN 用户代理向安全隧道代理请求建立安全隧道，安全隧道代理接

受后，在管理中心的控制和管理下在 Internet 上建立安全隧道，然后向用户提供透明的网络传输。VPN 用户代理包括安全隧道终端功能、用户认证功能和访问控制功能三个部分，它们共同向上层应用提供完整的 VPN 服务。

（1）安全传输平面。安全传输平面实现在 Internet 上安全传输和相应的系统管理功能，这是由安全隧道代理和管理中心共同完成的。

① 安全隧道代理。安全隧道代理可以在管理中心的控制下将多段点到点的安全通路连接成一条端到端的安全隧道。它是 VPN 的主体，其主要作用有：

建立与释放安全隧道。按照用户代理的请求，在用户代理与安全隧道代理之间建立点到点的安全通道，并在这个安全通道中进行用户身份验证和服务等级协商等交互。在安全通道中进行初始化过程，可以充分保护用户身份验证等重要信息的安全。然后在管理中心的控制下建立发送端到接收端之间由若干点到点的安全通道依次连接而成的端到端的安全隧道。在信息传输结束之后，由通信双方中的任一方代理提出释放隧道连接请求，就可以中断安全隧道连接。

用户身份的验证。在建立安全隧道的初始化过程中，安全隧道代理要求用户代理提交用户认证管理中心提供的证书，通过验证该证书可以确认用户代理的身份。必要时还可以由用户代理对安全隧道代理进行反向认证以进一步提高系统的安全性。

服务等级的协商。用户身份验证通过之后，安全隧道代理与用户代理进行服务等级的协商，根据其要求与 VPN 系统当时的实际情况确定提供的服务等级并报告至管理中心。

信息的透明传输。安全隧道建立之后，安全隧道代理负责通信双方之间信息的传输，并根据商定的服务参数进行相应的控制，对其上的应用提供透明的 VPN 传输服务。

控制与管理安全隧道。在维持安全隧道连接期间，安全隧道代理还要按照管理中心的管理命令对已经建立好的安全隧道进行网络性能及服务等级等有关的管理与调整。

② VPN 管理中心。VPN 管理中心是整个 VPN 的核心部分，它与安全隧道代理直接联系，负责协调安全传输平面上的各安全隧道代理之间的工作。具体功能包括：

安全隧道的管理与控制。确定最佳路由，并向该路由上包含的所有安全隧道代理发出命令，建立安全隧道连接。隧道建立以后，管理中心继续监视各隧道连接的工作状态，对出错的安全隧道，管理中心负责重新选择路由并将该连接更换到新的路由。在通信过程中，还可以根据需要向相应安全隧道上的代理发送管理命令，以优化网络性能、调整服务等级等。

网络性能的监视与管理。管理中心不断监视各安全隧道代理的工作状态，收集各种 VPN 性能参数，并根据收集到的数据完成 VPN 性能优化、故障排除等功能。同时，管理中心还负

责完成对各种 VPN 事件进行日志记录、用户计费、追踪审计、故障报告等常用的网络管理功能。

（2）公共功能平面。公共功能平面是安全传输平面的辅助平面，向 VPN 用户代理提供相对独立的用户身份认证与管理、密钥的分配与管理功能，分别由用户认证管理中心和 VPN 密钥分配中心完成。

① 认证管理中心。认证管理中心提供用户身份认证和用户管理。用户认证就是以第三者身份客观地向 VPN 用户代理和安全隧道代理中的一方或双方提供用户身份的认证，以便他们能够相互确认对方的身份。

用户管理是指与用户身份认证功能直接相关的用户管理部分，即对各用户（包括用户代理、安全隧道代理及认证管理中心等）的信用程度和认证情况进行日志记录，并可在 VPN 与建立安全隧道双方进行服务等级的协商时参考。这里的管理是面向服务的，而与用户权限、访问控制等方面有关的用户管理功能则不在此列。

② 密钥分配中心。密钥分配中心向需要进行身份验证和信息加密的双方提供密钥的分配、回收与管理功能。在 VPN 系统里，用户代理、安全隧道代理、认证管理中心等都是密钥分配中心的用户。

希赛教育专家提示：采用 VPN 技术，既能保证整个企业网络的连通性与数据的共享，又能保证财务等重要数据的安全，是一种实现企业内部本地网络互连的良好方案。

16.8 系统的安全性设计

要设计一个安全的系统，除了要了解一些前面讲到的常用的保护手段和技术措施外，还要对系统中可能出现的安全问题或存在的安全隐患有充分的认识，这样才能对系统的安全作有针对性的设计和强化，即“知己知彼，百战百胜”。

下面以物理安全、防火墙、入侵检测为例讲解系统安全中可能出现的问题及如何采取相应的措施。

16.8.1 物理安全问题与设计

物理安全包括物理设备本身是否安全可靠，还包括设备的位置与环境的安全、限制物理访问、地域因素等几个方面。

信息系统的所有重要的物理设备、设施都应该放在专门的区域，并尽可能集中，同时严

格限制外来人员来访，尽可能地减少未经授权的访问。

物理安全还要求在设计中注意物理设备的冗余备份，例如，核心设备或部件都应该是热备份系统，具有实时或准实时切换的能力。

物理安全还要求严格限制对网络信息点、线缆等网络基础设施及其所在地进行物理访问，要想访问必须经过专门的授权。

物理安全还包括环境方面的因素，在设计之初就要对信息系统中的温度、湿度、灰尘、振动、雷电、电力等方面的参数有明确的要求，要对自然灾害（地震、台风、闪电等）有充分的考虑，还要对电磁泄漏等方面的要求作明确的定义。设计系统时要对这些因素全盘考虑，并采取适当的防护措施或强化手段。例如，机房这种重要的地点，除了所在的建筑物要有防雷系统外，还可以加装一套专用的防雷系统。这样可以保证即使建筑物遭到雷击时，万一建筑物的避雷系统未能充分保护好昂贵的信息系统，那么单独为机房安装的专用避雷系统也能保障机房设备免受损失。

16.8.2 防火墙及其在系统安全中的应用

网络安全隐患主要是由网络的开放性、无边界性、自由性造成的，所以保护网络安全可以首先考虑把被保护的网络从开放的、无边界的、自由的公共网络环境中独立出来，使之成为有管理的、可控制的、安全的内部网络。也只有做到这一点，实现信息网络的通信安全才有可能。

目前，最基本的网络分隔手段就是防火墙，它也是目前用来实现网络安全的一种主要措施。利用防火墙，可以用来在拒绝未经允许的网络连接、阻止敏感数据的泄漏的同时，保证合法用户的合法网络流量畅通无阻，可以实现内部可信任网络（如企业网）与外部不可信任网络（如 Internet）之间，或是内部不同子网之间的隔离与控制，保证网络系统及网络服务的可用性。

1．防火墙的基本原理防火墙通常使用的采用包过滤、状态检测、应用网关等几种方式控制网络连接。

包过滤防火墙是一种简单而有效的安全控制技术，它根据在防火墙中预先定义的规则（允许或禁止与哪些源地址、目的地址、端口号有关的网络连接），对网络层和传输层的数据包进行检查，进而控制数据包的进出。包过滤的优点是对用户透明、传输性能高。但是由于只能在网络层、传输层进行控制，安全控制的方式也只限于源地址、目的地址和端口号这

几种，对于应用层的信息无法感知，因而只能进行较为初步的安全控制，对于拥塞攻击、内存覆盖攻击或病毒等高层次的攻击手段，则无能为力。

状态检测防火墙保持了包过滤防火墙的优点，所以性能比较好，而且对应用是透明的。同时，状态检测防火墙改进了包过滤防火墙仅仅检查进出网络的数据包，不关心数据包状态的缺点，在防火墙的内部建立状态连接表，维护了连接，将进出网络的数据当成一个个的事件来处理。对于每一个网络连接，状态检测根据预先设置的安全规则，允许符合规则的连接通过，并在内存中记录下该连接的相关信息，生成状态表。对该连接的后续数据包，只要符合状态表，就可以通过。这种方式的好处在于：由于不需要对每个数据包进行规则检查，而是对一个连接的后续数据包（通常是大量的数据包）通过散列算法，直接进行状态检查，从而使得性能得到了较大提高。

与不关心应用层数的前两种方式不同，应用网关防火墙检查所有应用层的信息包，并将检查的内容信息放入决策过程，从而提高网络的安全性。然而，应用网关防火墙是通过打破客户机/服务器模式实现的。每个客户机/服务器通信需要两个连接：一个是从客户端到防火墙，另一个是从防火墙到服务器。另外，每个网关需要一个不同的应用进程，或一个后台运行的服务程序，对每个新的应用必须添加针对此应用的服务程序，否则不能使用该服务。所以，应用网关防火墙使用起来比较麻烦，而且通用性比较差。

2．防火墙的优点在系统中使用防火墙，对于系统的安全有很多的优点：

（1）可以隔离网络，限制安全问题的扩散。防火墙可以隔离不同的网络，或者用来隔离网络中的某一个网段，这样就能够有效地控制这个网段或网络中的问题在不同的网络中传播，从而限制安全问题的扩散。

（2）通过防火墙可以对网络中的安全进行集中化管理，简化网络安全管理的复杂度。只要在防火墙上配置好过滤策略，就能使防火墙成为一个网络安全的检查站，所有进出网络的信息都需要通过防火墙，把非法访问拒于门外。从而实现安全的集中统一的管理，并且能够简化安全管理的复杂度。

（3）能够有效地记录 Internet 上的活动。因为所有进出内部网络的信息都必须通过防火墙，所以防火墙能够收集内部网络和外部网络之间或者不同网段之间所发生的事件，为管理员的进一步分析与安全管理提供依据。

3．正确使用防火墙虽然防火墙的技术日渐成熟起来，成为维护网络安全的一个重要的手段。但是，它也不能完全解决网络上的安全问题。在实际使用过程中还有一些安全性是防火墙不能实现的，在实际工作中，一般应注意如下几点：

（1）防火墙虽然能对来自外部网络的非法连接作很严格的限制，但是对来自本地网络内部的攻击却无从防范。事实上，大多数攻击不是来自外部，而是来自内部。因此，即使使用了防火墙，对本地网络内部的主机、应用系统、数据库等也要采取其他有效的措施，才能真正做到安全。

（2）即使对于来自外部的攻击，目前的任何防火墙也不能做到完全阻挡所有的非法入侵。随着各种新技术的陆续涌现，非法分子对系统的深入研究与剖析，各种新的应用需求不断被开发，防火墙本身也会受到越来越多的威胁。对这些新的动态、趋势要密切关注，不断地升级防火墙、修正完善防火墙的配置，才能使防火墙本身更加坚固，进而长久地发挥安全保护作用。

（3）防火墙不能防范病毒，无法抵御基于数据的攻击。尽管防火墙的过滤技术在不断完善，可是由于病毒的类型太多，隐藏方式也非常复杂，而且它们很多都是隐藏在数据文件中，因此要防火墙对所有的包含病毒的文件作出限制是不太现实的，而应当在系统中单独安装专门的病毒网关或者在主机上安装相应的防病毒软件、反间谍软件等工具软件，才能较好地防范此类安全隐患。

（4）防火墙不能防范全部的威胁，而只能防备已知的威胁。所以在使用过程中，应当经常根据需要配合使用入侵检测系统。

（5）防火墙不能防范不通过它的链接。防火墙可以有效地过滤经过它的信息传输，但不能防范不通过它的信息传输，例如，如果允许拨号访问防火墙后面的内部系统，则防火墙没有任何办法对它进行控制。

16.8.3 入侵检测系统

传统上，一般采用防火墙作为系统安全的边界防线。但是，随着攻击者的知识日趋丰富，攻击工具与手法的日趋复杂多样，单纯的防火墙已经无法满足对安全高度敏感的部门的需要，网络的防卫必须采用一种纵深的、多样的手段。

与此同时，当今的网络环境也变得越来越复杂，各式各样的复杂的设备，需要不断升级、补漏，系统管理员的工作不断加重，不经意的疏忽便有可能造成安全的重大隐患。所以，信息系统中存在着不少可以被攻击者所利用的安全弱点、漏洞及不安全的配置，主要表现在操作系统、网络服务、TCP/IP 协议、应用程序（如数据库、浏览器等）、网络设备等几个方面。正是这些弱点、漏洞和不安全设置给攻击者以可乘之机。

另外，由于大部分网络缺少预警防护机制，即使攻击者已经侵入到内部网络，侵入到关键的主机，并从事非法的操作，系统管理员也很难察觉到。这样，攻击者就有足够的时间来做他们想做的任何事情。

要防止和避免遭受攻击和入侵，不仅要找出网络中存在的安全弱点、漏洞和不安全的配置，然后采取相应措施解决这些弱点、漏洞，对不安全的配置进行修正，最大限度地避免遭受攻击和入侵；还要对网络活动进行实时监测，一旦监测到攻击行为或违规操作，能够及时作出反应，包括记录日志、报警甚至阻断非法连接。

在这种环境下，入侵检测（Intrusion Detection）技术受到人们愈来愈多的关注，而且已经开始在各种不同的环境中发挥其关键作用。入侵检测系统可以在系统中发生一些不正常的操作时发出警报，防患于未然。设置硬件防火墙，可以提高网络的通过能力并阻挡一般性的攻击行为；而采用入侵检测系统，则可以对越过防火墙的攻击行为及来自网络内部的违规操作进行监测和响应。

入侵检测技术，通过对计算机网络或计算机系统中的若干关键点收集信息并对其进行分析，从中发现网络或系统中是否有违反安全策略的行为和被攻击的迹象。与其他安全产品不同的是，入侵检测系统需要更多的智能，它要根据智能库对收集到的数据进行分析，并采取相应措施。

作为对防火墙极其有益的补充，入侵检测系统（IDS）能够帮助人们快速发现系统攻击的发生，扩展了系统管理员的安全管理能力（包括安全审计、监视、进攻识别和响应等），提高了信息系统的安全性。入侵检测系统被认为是防火墙之后的第二道安全闸门，它能在不影响网络性能的情况下对网络进行监听，从而提供对内部攻击、外部攻击和误操作的实时保护。

入侵检测系统作为一种积极主动的安全防护工具，能够在计算机网络和系统受到危害之前进行报警、拦截和响应。其主要功能包括：通过检测和记录系统中的安全违规行为，惩罚信息系统攻击，防止入侵事件的发生；检测其他安全措施未能阻止的攻击或安全违规行为；检测黑客在攻击前的探测行为，预先给管理员发出警报；报告信息系统中存在的安全威胁；提供有关攻击的信息，帮助管理员诊断系统中存在的安全弱点，利于其进行修补。

在大型、复杂的计算机系统中布置入侵检测系统，可以明显提高信息系统安全管理的质量。

1．入侵检测技术入侵检测系统的处理过程分为数据采集阶段、数据处理及过滤阶段、入侵分析及检测阶段、报告及响应阶段 4 个阶段。数据采集阶段主要收集目标系统中引擎

提供的通信数据包和系统使用等情况。数据处理及过滤阶段是把采集到的数据转换为可以识别是否发生入侵的数据的阶段。分析及检测阶段通过分析上一阶段提供的数据来判断是否发生入侵。这一阶段是整个入侵检测系统的核心阶段。报告及响应阶段针对上一个阶段中得出的判断作出响应。如果被判断为发生入侵，系统将对其采取相应的响应措施，或者通知管理人员发生入侵，以便于采取措施。

在入侵检测系统的工作过程中，对信息系统中的各种事件进行分析，从中检测出违反安全策略的行为是入侵检测系统的核心功能。检测技术分为两类：一种是基于标识（signature-based）的入侵检测，另一种是基于异常情况（anomaly-based）的入侵检测。

基于标识的检测技术，先定义出违背安全策略的事件的特征，如网络数据包的某些头信息等。然后对收集到的数据进行分析，通过判别这类特征是否在所收集到的数据中出现来判断是否受到入侵。此方法非常类似杀毒软件的特征码检测，比较简单有效。

而基于异常的检测技术则先定义一组系统“正常”情况的数值，如 CPU 利用率、网络流量规律、文件校验和等（这类数据可以人为定义，也可以通过观察系统，并用统计的办法得出），然后将系统运行时的数值与所定义的“正常”情况比较，得出是否有被攻击的迹象。这种检测方式的核心在于如何定义所谓的“正常”情况。

两种检测技术的方法、所得出的结论有时会有非常大的差异。基于标识的检测技术的核心是维护一个知识库。对于已知的攻击，它可以详细、准确地报告出攻击类型，但是对未知攻击却效果有限，而且知识库必须不断更新。基于异常的检测技术则无法准确判别出攻击的手法，但它可以判别更广泛，甚至未发觉的攻击。如果条件允许，两者结合的检测会达到更好的效果。

2．入侵检测系统的种类和选用一般来说，入侵检测系统可分为主机型和网络型。

主机型入侵检测系统往往以系统日志、应用程序日志等作为数据源，当然也可以通过其他手段（如监控系统调用）从所在的主机收集信息进行分析。主机型入侵检测系统保护的一般是其所在的主机系统。主机型入侵检测系统需要为不同平台开发不同的程序，而且会增加系统负荷，还要在每一台主机安装，比较麻烦，但是可以充分利用操作系统本身提供的功能，并结合异常分析，更准确地报告攻击行为。

网络型入侵检测系统则以网络上的数据包作为数据源，通过在一台主机或网络设备上监听本网段内的所有数据包来进行分析判断。一般网络型入侵检测系统担负着保护整个网段的任务。这种系统应用十分简便：一个网段上只需安装一个或几个这样的系统，便可以监测整个网段的情况，但是它不跨越多个物理网段，对于复杂结构的网络（如交换环境）监测效果

有一定影响。

主机型入侵检测系统和网络型入侵检测系统各有利弊，应用中可以根据实际需要从中选择。

16.9 安全性规章

16.9.1 安全管理制度

信息系统安全，不仅要从技术角度采取若干措施，还要从组织管理的角度出发，制定明确的安全管理的规章制度，以确保安全技术实施的有效性。只有依靠安全管理规章的有力支持和保障，信息安全的技术解决方案才能够切实地取得预期的效果。

事实上，管理的缺失是信息安全失败的非常重要的原因。有统计表明，危害信息系统安全的因素中，70%以上来自组织内部。系统管理员随意性的配置或者软件升级不及时造成的安全漏洞，使用脆弱的用户口令，随意下载使用来自网络的软件，在防火墙内部架设拨号服务器却没有对账号认证等严格限制，用户安全意识不强，将自己的账号随意转借他人或与别人共享等，这些管理上的问题无论多么高超的安全技术都不能解决，都会使信息系统处于危险之中。如果没有健全的安全性规章或者安全性规章不能贯彻落实，即便设计和实现了再好的安全设备和系统，信息系统安全也不过是空谈而已。

所以建立定期的安全检测、口令管理、人员管理、策略管理、备份管理、日志管理等一系列安全性规章并认真贯彻执行对于维护信息系统的安全来说是非常必要的。

为了更好地落实安全性规章，首先需要根据实际情况，建立和健全信息系统安全委员会、安全小组、安全员。安全组织成员应当由主管领导、安全保卫、信息中心、人事、审计等部门的工作人员组成，必要时可聘请相关部门的专家组成。如果有必要，安全组织也可成立专门的独立机构。设立信息安全部门和安全人员，不但可以有效地制定并贯彻落实安全性规章制度，还可以提高对安全事件的反应能力和响应速度。

有了信息安全部门和人员，还要制定安全管理制度。只有建立健全的安全管理制度，并在信息系统的运行过程中始终坚持贯彻执行，才能从根本上为信息系统的正常运行，以及信息系统安全技术的执行提供良好的、坚固的基础。安全管理制度应该包括下面一些主要方面的内容：

（1）机房安全管理制度。

（2）系统运行管理制度，包括系统启动、关闭、系统状态监控、系统维护等。

（3）人员管理制度，包括管理人员、设计人员、操作人员、人事变更等。

（4）软件管理制度。

（5）数据管理制度。

（6）密码口令管理制度。

（7）病毒防治管理制度。

（8）用户登记和信息管理制度。

（9）工作记录制度。

（10）数据备份制度。

（11）审计制度。

（12）安全培训制度等。此外，有了制度而不认真执行等于没有制度，所以，只有在系统的运行过程中，管理人员、操作人员、用户之间的相互配合与相互协作，共同遵守既定的安全性规章，才能保证信息系统的安全措施是有用的、有效的。

总之，信息安全所涉及的方面很多，只有在各个方面都进行全面管理，才能在此基础上，与所采用的安全技术和设备一起，有效保证信息系统安全。

16.9.2 计算机犯罪与相关法规

随着计算机技术的不断发展，针对信息系统的各种入侵和攻击事件也与日俱增，而且由此带来的影响和损失也越来越大，有些事件甚至已经严重地危害到国家安全、经济发展和社会稳定。因此，提高信息安全性已经不再仅仅局限于采取适当的安全技术措施，完善安全性规章制度，更需要正确地运用法律手段来对付日益严重的计算机犯罪，避免重大损失的问题。

1．计算机犯罪

所谓计算机犯罪是指针对和利用计算机系统，通过非法操作或者以其他手段，对计算机系统的完整性或正常运行造成危害的行为。

计算机犯罪的犯罪对象是计算机系统或其中的数据，包括计算机设备、系统程序、文本资料、运算数据、图形表格等。所谓非法操作，是指一切没有按照操作规程或是超越授权范围而对计算机系统进行的操作。非法操作是对计算机系统造成损害的直接原因。

计算机犯罪是随着计算机技术的发展而出现和发展的，在不同的历史时期，具有不同的特点。大体上，计算机犯罪可以划分成两个阶段。

第一个阶段是计算机单机时代，即早期的电脑犯罪阶段，时间大致从 20 世纪 50 年代

至 80 年代。这个时期的主要形式是计算机诈骗，针对计算机内部信息的窃取和破坏。

第二个阶段是计算机网络时代，时间大致从 20 世纪 80 年代到现在。在这个时期，由于计算机网络的迅速发展及其应用范围越来越广泛，而且计算机软件日益复杂化、普及化，计算机犯罪呈现出一些新的特点：

（1）呈现国际化趋势。互联网的发展是跨越国界的，随之而来的就是计算机犯罪由区域性犯罪向跨地区、跨国界的国际性犯罪发展。

（2）从犯罪所针对的对象看，向全社会各单位和个人蔓延。计算机犯罪由早期的主要攻击金融系统、政府机关向攻击其他所有行业、所有部门的信息系统蔓延；由攻击单位、团体的信息系统向攻击个人信息系统蔓延。这两种趋势的出现都是因为计算机已经从早期的特殊部门向全社会众多机关团体以及个人普及。

（3）从组织形式上看，由个人犯罪向群体犯罪、组织犯罪发展；由单一目的犯罪向综合性犯罪发展。

（4）从犯罪主体看，所涉及人员范围越来越广泛，并呈现低龄化趋势。从年龄结构来看，低龄化、普遍化是主要特点。从犯罪人员素质层次看，已经从早期的高学历、高技能型向普通人群发展。这些也都是因为计算机技术的普及，使得越来越多的人能够方便地学习到更多的计算机技术，通过长时间的学习和实践，青少年、低学历人员也能够逐渐掌握这些技术，成为计算机和网络犯罪的主体。

（5）从危害程度看，后果越来越严重。由于知识经济的发展，各企事业单位的日常业务越来越依赖于信息系统，大量政治、军事、经济等方面的重要文件和数据，以及大量的社会财富集中于信息系统中，例如网络银行、股票等往往就表现为计算机系统中账户上的数据。一旦犯罪分子侵入这样的信息系统，必将对国家安全、经济发展、社会进步产生巨大的影响，甚至造成不可挽回的损失。

（6）通过网络窃取机密信息将成为间谍活动的主要形式之一。随着越来越多的企事业单位和个人连接互联网，其中的很多机密信息和数据都面临着网络窃密行为的威胁。对于没有采取严格安全措施的系统，通过网络窃取其机密信息相对于其他方式更加隐蔽、快捷。例如，通过后门程序盗窃用户的账号和密码，通过系统漏洞取得系统特权，非法窃取商业机密等。

这些计算机犯罪行为显然具有很大的危害，它们影响社会的稳定，危及国家安全，扰乱经济秩序，影响社会治安，妨害青少年的健康成长，阻碍高科技产业的健康发展。因此，对于各种形式的计算机犯罪必须运用法律手段进行打击和惩处。加大对网络犯罪的打击力度，

是保证我国社会稳定、经济持续发展的一项重要任务。

2．我国的相关法律、法规

计算机犯罪，已经成为刑事犯罪的一种新形式。我国《刑法》已经增加了计算机犯罪的相关内容，并将计算机犯罪分为 5 种类型。一类是直接以计算机信息系统为犯罪对象的犯罪，另一类是以计算机为犯罪工具实施其他犯罪。具体的，《刑法》关于计算机犯罪的规定有：

第二百八十五条（非法侵入计算机信息系统罪）违反国家规定，侵入国家事务、国防建设、尖端科学技术领域的计算机信息系统的，处三年以下有期徒刑或者拘役。

第二百八十六条（破坏计算机信息系统罪）违反国家规定，对计算机信息系统功能进行删除、修改、增加、干扰，造成计算机信息系统不能正常运行，后果严重的，处 5 年以下有期徒刑或者拘役；后果特别严重的，处 5 年以上有期徒刑。违反国家规定，对计算机信息系统中存储、处理或者传输的数据和应用程序进行删除、修改、增加的操作，后果严重的，依照前款的规定处罚。故意制作、传播计算机病毒等破坏性程序，影响计算机系统正常运行，后果严重的，依照第一款的规定处罚。

第二百八十七条（利用计算机实施的各类犯罪）利用计算机实施金融诈骗、盗窃、贪污、挪用公款、窃取国家秘密或者其他犯罪的，依照本法有关规定定罪处罚。

这些规定对于我国大规模地推广应用各种信息系统，对于保护信息系统的生产者和使用者的合法权益，对于信息系统的安全运作都具有极为重要的作用。

除了《刑法》之外，我国在信息系统安全方面，自 1994 年以来，国务院及其有关部委相继修改和出台了若干相关法规和管理规定，其中对于我国境内发生的各种计算机犯罪及其处罚都有明文规定。因此，为了做好信息系统安全，有必要详细了解这些法律、法规，包括《中华人民共和国宪法》、《中华人民共和国刑法》、《中华人民共和国国家安全法》、《中华人民共和国保守国家秘密法》、《中华人民共和国计算机信息系统安全保护条例》、《中华人民共和国计算机信息网络国际联网管理暂行规定》、《中华人民共和国治安管理处罚条例》《中华人民共和国计算机信息网络国际联网管理暂行规定实施办法》、《中华人民共和国专利法》、《中华人民共和国反不正当竞争法》、《中华人民共和国商标法》、《中华人民共和国海关法》、《中华人民共和国标准化法》、《关于对<中华人民共和国计算机信息系统安全保护条例>中涉及的“有害数据”问题的批复》、《科学技术保密规定》、《计算机信息系统安全专用产品检测和销售许可证管理办法》、《公安部关于对与国际联网的计算机信息系统进行备案工作的通知》、《计算机信息网络国际联网安全保护管理办法》、《电子出版物管理规定》、《中国互联网

络域名注册暂行管理办法》、《从事放开经营电信业务审批管理暂行办法》、《计算机信息网络国际联网出入口信道管理办法》、《中国公用计算机互联网国际联网管理办法》、《中国公众多媒体通信管理办法》、《计算机软件保护条例》、《商用密码管理条例》、《计算机信息系统国际联网保密管理规定》、《计算机病毒防治管理办法》、《信息安全等级保护管理办法》等。

第 17 章：系统的可靠性分析与设计

系统的可靠性分析与设计是系统架构设计师在系统分析与设计阶段、系统集成阶段应该重点考虑的问题。内容主要为可靠性设计、系统的故障模型、系统的可靠性模型、组合模型可靠性计算、马尔柯夫模型可靠性计算，以及硬件冗余、信息校验码等方面；另外也涉及系统可靠性分析与计算、系统可靠性评估和系统配置方法等概念与理论的实际工程运用等内容。

17.1 可靠性概述

与可靠性相关的概念主要有：可靠度、可用度、可维度、平均无故障时间、平均故障修复时间及平均故障间隔时间等。

（1）可靠度。系统的可靠度 R(t)是指在 t=0 时系统正常的条件下，系统在时间区间[0,t]内能正常运行的概率。

（2）可用度。系统的可用度 A(t)是指系统在时刻 t 可运行的概率。

（3）可维度。系统的可维度 M(t)是指系统失效后，在时间间隔内被修复的概率。

（4）平均无故障时间。可靠度为 R(t)的系统平均无故障时间（Mean Time To Failure，MTTF）定义为从 t=0 时到故障发生时系统的持续运行时间的期望值：

$$\mathrm{MTTF}=\int_0^{\infty} R(t)\mathrm{d}t$$

如果 $R(t)=\mathrm{e}^{-\lambda t}$ 则：

$$\mathrm{MTTF}=1/\lambda$$

式中 λ 为失效率，是指器件或系统在单位时间内发生失效的预期次数，在此处假设为常数。

（5）平均故障修复时间。可用度为 A(t)的系统平均故障修复时间（Mean Time To Repair，MTTR）可以用类似于求 MTTF 的方法求得。

设 A1(t)是在风险函数 Z(t)=0 且系统的初始状态为 1 状态的条件下 A(t)的特殊情况，

则：

$$\mathrm{MTTR} = \int_0^{\infty} A_1(t)\mathrm{d}t$$

此处假设修复率 μ(t) = μ（常数），修复率是指单位时间内可修复系统的平均次数，则：

MTTR =1/ μ

（6）平均故障间隔时间。平均故障间隔时间（Mean Time Between Failure，MTBF）常常与 MTTF 发生混淆。因为两次故障（失败）之间必然有修复行为，因此，MTBF 中应包含 MTTR。对于可靠度服从指数分布的系统，从任一时刻 t 0 到达故障的期望时间都是相等的，因此有：

MTBF = MTTR + MTTF

17.2 系统故障模型

任何系统，不管现在运行得多么稳定，总有发生故障的时候。那么，一般的系统故障遵循一个什么样的规律呢？这就是本节要讨论的内容。

17.2.1 故障的来源以及表现

下面先介绍几个概念。

（1）失效：硬件的物理改变。

（2）故障：由于部件的失效、环境的物理干扰、操作错误或不正确的设计引起的硬件或软件中的错误状态。

（3）错误（差错）：故障在程序或数据结构中的具体位置。错误与故障位置之间可能出现一定距离。故障或错误有如下几种表现形式。

永久性：描述连续稳定的失效、故障或错误。在硬件中，永久性失效反映了不可恢复的物理改变。

间歇性：描述那些由于不稳定的硬件或变化着的硬件或软件状态所引起的、仅仅是偶然出现的故障或错误。

瞬时性：描述那些由于暂时的环境条件而引起的故障或错误。

一个故障可能由物理失效、不适当的系统设计、环境影响或系统的操作员所引起。永久性失效会导致永久性故障。间歇性故障可能由不稳定、临界稳定或不正确的设计所引起。环境条件会造成瞬时性故障。所有这些故障都可能引起错误。不正确的设计和操作员失误会直接引起错误。由硬件的物理条件，不正确的硬件或软件设计，或不稳定但重复出现的环境条件所引起的故障可能是可检测的，并且可以通过替换或重新设计来修复；然而，由于暂时的环境条件所引起的故障是不能修复的，因为其硬件本身实际上并没有损坏。瞬时和间歇故障已经成为系统中的一个主要错误源。

17.2.2 几种常用的故障模型

故障的表现形式千差万别，可以利用故障模型对千差万别的故障表现进行抽象。故障模型可以在系统的各个级别上建立。一般说来，故障模型建立的级别越低，进行故障处理的代价也越低，但故障模型覆盖的故障也越少。如果在某一级别的故障模型不能包含故障的某些表现，则可以用更高一级别的模型来概括。下面介绍几种常用的故障模型。

1．逻辑级的故障模型

固定型故障指电路中元器件的输入或输出等线的逻辑固定为 0 或固定为 1，如某线接地、电源短路或元件失效等都可能造成固定型故障。短路故障是指一个元件的输出线的逻辑值恒等于输入线的逻辑值；元件的开路故障是元件的输出线悬空，逻辑值可根据具体电路来决定。桥接故障指两条不应相连的线连接在一起而发生的故障。

2．数据结构级的故障

故障在数据结构上的表现称为差错。常见的差错如下。

独立差错：一个故障的影响表现为使一个二进制位发生改变。

算术差错：一个故障的影响表现为使一个数据的值增加或减少 $2i(i=0,1,2,\cdots)$。

单向差错：一个故障的影响表现为使一个二进制向量中的某些位朝一个方向（0 或 1）改变。

3．软件故障和软件差错

软件故障是指软件设计过程造成的与设计说明的不一致的情况，软件故障在数据结构或程序输出中的表现称为软件差错。与硬件不同，软件不会因为环境压力而疲劳，也不会因为时间的推移而衰老。因此，软件故障只与设计有关。常见的软件差错有以下几种。

非法转移：程序执行了说明中不存在的转移。

误转移：程序执行了尽管说明中存在，但依据当前控制数据不应进行的转移。

死循环：程序执行时间超过了规定界限。

空间溢出：程序使用的空间超过了规定的界限。

数据执行：指令计数器指向数据单元。

无理数据：程序输出的数据不合理。

4．系统级的故障模型故障在系统级上的表现为功能错误，即系统输出与系统设计说明的不一致。如果系统输出无故障保护机构，则故障在系统级上的表现就会造成系统失效。

17.3 系统配置方法

容错技术是保证系统在某些组成部分出现故障或差错时仍能正常工作的技术。通常根据不同的系统配置方法而采用相应容错技术：单机容错技术、双机热备份技术和服务器集群技术。

17.3.1 单机容错技术

容错技术是保证系统在某些组成部分出现故障或差错时仍能正常工作的技术。系统的故障可分为两类：一类是“致命的”，不可能自行修复，例如系统的主要部件全部损坏；另一类是局部的，可能被修复，例如部分元件失效、线路故障、偶然干扰引起的差错等。容错技术正是用于构造一种能够自动排除非致命性故障的系统，即容错系统。

在单机容错技术中，提高系统工作可靠性的方法主要有自检技术和冗余技术。容错又有多种形式，如硬件容错、软件容错、整机容错等。

1．自检技术

自检指系统在发生非致命性故障时能自动发现故障和确定故障的性质、部位，并自动采取措施更换和隔离产生故障的部件。自检需采用诊断技术，常用专门程序实现，属于程序设计的范围。容错系统的实现要求系统必须具有重复部件或备份部件，或具有不止一个完成某种功能的通道。因此自检技术常配合冗余技术使用。计算机的容错系统一般都需要应用自检技术。

2．冗余技术

冗余可分为硬件冗余（增加硬件）、软件冗余（增加程序，如同时采用不同算法或不同人编制的程序）、时间冗余（如指令重复执行、程序重复执行）、信息冗余（如增加数据位）等。冗余技术中最常用的两种方法是重复线路和备份线路。重复线路指用多个相同品种和规

格的元件或构件并联起来，当作一个元件或构件使用，只要有一个不出故障，系统就能够正常工作。在并联工作时每一个构件的可靠性概率是互相独立的。备份线路与重复线路的差别是参加备份的构件并不接入系统，只有在处于工作状态的构件发生故障后才把输入和输出接到备份构件上来，同时切断故障构件的输入、输出。

容错技术已获得广泛应用，常用于对可靠性要求高的系统，特别是用于危及人身安全的关键部位。在这些部位大多采用双重冗余，也有采用三重、四重甚至五重冗余的。现代的大型复杂系统常常是容错能力很强的系统。容错技术在计算机中应用得最早、最广泛。

17.3.2 双机热备份技术

双机热备份技术是一种软硬件结合的较高容错应用方案。该方案是由两台服务器系统和一个外接共享磁盘阵列柜和相应的双机热备份软件组成。其中的外接共享磁盘阵列柜也可以没有，而是在各自的服务器中采取 RAID（Redundant Array of Independent Disk，独立冗余磁盘阵列）卡。

在这个容错方案中，操作系统和应用程序安装在两台服务器的本地系统盘上，整个网络系统的数据是通过磁盘阵列集中管理和数据备份的。数据集中管理是通过双机热备份系统，将所有站点的数据直接从中央存储设备读取和存储，并由专业人员进行管理，极大地保护了数据的安全性和保密性。用户的数据存放在外接共享磁盘阵列中，在一台服务器出现故障时，备机主动替代主机工作，保证网络服务不间断。

双机热备份系统采用“心跳”方法保证主系统与备用系统的联系。所谓“心跳”，指的是主、从系统之间相互按照一定的时间间隔发送通信信号，表明各自系统当前的运行状态。一旦“心跳”信号表明主机系统发生故障，或者备用系统无法收到主机系统的“心跳” 信号，则系统的高可用性管理软件认为主机系统发生故障，立即将系统资源转移到备用系统上，备用系统替代主机工作，以保证系统正常运行和网络服务不间断。

双机热备份方案中，根据两台服务器的工作方式可以有三种不同的工作模式，即：双机热备模式、双机互备模式和双机双工模式。

双机热备模式即目前通常所说的 active/standby 方式，active 服务器处于工作状态；而 standby 服务器处于监控准备状态，服务器数据包括数据库数据同时往两台或多台服务器写入（通常各服务器采用 RAID 磁盘阵列卡），保证数据的即时同步。当 active 服务器出现故障时，通过软件诊测或手工方式将 standby 机器激活，保证应用在短时间内完全恢复正常

使用。典型应用有证券资金服务器或行情服务器。这是目前采用较多的一种模式，但由于另外一台服务器长期处于后备的状态，所以浪费了一部分计算资源。

用户可以根据系统的重要性及终端用户对服务中断的容忍程度决定是否使用双机热备份。例如，网络中的用户最多能容忍多长时间恢复服务，如果服务不能很快恢复会造成什么样的后果作为是否采用双机热备份的根据。对于承担企业关键业务应用的服务器需要极高的稳定性和可用性，并需要提供每周 7（天）×24（小时）不间断服务的应用，推荐使用双机热备份。

双机互备模式，是两个相对独立的应用在两台机器同时运行，但彼此均设为备机，当某一台服务器出现故障时，另一台服务器可以在短时间内将故障服务器的应用接管过来，从而保证了应用的持续性，但对服务器的性能要求比较高。

双机双工模式是集群的一种形式，两台服务器均处于活动状态，同时运行相同的应用，以保证整体系统的性能，也实现了负载均衡和互为备份，通常使用磁盘柜存储技术。Web 服务器或 FTP 服务器等用此种方式比较多。

17.3.3 服务器集群技术

集群技术指一组相互独立的服务器在网络中组合成为单一的系统工作，并以单一系统的模式加以管理。此单一系统为客户工作站提供高可靠性的服务。大多数情况下，集群中所有的计算机拥有一个共同的名称，集群内任一系统上运行的服务可被所有的网络客户使用。

集群必须可以协调管理各分离的构件出现的错误和故障，并可透明地向集群中加入构件。一个集群包含多台（至少二台）共享数据存储空间的服务器。其中任何一台服务器运行应用时，应用数据被存储在共享的数据空间内。每台服务器的操作系统和应用程序文件存储在其各自的本地储存空间上。

集群内各节点服务器通过一个内部局域网相互通信，当一台节点服务器发生故障时，这台服务器上所运行的应用程序将在另一节点服务器上被自动接管。当一个应用服务发生故障时，应用服务将被重新启动或被另一台服务器接管。当以上的任一故障发生时，客户都将能很快连接到其他应用服务器上。

17.4 系统可靠性模型

与系统故障模型对应的就是系统的可靠性模型。人们经常说某系统“十分可靠”，那么

这个“十分”究竟如何衡量呢？下面介绍几种常用的模型。

17.4.1 时间模型

最著名的时间模型是由 Shooman 提出的可靠性增长模型，这个模型基于这样一个假设：一个软件中的故障数目在 t = 0 时是常数，随着故障被纠正，故障数目逐渐减少。

在此假设下，一个软件经过一段时间的调试后剩余故障的数目可用下式来估计：

$$E_r(\tau)=\frac{E_0}{\mathrm{I}-E_c(\tau)}$$

其中，τ 为调试时间，Er(τ) 为在时刻 τ 软件中剩余的故障数，E0 为 τ = 0 时软件中的故障数，Er(τ) 为在[0，τ]内纠正的故障数，I 为软件中的指令数。

由故障数 Er(τ) 可以得出软件的风险函数：

$$Z(t)=C\cdot Er(\tau)$$

其中 C 是比例常数。于是，

软件的可靠度为：

$$R(t)=e^{-\int_0^t z(t)\mathrm{d}t}=e^{-\mathrm{c}(E_0/I-E_c(\tau))}$$

软件的平均无故障时间为：

$$\mathrm{MTBF}=\int_0^{\infty}R(t)\mathrm{d}t=\frac{1}{\mathrm{C}(E_0/\mathrm{I}-E_c(\tau))}$$

希赛教育专家提示：在 Shooman 的模型中，需要确定在调试前软件中的故障数目，这往往是一件很困难的任务。

17.4.2 故障植入模型

故障植入模型是一个面向错误数的数学模型，其目的是以程序的错误数作为衡量可靠性的标准，模型的原型是 1972 年由 Mills 提出的。

Mills 提出的故障植入模型的基本假设如下：

（1）程序中的固有错误数是一个未知的常数。

（2）程序中的人为错误数按均匀分布随机植入。

（3）程序中的固有错误数和人为错误被检测到的概率相同。

（4）检测到的错误立即改正。

用 N0 表示固有错误数，N1 表示植入的人为错误数，n 表示检测到的错误数，ξ 表示被检测到的错误中的人为错误数，则：

$$P_r\{\xi = y, N_0, N_1, n\} = \frac{\binom{N_1}{y}\binom{N_0}{n-y}}{\binom{N_1+N_0}{n}}$$

对于给定的 N1，n，在测试中检测到的人为错误数为 k，用最大似然法求解可得固有错误数 N0 的点估计值为：

$$\hat{N}_0 = \left.\frac{N_1(n-k)}{k}\right|_{integer}$$

考虑到实施植入错误时遇到的困难，Basin 在 1974 年提出了两步查错法，这个方法是由两个错误检测人员独立对程序进行测试，检测到的错误立即改正。用 N0 表示程序中的固有错误数，N1 表示第一个检测员检测到的错误数，n 表示第二个检测员检测到的错误数，用随机变数 η 表示两个检测员检测到的相同的错误数，则：

$$P_r\{\eta = y, N_0, N_1, n\} = \frac{\binom{N_1}{y}\binom{N_0-N_1}{n-y}}{\binom{N_0}{n}}$$

如果实际测得的相同错误数为 k，则程序固有错误数 N 0 的点估计值为：

$$\hat{N}_0 = \left.\frac{N_1 n}{k}\right|_{integer}$$

17.4.3 数据模型

在数据模型下，对于一个预先确定的输入环境，软件的可靠度定义为在 n 次连续运行中软件完成指定任务的概率。

最早的一个数据模型是 Nelson 于 1973 年提出的，其基本方法如下：

设说明所规定的功能为 F，程序实现的功能为 F′ ，预先确定的输入集。

E={ ei:i=1,2,- - -,N }

令导致软件差错的所有输入的集合为 Ee ，即:

$$E_e = \{e_j : e_j \in E \text{ and } F'(e_j) \neq F(e_j)\}$$

则软件运行一次出现差错的概率为：

P1=|Ee|/|E|

一次运行正常的概率为：

R1=1−P1=1−Ee /E

在上述讨论中，假设了所有输入出现的概率相等，如果不相等，且 ei 出现的概率为 pi(i=1,2,- - -,n)，则软件运行一次出现差错的概率为：

$$p_1 = \sum_{i=1}^{n} (Y_i \cdot p_i)$$

其中：

$$Y_i = \begin{cases} 0 & \text{如果 } F'(e_i) = F(e_i) \\ 1 & \text{如果 } F'(e_i) \neq F(e_i) \end{cases}$$

于是，软件的可靠度（n 次运行不出现差错的概率）为：

R(n)=R1n=(1-P1)n

只要知道每次运行的时间，上述数据模型中的 R(n)就很容易转换成时间模型中的 R(t)。

17.5 系统的可靠性分析和可靠度计算

本节介绍如何计算一个组合系统可靠性的几种方法。

17.5.1 组合模型

组合模型是计算机容错系统可靠性最常用的方法。一个系统只要满足以下条件，就可以用组合模型来计算其可靠性。作如下假设。

（1）系统只有两种状态：运行状态和失效状态。

（2）系统可以划分成若干个不重叠的部件，每个部件也只有两种状态：运行状态和失效状态。

（3）部件的失效是独立的。

（4）系统失效当且仅当系统中的剩余资源不满足系统运行的最低资源要求（系统的状态只依赖于部件的状态）时。

（5）已知每个部件的可靠性，可靠性指可用度或可靠度等概率参数。组合模型的目标就是根据各部件的可靠性 Ri(t)来计算系统的可靠度 Rsys(t)，组合模型的基本思想如下。

1．枚举所有系统状态

假设系统被划分为 n 个部件，则系统状态是一个 n 维向量，q = (s1,s2 ,---,sn) ，其中：

si={0，如果部件 i 处于运行状态；1，如果部件 i 处于失效状态（ i = 1, 2,---, n ）}，一个具有 n 个部件的系统共有 2 n 个状态。

2. 计算每个系统状态的概率系统状态的概率是指系统处于该状态的概率。设系统状态 q = (s1,s2 ,---,sn) ，q 的所有 0

分量对应的部件用 A0 来表示（ A0 是所有处于运行状态的部件的集合），q 的所有 1 分量对应的部件用 A1 来表示（ A1 是所有处于失效状态的部件的集合）。于是，系统状态的概率为：

$$P_q = (\prod_{i \in A_0} R_i)(\prod_{j \in A_1} (1 - R_j))$$

3．可靠性计算直接计算一个复杂系统的可靠性是很困难的，通常的方法是把整个系统分解为简单的子系统，通过子系统的组合来计算整个系统的可靠性。

（1）串联系统。在一个由 n 个模块（部件）构成的系统中，如果系统中任意一个模块失效将导致系统失败（系统的最低资源要求是所有模块全部运行，只有全 0 系统（0，0，…，0）能够使系统运行）。

用随机变量 ξi 表示模块 i 发生失效的时间，用随机变量 ξs 表示系统发生失效的时

间，则 ξs 可表示为：

$$\xi_s = \min(\xi_1, \xi_2, \cdots, \xi_n)$$

则系统可靠度为：

$$R_{\text{sys}}(t) = P_r(\xi_s > t) = P_r(\min(\xi_1, \xi_2, \cdots, \xi_n) > t)$$

$$= P_r(\xi_1 > t) P_r(\xi_2 > t) \cdots P_r(\xi_n > t) = \prod_{i=1}^{n} R_i(t)$$

其中， Ri(t)是模块 i 的可靠度，串联系统的可靠度是各个模块可靠度的乘积。这种系统可抽象地看成一个如图 17-1 示的串联系统，因此，上式称为串联可靠性公式。

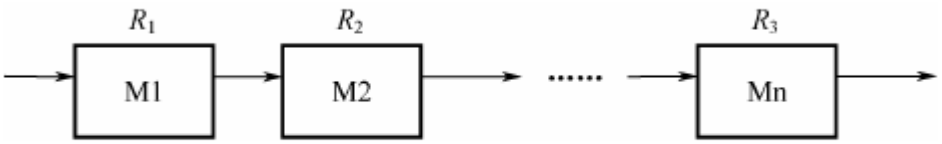


图 17-1　串联系统

串联系统的失效率为：

$$Q_{\text{sys}}(t) = 1 - \prod_{i=1}^{n} R_i(t) = 1 - \prod_{i=1}^{n} (1 - Q_i(t))$$

其中， Qi(t) =1− Ri(t) 是模块 i 的失效概率。

（2）并联系统。在一个由 n 个模块（部件）构成的系统中，只要有一个模块可运行，系统就可运行（系统的基本资源是一个模块，除了全 1 系统状态（1，1，1，…，1）外，系统都是可运行的），因此：

$$\xi_s = \max(\xi_1, \xi_2, \cdots, \xi_n)$$

系统的失效概率分布函数可以表示为：

$$Q_{sys}(t) = P_r(\xi \leqslant t) = P_r(\max(\xi_1, \xi_2, \cdots, \xi_n) \leqslant t)$$
$$= \prod_{i=1}^{n} P_r(\xi_i \leqslant t) = \prod_{i=1}^{n} Q_i(t)$$

其中 Qi(t) 是模块 i 的失效分布函数。

并联系统的可靠度为：

$$R_{sys}(t) = 1 - Q_{sys}(t) = 1 - \prod_{i=1}^{n} Q_i(t) = 1 - \prod_{i=1}^{n} (1 - R_i(t)) \quad (16.27)$$

其中，Qi(t)=1-Ri(t) 是模块 i 的失效概率

这种系统可抽象地看成一个如图 17-2 所示的并联系统，因此，上式称为并联可靠性公式。

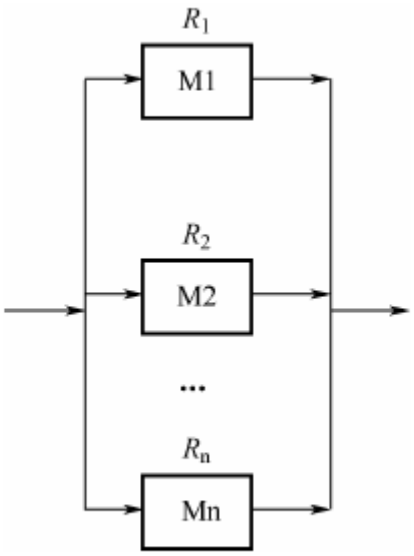


图 17-2 并联系统

（3）串并联系统。如果一个系统由 N 个子系统并联而成，而每个并联的子系统又由 n 个元件串联而成，这样的系统称为串并联系统。

设第 j 个子系统的第 i 个元件的可靠度为 Rij(i=1,2,L,n;j=1,2,L,N)，则该串并联系统的可靠度为：

$$R_{sp}=1-\prod_{j=1}^{N}(1-\prod_{i=1}^{n}R_{ij})$$

如果 Rij 全相等为 R，则：

$$R_{sp}=1-(1-R^{n})^{N}$$

17.5.2 马尔柯夫模型

马尔柯夫模型的两个核心概念是状态和状态转移。系统的状态表示了在任何瞬间用以描述该系统所必须知道的一切。对于可靠性分析，马尔柯夫模型的每个状态表示了有效和失效模块的不同组合。如果每个模块都是处于有效和失效两种情况之一，则一个 n 模块系统的完整模型有 2 n 个状态。

状态转移是指随着时间的流逝，因模块的失效和修复，系统发生的状态变化。

作为马尔柯夫模型基础的基本假设是：给定状态的转移概率仅取决于当前的状态。系统从一个状态 i 转移到另一个状态 j 的转移率定义为单位时间内从状态 i 转移到状态 j 的概率。对于一个模块来说，从运行状态到失效状态的转移率就是模块的失效率，从失效状态到运行状态的转移率就是模块的修复率。一个失效率为 λ，修复率为 μ 的模块的状态图如图 17-3 所示。

对于由 n 个模块构成的系统，共有 2 n 个状态。从理论上说，任意两个状态之间都存在转移的可能性。但因失效是独立的，在很短的时间内发生多个失效的可能性远小于发生一个失效的可能性。因此，只考虑任一时刻只有一个模块失效的转移；同样，也只考虑任意时刻只有一个模块修复的转移。系统的状态图也可以表示为层次图。第一层只有一个状态，对应于所有模块都运行的情况；第二层有 n 个状态，对应于一个模块失效的各种情况；第 i +1 层有 Ci 个状态，对应于 n 个模块中有 i 个失效的各种情况；第 n+1 层也只有一个状态，对应于全部模块都失效的情况。

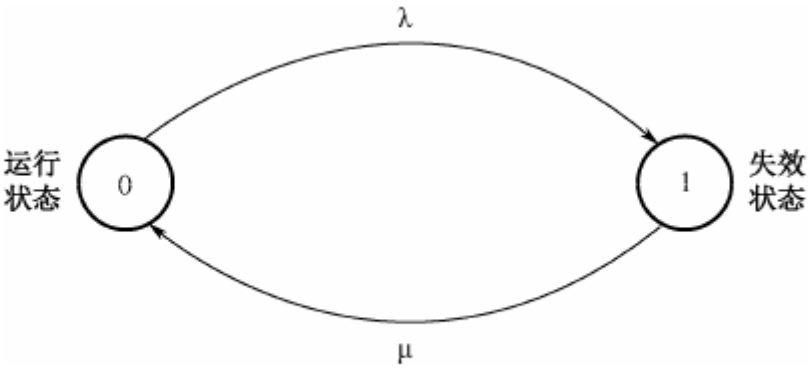


图 17-3　（λ，μ）模块的状态图

根据系统的状态图，可以计算出系统处于任意状态的概率。

设系统在 t 时刻处于状态 0 和 1 的概率分别为 P0(t)和 P1(t)，于是，在 t + Δt 时刻系统处于 0 状态的概率为：

$$P_0(t+\Delta t)=P_0(t)-P_0(t)\cdot\lambda\cdot\Delta t+P_1(t)\cdot\mu\cdot\Delta t$$

同样，在 t+ Δt 时刻系统处于 1 状态的概率为：

$$P_1(t+\Delta t)=P_1(t)+P_0(t)\cdot\lambda\cdot\Delta t-P_1(t)\cdot\mu\cdot\Delta t$$

令 Δt → 0 取极限，得微分方程组：

$$\begin{bmatrix}\dot{P}_0(t)\\ \dot{P}_1(t)\end{bmatrix}=\begin{bmatrix}-\lambda & \mu\\ \lambda & -\mu\end{bmatrix}\begin{bmatrix}P_0(t)\\ P_1(t)\end{bmatrix}$$

其中，Pi(t) 是 Pi(t) 对 t 的一阶导数(i = 0,1)。

只要解此微分方程组就可以得出 P0(t)和 P1(t)。

对于有 n 个状态的状态图，设状态 i 到 j 的转移率为 α aij。考虑其中的任意一个状态 j，其他状态到 j 的转移和 j 到其他状态的转移，系统在 t+ Δt 时刻，处于状态 j 的概率可以表示为：

$$P_j(t+\Delta t)=P_j(t)-\sum_{\substack{i=1\\ i\neq j}}^{n}(P_j(t)\cdot\alpha_{ji}\cdot\Delta t)+\sum_{\substack{i=1\\ i\neq j}}^{n}(P_i(t)\cdot\alpha_{ij}\cdot\Delta t)$$

由此可得：

$$\dot{P}_j(t)=\sum_{\substack{i=1\\ i\neq j}}^{n}(P_i(t)\cdot\alpha_{ij})-\left(\sum_{\substack{i=1\\ i\neq j}}^{n}\alpha_{ji}\right)\cdot P_j(t)\qquad j=1,2,\cdots,n$$

用矩阵方程把 Pj(t)(j = 1, 2, , n) 全部表示出来就是：

$$P(t)=T\cdot P(t)$$

或

$$\begin{bmatrix}\dot{P}_1(t)\\ \dot{P}_2(t)\\ \dot{P}_3(t)\\ \vdots\\ \dot{P}_n(t)\end{bmatrix}=\begin{bmatrix}-\beta_1 & \alpha_{21} & \alpha_{31} & \cdots & \alpha_{n1}\\ \alpha_{12} & -\beta_2 & \alpha_{32} & \cdots & \alpha_{n2}\\ \alpha_{13} & \alpha_{23} & -\beta_3 & \cdots & \alpha_{n3}\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ \alpha_{1n} & \alpha_{2n} & \alpha_{3n} & \cdots & -\beta_n\end{bmatrix}\cdot\begin{bmatrix}P_1(t)\\ P_2(t)\\ P_3(t)\\ \vdots\\ P_n(t)\end{bmatrix}$$

其中， T 称为状态转移矩阵，其对角线上的元素：

$$\beta_j=\sum_{\substack{i=1\\ i\neq j}}^{n}\alpha_{ji}(j=1,2,\cdots,n)$$

这一矩阵方程称为查普曼—科尔莫戈罗夫（Chapman-Kolmoqorov）方程，由它可解出系统处于任意状态的概率。解方程最常用的是拉普拉斯变换解法。

希赛教育专家提示：马尔柯夫模型是计算系统可靠性的强有力工具，用组合模型能计算的可靠性，用马尔柯夫模型也能计算，马尔柯夫模型还能计算许多组合模型不能计算的可靠性。

17.6 提高系统可靠性的措施

防止故障造成系统失效的两种技术是故障掩蔽技术和系统重组技术，故障掩蔽技术是指防止故障造成差错的各种技术，系统重组技术是防止差错导致系统失效的各种技术。故障掩蔽技术和系统重组技术是达到容错的两种基本途径。而它们又是建立在资源冗余的基础上的。资源冗余有硬件冗余、信息冗余、时间冗余和软件冗余 4 种形式。本节主要介绍前两种形式。

17.6.1 硬件冗余

硬件冗余最常用的是三模冗余（Triple Modular Redundancy，TMR），三个相同的模块接收三个相同的输入，产生的三个结果送至多数表决器。表决器的输出取决于三个输入的多数，若有一个故障模块，则另两个正常模块的输出可将故障模块的输出掩蔽，从而不在表决器输出产生差错。

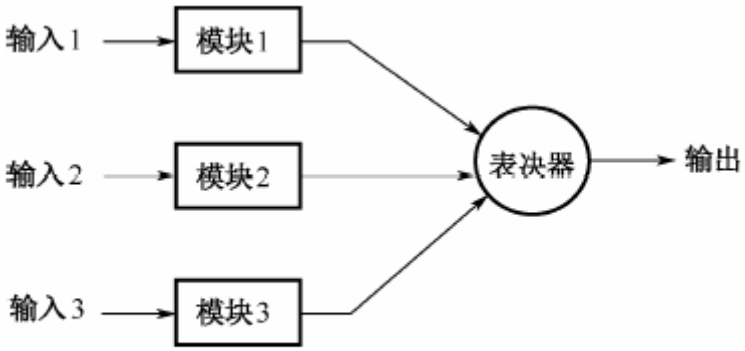


图 17-4　TMR 系统结构图

图 17-4 是一个简单的 TMR 系统，用组合模型可以计算出系统的可靠度为：

$$R = R_v(R_m^3 + 3R_m^2(1-R_m)) = R_v(3R_m^2 - 2R_m^3)$$

由于 Rm=e−λt ，因此：

$$R = R_v(3e^{-2\lambda t} - 2e^{-3\lambda t})$$

其中 Rv 和 Rm 分别表示表决器和模块的可靠度（假设各模块的可靠度相同）。在无修复的屏蔽冗余系统中，当屏蔽冗余因模块中的故障而耗尽时，再发生模块故障将导致输出的错误。假若模块具有修复能力，则系统的可靠性将会大大提高。下面讨论修复对 TMR 结构可靠性的影响。

设模块的失效率为λ，修复率为 μ，TMR 系统可靠度的马尔柯夫模型如图 17-5 所示

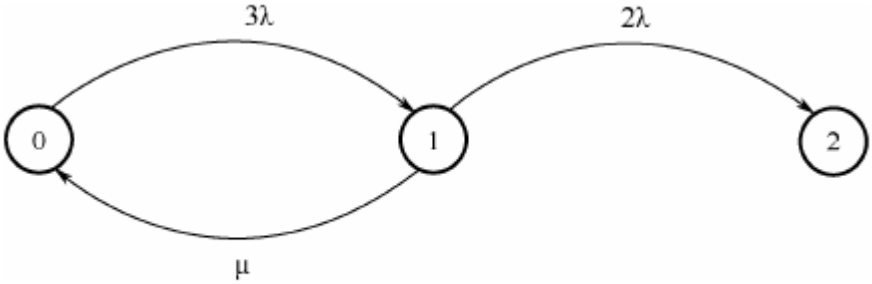


图 17-5 （λ，μ）TMR 的马尔柯夫模型状态图

TMR 系统的拉氏系数矩阵为：

$$\boldsymbol{A}=\begin{bmatrix} s+3\lambda & -\mu & 0 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & s \end{bmatrix}$$

设系统的初态 P0(0)=(100)T ，则：

$$P_2^*(s)=\frac{\begin{vmatrix} s+3\lambda & -\mu & 1 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & 0 \end{vmatrix}}{\begin{vmatrix} s+3\lambda & -\mu & 1 \\ -3\lambda & s+2\lambda+\mu & 0 \\ 0 & -2\lambda & s \end{vmatrix}}$$

$$=\frac{6\lambda^2}{s(s^2+(5\lambda+\mu)s+6\lambda^2)}$$

$$\approx\frac{6\lambda^2}{s^2(s+5\lambda+\mu)}$$

对上式作部分分式分解可得：

$$\frac{6\lambda^2}{s^2(s+5\lambda+\mu)}=-\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s}+\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s^2}+\frac{1}{(5\lambda+\mu)^2}\cdot\frac{1}{s+5\lambda+\mu}$$

查拉氏逆变换表可得：

$$P_2(t)\approx-\frac{1}{(5\lambda+\mu)^2}+\frac{t}{(5\lambda+\mu)}+\frac{1}{(5\lambda+\mu)^2}\cdot e^{-(5\lambda+\mu)t}$$

系统的可靠度：

$$R_{TMR}(t)=1-P_2(t)\approx 1+\frac{1}{(5\lambda+\mu)^2}-\frac{t}{(5\lambda+\mu)}-\frac{1}{(5\lambda+\mu)^2}\cdot e^{-(5\lambda+\mu)t}$$

用同样的方法，可以计算出带修复的 TMR 的可用度 ATMR(t) 。

由 TMR 的完全状态图，可以计算出系统的拉氏系数矩阵为：

$$A=\begin{bmatrix} s+3\lambda & -\mu & 0 & 0 \\ -3\lambda & s+2\lambda+\mu & -2\mu & 0 \\ 0 & -2\lambda & s+2\mu+\lambda & -3\mu \\ 0 & 0 & -\lambda & s+3\mu \end{bmatrix}$$

用马尔柯夫模型可解出：

$$P_0(t)=\frac{\mu^3+3\lambda\mu^2e^{-(\lambda+\mu)t}+3\lambda^2\mu e^{-2(\lambda+\mu)t}+\lambda^3e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_1(t)=\frac{3\lambda\mu^2+3\lambda\mu(2\lambda-\mu)e^{-(\lambda+\mu)t}+3\lambda^2(\lambda-2\mu)e^{-2(\lambda+\mu)t}-3\lambda^3e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_2(t)=\frac{3\lambda^2\mu+3\lambda^2(\lambda-2\mu)e^{-(\lambda+\mu)t}-3\lambda^2(2\lambda-\mu)e^{-2(\lambda+\mu)t}+3\lambda^3e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

$$P_3(t)=\frac{\lambda^3-3\lambda^3e^{-(\lambda+\mu)t}+3\lambda^3e^{-2(\lambda+\mu)t}-\lambda^3e^{-3(\lambda+\mu)t}}{(\lambda+\mu)^3}$$

则 TMR 结构系统的可用度为：

ATMR(t)=P0(t)+P1(t)

对 TMR 函数积分，可得出平均无故障时间 MTTF：

$$\mathrm{MTTF}=\frac{5\lambda+\mu+\sqrt{\lambda^2+10\lambda\mu+\mu^2}}{(5\lambda+\mu)\sqrt{\lambda^2+10\lambda\mu+\mu^2}-\lambda^2-10\lambda\mu-\mu^2}-\frac{5\lambda+\mu-\sqrt{\lambda^2+10\lambda\mu+\mu^2}}{(5\lambda+\mu)\sqrt{\lambda^2+10\lambda\mu+\mu^2}+\lambda^2+10\lambda\mu+\mu^2}$$

简化得：

$$\mathrm{MTTF}=\frac{5}{6\lambda}+\frac{\mu}{6\lambda^2}$$

上面的分析没有考虑表决的可靠性，为了能够容忍表决器的故障，可以对表决器也采用

3 倍冗余。一般说来，对于一个由若干模块构成的系统，可以对每个模块分别实施 TMR 技术。

TMR 的推广是 N 模冗余（N-Modular Redundancy，NMR）。与三模冗余原理相同，但采用 N 个相同的模块， N > 3 ，且 N 为奇数，以方便进行多数表决。NMR 的结构如图 17-6 所示。

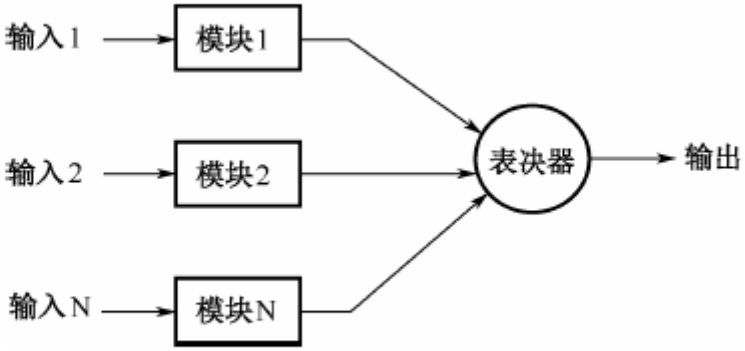


图 17-6　NMR 系统结构图

17.6.2 信息冗余

信息冗余是指通过在数据中附加冗余的信息以达到故障检测、故障掩蔽或容错的目的。应用最广泛的是海明校验码、奇偶校验码。

1．海明校验码

海明校验码是由 Richard Hamming 于 1950 年提出，目前仍然被广泛采用的一种很有效的校验方法，是只要增加少数几个校验位，就能检测出二位同时出错，亦能检测出一位出错并能自动恢复该出错位的正确值的有效手段，后者称为自动纠错。它的实现原理，是在 k 个数据位之外加上 r 个校验位，从而形成一个 k+r 位的新的码字，使新的码字的码距比较均匀地拉大。把数据的每一个二进制位分配在几个不同的偶校验位的组合中，当某一位出错后，就会引起相关的几个校验位的值发生变化，不但可以发现出错，还能指出是哪一位出错，为进一步自动纠错提供了依据。

基本的海明纠错码能纠正一位错。它的原理是基于重叠奇偶校验的概念：将原始数据位分成若干个重叠的组，每组设一位奇偶校验位。由于组间有重叠，因此每位原始数据从属于多于一个组。而且每位原始数据的从属关系是不一样的，纠错时，根据哪些组的奇偶校验位出错，就可以唯一地确定是哪一位数据出错。将该位取反就完成了纠错。

海明校验码是一种特殊的（n，k）线性纠错码，线性纠错码可借助它们的奇偶校验矩阵（Parity-Check Matrix，PCM）来描述。（n，k）线性码的 PCM 是一个(n−k)×n 的矩阵，其元素是 0 和 1。矩阵的每一列与码字中的一个位相对应，而每一行与校验位相对应。

推导并使用长度为 m 位的码字的海明码，所需步骤如下：

（1）确定最小的校验位数 k，将它们记成 D1、D2、…、Dk，每个校验位符合不同的奇偶测试规定。

（2）原有信息和 k 个校验位一起编成长为 m+k 位的新码字。选择 k 校验位（0 或 1）以满足必要的奇偶条件。

（3）对所接收的信息作所需的 k 个奇偶检查。

（4）如果所有的奇偶检查结果均正确，则认为信息无错误。

如果发现有一位或多位错了，则错误的位由这些检查的结果来唯一地确定。

2．循环冗余校验码

循环冗余校验码（Cyclic Redundancy Chec，CRC）也广泛应用于移动通信和磁盘数据存储中。CRC 也是给信息码加上几位校验码，以增加整个编码系统的码距和查错纠错能力。

CRC 的基本原理是：在 K 位信息码后再添加 R 位的校验码，整个编码长度为 N 位，因此，这种编码又称(N,K)码。对于一个给定的(N,K)码，可以证明存在一个最高次幂为 N-K=R 的多项式 G(x)。根据 G(x)可以生成 R 位的校验码，而 G(x)叫作这个 CRC 码的生成多项式。

校验码的具体生成过程为：假设发送信息用信息多项式 C(x)表示，将 C(x)左移 R 位，则可表示成 C(x)×2R 这样 C(x)的右边就会空出 R 位，这就是校验码的位置。通过 C(x)×2R 除以生成多项式 G(x)得到的余数就是校验码。

CRC 码的生成步骤为：

（1）将 x 的最高幂次为 R 的生成多项式 G(x)转换成对应的 R+1 位二进制数。

（2）将信息码左移 R 位，相当于对应的信息多项式 C(x)×2R 。

（3）用生成多项式（二进制数）对信息码做模 2 除，得到 R 位的余数。

（4）将余数拼到信息码左移后空出的位置，得到完整的 CRC 码。

17.7 备份与恢复

在计算机系统中，硬件故障、系统软件和应用软件的故障、操作员的失误，甚至病毒、人为破坏总是不可避免的，为了保证故障发生后，系统能尽快从错误状态恢复到某种逻辑一

致的状态，系统就必须有备份与恢复的机制。

系统的数据备份就是在系统其他地方创建数据与程序的电子复制，为重建系统中被破坏的或不正确的数据提供条件，备份最常用的技术是数据转储和建立日志文件。可以结合这两种技术为系统提供比较好的备份手段。

数据转储可分为静态转储和动态转储。静态转储是指在系统中无事务时进行的转储操作，动态转储是指转储操作与用户事务并发进行，而且转储工作不会影响事务的运行，但它不能保证副本中的数据正确有效。

建立日志文件是指把所有事务对系统的修改活动都登记下来。若发生了故障，对于静态转储，可以在重装后备副本之后，利用日志文件进行恢复，避免重新运行事务；对于动态转储，可以把转储得到的副本和转储期间的日志文件结合起来进行故障恢复，使系统恢复正常工作状态。

备份通常分为联机备份和脱机备份两种方式。

脱机备份也叫冷备份，是一种静态转储技术，备份系统所有的物理文件（控制文件、数据文件、重做日志和归档日志）和初始化文件。这种方式的优点是在恢复过程中步骤最少，它比热备份快并且出错机会少，定期的脱机备份加上一组好的重做日志可以把系统的数据恢复到任何一个时间点上。

联机备份也叫热备份，是一种动态转储技术，由于只备份所需的文件，因而被看作是部分备份。它在系统运行时执行。这种方式的优点是可以实现完全的时间点恢复，同时由于数据库一直处于打开状态，减少了系统对物理资源的要求，改善了数据的执行；但联机备份比较复杂，需要对系统的核心有比较深刻的认识，对备份策略进行反复的测试，才能最终确定它的正确性和可用性。

第 18 章：软件的知识产权保护

知识产权也称为“智力成果权”、“智慧财产权”。它是人类通过创造性的智力劳动而获得的一项权利。根据我国民法通则的规定，知识产权是指民事权利主体（自然人、法人）基于创造性的智力成果。知识产权具有无形性、专有性、地域性和时间性四大特点。

计算机软件具有固定的表达形式，容易复制等特征，大多数国家将其列为版权法的保护范畴，也是知识产权保护中的一个重要方面，因此作为一个软件从业人员，一方面应该了解

法规，带头维护知识产权；另一方面也应学会利用知识产权维护自身的合法利益。

我国十分重视知识产权的保护，出台了一系列的相关法律法规。其中主要包括《著作权法》、《计算机软件保护条件》、《专利法》、《商标法》和《反不正当竞争法》。本章就针对这些主要的法律法规进行详细的介绍。

18.1 著作权法及实施条例

1990 年 9 月通过，1991 年 6 月 1 日正式实施的《中华人民共和国著作权法》是知识产权保护领域的最重要的法律基础。另外国家还颁发了《中华人民共和国著作权法实施条例》作为执行补充，该条例于 1991 年 5 月通过，2002 年 9 月修订。在这两部法律法规中，十分详细、明确地对著作权保护及具体实施作出大量明确的规定。

18.1.1 著作权法客体

著作权法及实施条件的客体是指受保护的作品。这里的作品，是指文学、艺术和自然科学、社会科学、工程技术领域内具有独创性并能以某种有形形式复制的智力成果。

1．作品类型其中包括以下 9 种类型。

文字作品：包括小说、诗词、散文、论文等以文字形式表现的作品。

口述作品：是指即兴的演说、授课、法庭辩论等以口头语言形式表现的作品。

音乐、戏剧、曲艺、舞蹈、杂技作品。

美术、摄影作品。

电影、电视、录像作品。

工程设计、产品设计图纸及其说明。

地图、示意图等图形作品。

计算机软件。

法律、行政法规规定的其他作品。

2．职务作品

为完成单位工作任务所创作的作品，称为职务作品。如果该职务作品是利用单位的物质技术条件进行创作，并由单位承担责任的；或者有合同约定，其著作权属于单位。那么作者将仅享有署名权，其他著作权归单位享有。

其他职务作品，著作权仍由作者享有，单位有权在业务范围内优先使用，并且在两年内，

未经单位同意，作者不能够许可其他人、单位使用该作品。

18.1.2 著作权法的主体

著作权法及实施条例的主体是指著作权关系人，通常包括著作权人、受让者两种。

1．著作权人与受让者

著作权人又称为原始著作权人，是根据创作的事实进行确定的，创作、开发者将依法取得著作权资格。

受让者又称为后继著作权人，是指没有参与创作，通过著作权转移活动成为享有著作权的人。

2．著作权人的确定

著作权法在认定著作权人时，是根据创作的事实进行的，而创作就是指直接产生文学、艺术和科学作品的智力活动。而为他人创作进行组织，提供咨询意见、物质条件或者进行其他辅助工作，不属于创作的范围，不被确认为著作权人。

如果在创作的过程中，有多人参与，那么该作品的著作权将由合作的作者共同享有。合作的作品是可以分割使用的，作者对各自创作的部分可以单独享有著作权，但不能够在侵犯合作作品整体的著作权的情况下行使。

如果遇到作者不明的情况，那么作品原件的所有人可以行使除署名权以外的著作权，直到作者身份明确。

希赛教育专家提示：如果作品是委托创作的，著作权的归属应通过委托人和受托人之间的合同来确定。如果没有明确的约定，或者没有签订相关合同，则著作权仍属于受托人。

18.1.3 著作权

根据著作权法及实施条例规定，著作权人对作品享有 5 种权利。

发表权：即决定作品是否公之于众的权利。

署名权：即表明作者身份，在作品上署名的权利。

修改权：即修改或者授权他人修改作品的权利。

保护作品完整权：即保护作品不受歪曲、篡改的权利。

使用权、使用许可权和获取报酬权、转让权：即以复制、表演、播放、展览、发行、摄制电影、电视、录像或者改编、翻译、注释、编辑等方式使用作品的权利；以及许可他人以上述

方式使用作品，并由此获得报酬的权利。

1．著作权保护期限根据著作权法相关规定，著作权的保护是有一定期限的。

① 著作权属于公民。署名权、修改权、保护作品完整权的保护期没有任何限制，永远属于保护范围。而发表权、使用权和获得报酬权的保护期为作者终生及其死亡后的 50 年（第 50 年的 12 月 31 日）。作者死亡后，著作权依照继承法进行转移。

② 著作权属于单位。发表权、使用权和获得报酬权的保护期为 50 年（首次发表后的第 50 年的 12 月 31 日），若 50 年内未发表的，不予保护。但单位变更、终止后，其著作权由承受其权利义务的单位享有。

2．使用许可当

第三方需要使用时，需得到著作权人的使用许可，双方应签订相应的合同。合同中应包括许可使用作品的方式，是否专有使用，许可的范围与时间期限，报酬标准与方法，违约责任。在合同未明确许可的权力，需再次经著作权人许可。合同的有效期限不超过 10 年，期满时可以续签。

对于出版者、表演者、录音录像制作者、广播电台、电视台而言，在下列情况下使用作品，可以不经著作权人许可、不向其支付报酬。但应指名作者姓名、作品名称，不得侵犯其他著作权。

为个人学习、研究或者欣赏，使用他人已经发表的作品。

为介绍、评论某一个作品或者说明某一个问题，在作品中适当引用他人已经发表的作品。

为报道时间新闻，在报纸、期刊、广播、电视节目或者新闻记录影片中引用已经发表的作品。

报纸、期刊、广播电台、电视台刊登或者播放其他报纸、期刊、广播电台、电视台已经发表的社论、评论员文章。

报纸、期刊、广播电台、电视台刊登或者播放在公众集会上发表的讲话，但作者声明不许刊登、播放的除外。

为学校课堂教学或者科学研究，翻译或者少量复制已经发表的作品，供教学或者科研人员使用，但不得出版发行。

国家机关为执行公务使用已经发表的作品。

图书馆、档案馆、纪念馆、博物馆、美术馆等为陈列或者保存版本的需要，复制本馆收藏的作品。

免费表演已经发表的作品。

对设置或者陈列在室外公共场所的艺术作品进行临摹、绘画、摄影、录像。

将已经发表的汉族文字作品翻译成少数民族文字在国内出版发行。

将已经发表的作品改成盲文出版。

18.2 计算机软件保护条例

1991 年 6 月通过，10 月 1 日正式实施的《计算机软件保护条例》是我国计算机软件保护的法律依据。该条例最新版本是在 2001 年底通过，2002 年 1 月 1 日正式实施的。

由于计算机软件也属于《中华人民共和国著作权法》保护的范围，因此在具体实施时，首先适用于《计算机软件保护条例》条文规定，在《计算机软件保护条例》中没有规定适用条文的情况下，才依据《著作权法》的原则和条文规定执行。

《计算机软件保护条例》的客体是计算机软件，而在此计算机软件是指计算机程序及其相关文档。根据条例规定，受保护的软件必须由开发者独立开发，并且已经固定在某种有形物体上（如光盘、硬盘、软盘）。

希赛教育专家提示：《计算机软件保护条例》对软件著作权的保护只是针对计算机软件和文档，并不包括开发软件所用的思想、处理过程、操作方法或数学概念等。

1．著作权人的确定

（1）合作开发。对于由两个以上开发者或组织合作开发的软件，著作权的归属根据合同约定来确定。若无合同，共享著作权。若合作开发的软件可以分割使用，那么开发者对自己开发的部分单独享有著作权，可以在不破坏整体著作权的基础上行使。

（2）职务开发。如果开发者在单位或组织中任职期间，所开发的软件符合以下条件，则软件著作权应归单位或组织所有：

针对本职工作中明确规定的开发目标所开发的软件。

开发出的软件属于从事本职工作活动的结果。

使用了单位或组织的资金、专用设备、未公开的信息等物质、技术条件，并由单位或组织承担责任的软件。

（3）委托开发。如果是接受他人委托而进行开发的软件，其著作权的归属应由委托人与受托人签订书面合同约定；如果没有签订合同，或合同中未规定的，其著作权由受托人享有。

另外，由国家机关下达任务开发的软件，著作权的归属由项目任务书或合同规定，若未明确规定，其著作权应归任务接受方所有。

2．软件著作权根据《计算机软件保护条例》规定，软件著作权人对其创作的软件产品，享有以下权利：发表权、署名权、修改权、复制权、发行权、出租权、信息网络传播权、翻译权、使用许可权、获得报酬权、转让权。软件著作权自软件开发完成之日起生效。

（1)著作权属于公民。著作权的保护期为作者终生及其死亡后的 50 年(第 50 年的 12 月 31 日)。对于合作开发的，则以最后死亡的作者为准。在作者死亡后，将根据继承法转移除了署名权之外的著作权。

（2）著作权属于单位。著作权的保护期为 50 年（首次发表后的第 50 年的 12 月 31 日)，

若 50 年内未发表的，不予保护。但单位变更、终止后，其著作权由承受其权利义务的单位享有。

（3）合法复制品所有人权利。当得到软件著作权人的许可，获得了合法的计算机软件复制品，则复制品的所有人享有以下权利：

根据使用的需求，将该计算机软件安装到设备中（电脑、PDA 等信息设备)。

可以制作复制品的备份，以防止复制品损坏，但这些复制品不得通过任何方式转给其他人使用。

根据实际的应用环境，对其进行功能、性能等方面的修改。但未经软件著作权人许可，不得向任何第三方提供修改后的软件。

（4）使用许可的特例。如果使用者只是为了学习、研究软件中包含的设计思想、原理，而以安装、显示、存储软件等方式使用软件，可以不经软件著作权人许可，不向其支付报酬。

（5)侵权责任。根据计算机软件保护条件，侵犯软件著作权的法律责任包括民事责任、刑事责任和行政责任 3 种。

希赛教育专家提示：如果是因为可供选用的表达方式有限，而造成与原来存在的软件相似，则不构成对原有软件著作权的侵犯。

18.3 商标法及实施条例

任何能够将自然人、法人及组织的商品与他人的商品区别开的可视性标志，就是可以用于注册的商标。商标可以包括文字、图形、字母、数字、三维标志和颜色组合。商标必须报商标局核准注册。通常包括商品商标、服务商标、集体商标，以及证明商标。

除了一些与国家、政府、国际组织相同、相似的，以及一些带有民族歧视、影响社会道

德等性质的标志不能够作为商标注册外，县级以上行政区划的地名也不能够作为商标。

1．商标的使用期限商标的使用，是指将商标用于商品、包装、容器、交易文书、广告宣传、展览，以及其商业活动中。

注册商标的有效期是 10 年，从核准通过、正式注册之日起开始计算。在有效期满之后，可以续注册，但必须在期满前 6 个月提出申请，如未在此期间提出申请的，则给予 6 个月的宽限期，在宽限期还未提出申请的，将注销其商标。

2．注册商标的申请

要申请商标注册，应当按照公布的商品和服务分类表按类申请。每一件商标注册申请应当向商标局提交《商标注册申请书》1 份、商标图样 5 份（清晰，长宽不大于 250px，不小于 125px）；指定颜色的，并应当提交着色图样 5 份、黑白稿 1 份。

如果商标是三维标志，应在申请书中声明，并提交能够确定三维形状的图样。商标为外文或者包括外文的，应说明其含义。

如果有多个申请人，在同一天申请注册相同或近似的商标，则申请人应该提交其申请注册前在先使用该商标的证据，先使用者获得商标注册。如果都没有使用证据，那么将通过协商解决，协商无效，则通过抽签决定。

3．注册商标专用权保护

注册商标的专用权，以核准注册的商标和核定使用的商品为限。而若存在以下行为之一，就属于侵犯注册商标专用权。

① 未经商标注册人的许可，使用相同或近似商标。

② 销售侵犯商标专用权的商品（注：如果销售方不知道是侵权商品，并且能够证明自己是合法取得的，不承担相应责任）。

③ 伪造他人注册商标，或销售这些伪造的注册商标。

④ 未经商标注册人同意，更换其注册商标，并将更换商标的商品投入市场。当出现侵犯注册商标的专用权时，双方当事人可以协商解决。如果无法协商解决，可以向人民法院起诉，或提请工商局处理。法院可以根据侵权行为的情节判处 50 万元以下的赔偿。

4．注册商标使用的管理

当合法地注册了商标使用权后，就可以在商品、商品包装、说明书或者其他附着物上标明“注册商标”或者注册标记（包括©和®）。

希赛教育专家提示：若商标注册人死亡或者终止，自死亡或终止之日起 1 年期满，而没有继续办理转移手续，任何人都可以向商标局申请注销该注册商标。

18.4 专利法及实施细则

专利法的客体是发明创造，也就是其保护的对象。这里的发明创造是指发明、实用新型和外观设计。发明是指对产品、方法或者其改进所提出的新的技术方案；实用新型是指对产品的形状、构造及其组合，提出的适于实用的新的技术方案；外观设计是指对产品的形状、图案及其组合，以及色彩与形状、图案的结合所作出的富有美感并适于工业应用的新设计。

1．授予专利权的条件要想申请专利权的发明和实用新型，应当具备新颖性、创造性和实用性等特点。

新颖性：也就是在申请专利之前没有同样的发明或实用新型在国内外出现过（不过如果是自己在政府主办或承认的展会上展出，在规定的学术会议或技术会议上发表，他人未经同意泄露等情况，并不丧失新颖性）。

创造性：是指同原有的技术相比，有突出的特点和显著的进步。

实用性：是指其能够被制造或者使用，并且有积极的效果。而对于想申请专利权的外观设计，应保证与在国内外发表的外观设计不相同、不相近似。

希赛教育专家提示：对于科学发现、智力活动的规则和方法、疾病的诊断和治疗方法、动植物品种及用原子核变换方法获得的物质，不能够被授予专利权。

2．确定专利权人

根据专利法的规定，专利权归属于发明人或者设计人，这是指对发明创造作出创造性贡献的人。对于在发明创造过程中，只负责组织、提供方便、从事辅助工作的都不属于发明人或设计人。

（1）职务发明。执行单位任务，或者利用本单位的物质技术条件所完成的发明创造，被视为职务发明创造，通常包括：

① 在本职工作中作出的发明创造。

② 在履行单位交付的本职工作之外的任务所做出的发明创造。

③ 退职、退休或者调动工作后 1 年内做的，与其原来承担的任务相关的发明创造。对于职务发明的专利申请被批准后，单位是专利权人。对于利用单位的物质技术条件进行发明创造的，发明人、设计人与单位之间可以签订合同，重新规定专利权的归属。

（2）合作发明、设计。对于合作发明、设计的，其专利权应属共同所有，但可以根据合作方之间另行签订的合同来确定专利权的归属。

（3）委托发明。一个单位或者个人接受其他单位或个人的委托，所完成的发明创造，

若没有签订合同规定专利权归属，则专利权归属发明、设计者。

如果是非职务发明，则单位无权压制个人进行专利权申请。对于多个相类似的专利申请，则专利权归属最先提交的申请人。

3．专利权

未经专利权人许可，实施专利的，就属于侵犯专利权，专利权人可以起诉，申请调解。

假冒他人专利，没收违法所得，并处 3 倍以下，或 5 万元以下的罚款，情节严重的，依法追究刑事责任。以非专利产品冒充专利产品，责令整改，并可处 5 万元以下的罚款。侵犯专利权的赔偿数额，参照该专利许可使用费的倍数合理确定。专利诉讼的有效期是 2 年，以专利权人得知侵权行为之日起计算。对于以下情况，不视为侵犯专利权：

① 对于专利权人制造、进口或者经专利权人许可而制造、进口的专利产品，或者依照专利方法直接获得的产品售出后，使用、许诺销售或者销售该产品。

② 在专利申请日前已经制造相同产品，使用相同方法或者已经作好制造、使用的必要准备，并且在原有范围内继续制造、使用。

③ 临时通过中国的国外运输工具，按其自身需要使用了专利。

④ 专为科学研究和实验而使用有关专利的。我国现行专利法规定的发明专利权保护期限为 20 年，实用新型和外观设计专利权的期限为 10 年，均从申请日开始计算。

在保护期内，专利权人应该按时缴纳年费。在专利权保护期限内，如果专利权人没有按规定缴纳年费，或以书面声明放弃其专利权的，专利权可以在期满前终止。

另外，任何单位和个人都可以在授予专利之日起，请求专利复审，如果复审未通过，则将终止专利权。

对于具备实施条件的单位，可以以合理的条件请求发明或者实用新型专利权人许可实施其专利。若国家出现紧急状态或者非常情况时，可以为了公共利益强制实施发明专利、实用新型专利的许可。

18.5 反不正当竞争法

不正当竞争是指经营者违反规定，损害其他经营者的合法权益，扰乱社会经济秩序的行为。

（1）采用不正当的市场交易手段：例如假冒他人注册商标；擅自使用与知名商品相同或相近的名称、包装，混淆消费者；擅自使用他人的企业名称；在商品上伪造认证标志、名

优标志、产地等信息，从而达到损害其他经营者的目的。

（2）利用垄断的地位，来排挤其他经营者的公平竞争。

（3）利用政府职权，限定商品购买，以及对商品实施地方保护主义。

（4）利用财务或其他手段进行贿赂，以达到销售商品的目的。

（5）利用广告或者其他方法，对商品的质量、成分、性能、用途、生产者、有效期、产地等进行误导性的虚假宣传。

（6）以低于成本价进行销售，排挤竞争对手。不过对于鲜活商品、有效期将至及积压产品的处理，以及季节性降价，清债、转产、歇业等原因进行降价销售均不属于不正当竞争。

（7）搭售违背购买者意愿的商品。

（8）采用不正当的有奖销售。例如谎称有奖，却是内定人员中奖；利用有奖销售推销次价高产品；奖金超过 5 000 元的抽奖式有奖销售。

（9）捏造、散布虚伪事实，损害对手商誉。

（10）串通投标，排挤对手。

采用不正当竞争对别的经营者造成损害的，应承担赔偿责任。无法计算损失的，则赔偿侵权期因侵权所得的利润。

（1）对于假冒注册商标、姓名、认证、产地的不正当竞争行为根据《商标法》进行处罚；仿冒知名商标的，则可以根据情节处违法所得的 1 万～3 万元罚款，特别严重的追究刑事责任。

（2）通过贿赂达到销售目的，根据情节处以 1 万～20 万元罚款，严重的追究刑事责任。

（3）利用独占地位进行经营，根据情节处以 5 万～20 万元罚款；借此销售质次价高商品的，则没收违法所得，并罚款 1 万～3 万元。

（4）采用广告误导消费者，处以 1 万～20 万元罚款。

（5）采用不合法的有奖销售的，根据情节处以 1 万～10 万元的罚款。

（6）串通投标者，根据情节处以 1 万～20 万元的罚款。

在《反不正当竞争法》中，对商业秘密进行了保护。商业秘密是指不为公众所知，具有经济利益，具有实用性，并且已经采取了保密措施的技术信息与经营信息，如果存在以下行为的，视为侵犯商业秘密：

（1）以盗窃、利诱、胁迫等不正当手段获取别人的商业秘密。

（2）披露、使用以不正当手段获取的商业秘密。

（3）违反有关保守商业秘密的要求约定，披露、使用其掌握的商业秘密。

对于侵犯商业秘密的，将根据情节处以 1 万～20 万元罚款。

第 19 章：标准化知识

标准化是人类由自然人进入社会共同生活实践的必然产物，它随着生产的发展、科技的进步和生活质量的提高而发生、发展，受生产力发展的制约，同时又为生产力的进一步发展创造条件。

本章重点要求读者掌握标准化概论、标准分级与标准类型、软件开发规范与文档标准 3 个方面的知识。

19.1 标准化概论

标准化是一门综合性学科，其工作内容极为广泛，可渗透到各个领域。标准化工作的特征包括横向综合性、政策性和统一性。

为在一定的范围内获得最佳秩序，对活动或其结果规定共同的和重复使用的规则或特性的文件，称为标准。该文件经协商一致制定并经一个公认机构的批准。标准应以科学、技术和经验的综合成果为基础，以促进最佳社会效益为目的。标准化是指在经济、技术、科学及管理等社会实践中，对重复性事物和概念通过制定、发布和实施标准达到统一，以获得最佳秩序和最大社会效益。

根据《中华人民共和国标准化法》，“标准化工作的任务是制定标准、组织实施标准和对标准的实施进行监督”。“通过制定、发布和实施标准，达到统一”是标准化的实质。“获得最佳秩序和社会效益”则是标准化的目的。

19.2 标准分级与标准类型

根据制定机构和适用范围的不同，标准可分为若干个级别。按类型划分，标准可分为强制性标准和推荐性标准。

19.2.1 标准分级

根据《中华人民共和国标准化法》，国内标准分为国家标准、行业标准、地方标准和企

业标准。在全球范围内，标准的分级方式并不统一，一般可分为国际标准、行业标准、区域标准和企业标准等。

1．国际标准

国际标准是指由国际联合机构制定和公布，提供各国参考的标准。国际标准化组织、国际电工委员会和国际电信联盟制定的标准均为国际标准。此外，被 ISO 认可、收入 KWIC 索引中的其他 25 个国际组织制定的标准，也视为国际标准。

2．国家标准

国家标准是指由政府或国家级的机构制定或批准，适用于全国范围的标准，例如：GB（或 GB/T）：中华人民共和国国家标准。《中华人民共和国标准化法》规定，“国家标准由国务院标准化行政主管部门制定”。目前，国家标准由国家标准化管理委员会制定，国家质量监督检验检疫总局批准和公布。

ANSI（American NationalStandards Institute）：美国国家标准协会标准。

FIPS-NBS（Federal InformationProcessing Standards，National Bureau of Standards）：美国国家标准局联邦信息处理标准。

BS（British Standard）：英国国家标准。 JIS（Japanese Industrial Standard）：日本工业标准。

3．行业标准

行业标准是指由行业机构、学术团体或国防机构制定，并适用于某个业务领域的标准，举例如下。

IEEE：电气电子工程师学会标准。

GJB：中华人民共和国国家军用标准，由国防科学技术工业委员会批准，适合于国防部门和军队。

DOD-STD（Department OfDefense STanDards）：美国国防部标准，适用于美国国防部门。

MIL-S（MILitary Standards）：美国军用标准，适用于美国军队内部。

SJ：中国电子行业标准。

根据《中华人民共和国标准化法》，国内的行业标准由国务院有关行政主管部门制定，并报国务院标准化行政主管部门备案。

4．区域/地方标准

区域标准是指由区域性国际联合机构制定和公布，提供区域内各国参考和执行的标准，举例如下。

ARS：非洲地区标准，由非洲地区标准化组织制定。

EN：欧洲标准，由欧洲标准化委员会制定。

PAS：泛美标准，由泛美技术标准委员会制定。

在国内，地方标准是指由地方行政主管部门制定，仅适用于本地的标准，其标准代号一般以 DB 开头。根据《中华人民共和国标准化法》，地方标准由省、自治区、直辖市标准化行政主管部门制定，并报国务院标准化行政主管部门和国务院有关行政主管部门备案。

5．企业标准

企业标准是指一些大型企业或机构，由于工作需要制定的适用于本企业或机构的标准。

根据《中华人民共和国标准化法》，国内企业的产品标准须报当地政府标准化行政主管部门和有关行政主管部门备案。《GB/T 1 标准化工作导则》规定，企业标准以 Q 字开头。

目前，国外个别大型 IT 企业自定的某些标准已成为事实上的全球工业标准。

6．各级标准之间的关系

《中华人民共和国标准化法》明确规定了国家标准、行业标准、地方标准和企业标准之间的关系。

（1）对需要在全国范围内统一的技术要求，应当制定国家标准。

（2）对没有国家标准而又需要在全国某个行业范围内统一的技术要求，可以制定行业标准。在公布国家标准之后，该项行业标 准即行废止。

（3）对没有国家标准和行业标准而又需要在省、自治区、直辖市范围内统一的工业产品的安全、卫生要求，可以制定地方标准。在公布国家标准或者行业标准之后，该项地方标准即行废止。

（4）企业生产的产品没有国家标准和行业标准的，应当制定企业标准，作为组织生产的依据。已有国家标准或者行业标准的，国家鼓励企业制定严于国家标准或者行业标准的企业标准，在企业内部适用。

《中华人民共和国标准化法》同时规定，“国家鼓励积极采用国际标准”，但并没有明确指出：当国家标准与国际标准不一致时，应当采用哪个标准。按照国际惯例，当一国产品在另一国销售时，应当优先适用销售地的国家标准。

19.2.2 强制性标准与推荐性标准

《中华人民共和国标准化法》规定：国家标准、行业标准分为强制性标准和推荐性标准。

保障人体健康，人身、财产安全的标准和法律、行政法规规定强制执行的标准是强制性标准，其他标准是推荐性标准。省、自治区、直辖市标准化行政主管部门制定的工业产品的安全、卫生要求的地方标准，在本行政区域内是强制性标准。

强制性国家标准以 GB 开头，推荐性国家标准以 GB/T 开头。但应注意，此项规定并不适用于国家早期发布的标准，当时的国家标准统一以 GB 开头。

强制性标准可分为全文强制和条文强制两种形式。

（1）标准的全部技术内容需要强制时，为全文强制形式。此类标准，必须在“前言” 的第一段以黑体字写明：“本标准的全部技术内容为强制性。”

（2）标准中部分技术内容需要强制时，为条文强制形式。此类标准，应根据具体情况选用最简洁的方式，在标准“前言”的第一段以黑体字写明其强制性条文和非强制性条文。

根据国家质量技术监督局[2000]36 号文件，强制性内容的范围包括：

（1）有关国家安全的技术要求。

（2）保障人体健康和人身、财产安全的要求。

（3）产品及产品生产、储运和使用中的安全、卫生、环境保护、电磁兼容等技术要求。

（4）工程建设的质量、安全、卫生、环境保护要求及国家需要控制的工程建设的其他要求。

（5）污染物排放限值和环境质量要求。

（6）保护动植物生命安全和健康的要求。

（7）防止欺骗、保护消费者利益的要求。

（8）国家需要控制的重要产品的技术要求。

《中华人民共和国标准化法》规定：强制性标准，必须执行。不符合强制性标准的产品，禁止生产、销售和进口。生产、销售、进口不符合强制性标准的产品的，由法律、行政法规规定的行政主管部门依法处理，法律、行政法规未作规定的，由工商行政管理部门没收产品和违法所得，并处罚款；造成严重后果构成犯罪的，对直接责任人员依法追究刑事责任。

推荐性标准，国家鼓励企业自愿采用。这类标准，不具有强制性，任何单位均有权决定是否采用，主要通过经济手段或市场因素自行调节。违犯这类标准，不构成经济或法律方面的责任。但应当指出，推荐性标准一经接受并采用，或各方商定同意纳入经济合同中，就成为各方必须共同遵守的技术依据，具有法律上的约束性。

新中国成立以来，我国最初研制发布的强制性标准数量较多。上世纪 90 年代后，为了适应国内经贸发展，并与国际标准化接轨，国家标准主管部门曾多次对强制性标准的有关规

定进行调整，并对已有强制性标准进行反复的清理整顿，使强制性标准的数量得到适当控制。

第 20 章：应用数学

数学是一种严谨、缜密的科学，学习应用数学知识，可以培养系统架构设计师的抽象思维能力和逻辑推理能力，在从事系统分析工作时思路清晰，在复杂、紊乱的现象中把握住事物的本质，根据已知和未知事物之间的联系推断事物发展趋势和可能的结果。

应用数学虽然涉及的内容很多，但经常考查的知识点却往往集中于运筹方法与数据建模，所以本章将着重介绍这两个方面的内容。

20.1 运筹方法

运筹学是近代应用数学的一个分支，主要是将生产、管理等事件中出现的一些带有普遍性的运筹问题加以提炼，然后利用数学方法进行解决。前者提供模型，后者提供理论和方法。运筹学可以根据问题的要求，通过数学上的分析、运算，得出各种各样的结果，最后提出综合性的合理安排，以达到最好的效果。

运筹学作为一门用来解决实际问题的学科，在处理千差万别的各种问题时，一般有以下几个步骤：确定目标、制订方案、建立模型、制订解法。

20.1.1 网络计划技术

用网络分析的方法编制的计划称为网络计划，它是一种编制大型工程项目进度计划的有效方法。计划借助于网络表示各项工作与所需要的时间，以及各项工作的相互关系。通过网络分析研究工程费用与工期的关系，并找出在编制计划及计划执行过程中的关键路径，这种方法称为关键路径法（Critical Path Method，CPM）。

1．关键路径

在现代管理中，人们常用有向图来描述和分析一项工程的计划和实施过程，一项工程常被分为多个小的子工程，这些子工程称为活动。在有向图中，若以顶点表示活动，弧表示活动之间的先后关系，这样的图简称为 AOV（Activity On Vertex）网；若以顶点表示事件，弧表示活动，权表示完成该活动所需的时间（称为活动历时或持续时间），这样的图称为 AOE（Activity On Edge）网。例如，图 20-1 表示一个具有 10 个活动的某个工程的 AOE 网。

图中有 7 个顶点，分别表示事件 1～7，其中 1 表示工程开始状态，7 表示工程结束状态。

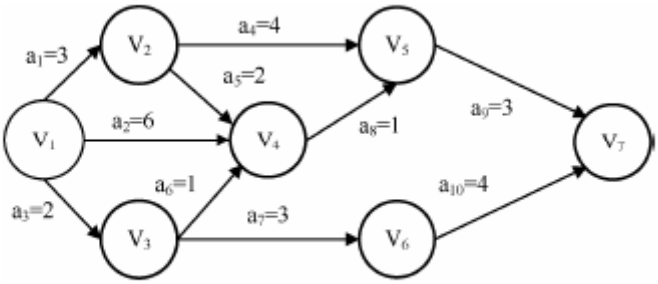


图 20-1　AOE 网络的例子

因 AOE 网中的某些活动可以并行地进行，所以完成工程的最少时间是从开始顶点到结束顶点的最长路径长度，称从开始顶点到结束顶点的最长路径为关键路径（临界路径），关键路径上的活动为关键活动。

为了找出给定的 AOE 网络的关键活动，从而找出关键路径，先定义几个重要的量：

（1）Ve(j)、Vl(j)：顶点 j 事件最早、最迟发生时间。

（2）e(i)、l(i)　　：活动 i 最早、最迟开始时间。

从源点 V1 到某顶点 Vj 的最长路径长度，称为事件 Vj 的最早发生时间，记作 Ve(j)。Ve(j) 也是以 Vj 为起点的出边<Vj,Vk>所表示的活动 ai 的最早开始时间 e(i)。在不推迟整个工程完成的前提下，一个事件 Vj 允许的最迟发生时间，记作 Vl(j)。显然，l(i)=Vl(j)−(ai 所需时间)，其中 j 为 ai 活动的终点。满足条件 l(i)=e(i)的活动为关键活动，关键活动所组成的路径称为关键路径。

求顶点 Vj 的 Ve(j)和 Vl(j)可按以下两步来做：

（1）由源点开始向汇点递推

$$\begin{cases} V_e(1)=0 \\ V_e(j)=Max\{V_e(i)+d(i,j)\}, < V_i,\ V_j > \in E_1,\ 2 \leqslant j \leqslant n \end{cases}$$

其中，E1 是网络中以 Vj 为终点的入边集合。

（2）由终点（汇点）开始向源点递推

$$\begin{cases} V_l(n)=V_e(n) \\ V_l(j)=Min\{V_l(k)-d(j,k)\}, < V_j,\ V_k > \in E_2,\ 2 \leqslant j \leqslant n-1 \end{cases}$$

其中，E2 是网络中以 Vj 为起点的出边集合。

要求一个 AOE 网的关键路径，一般需要根据以上变量列出一张表格，逐个检查。例如，求图 20-1 所示的 AOE 网的关键路径的表格如表 20-1 所示。

表 20-1　求关键路径的过程

V_j	$Ve(j)$	$Vl(j)$	a_i	$e(i)$	$l(i)$	$l(i)-e(i)$
V_1	0	0	$a_1(3)$	0	0	0
V_2	3	3	$a_2(6)$	0	0	0
V_3	2	3	$a_3(2)$	0	1	1
V_4	6	6	$a_4(4)$	3	3	0
V_5	7	7	$a_5(2)$	3	4	1
V_6	5	6	$a_6(1)$	2	5	3
V_7	10	10	$a_7(3)$	2	3	1
			$a_8(1)$	6	6	0
			$a_9(3)$	7	7	0
			$a_{10}(4)$	5	6	1

因此，图 20-1 的关键活动为 a1，a2，a4，a8 和 a9，其对应的关键路径有两条，分别为 V1→V2→V5→V7 和 V1→V4→V5→V7，长度都是 10。

2．网络优化

在得到了关键路径后，就相当于得到了项目的计算工期，得到了一个初始的计划方案。但通常还要对初始方案进行调整和完善。根据计划的要求，综合考虑进度、资源、费用等目标，即进行网络优化，确定最优的计划方案。

（1）时间优化。根据对计划进度的要求，缩短工程完成时间。既可以采取技术措施，缩短工程完工时间，也可以采取组织措施，充分利用非关键活动的总时差（最迟开始时间-最早开始时间），合理调配技术力量及人、财、物等资源，缩短关键活动的持续时间。还可以通过改变工作之间的逻辑关系，采用并行的方式来缩短工期。

（2）时间-资源优化。在编制网络计划、安排工程进度的同时，就要考虑尽量合理地利用现有资源，并缩短工程周期。但是，由于一项工程所包含的活动繁多，涉及的资源利用情况比较复杂，往往不可能在编制网络计划时，一次性把进度和资源利用都能够作出统筹合理的安排，而是需要进行几次综合平衡之后，才能得到在时间进度及资源利用等方面都比较合理的计划方案。具体的要求和做法是：优先安排关键活动所需要的资源；利用非关键活动的总时差，错开各活动的开始时间，拉平资源需要量的高峰；在确实受到资源限制，或者在考虑综合经济效益的条件下，也可以适当地推迟工程完工时间。

（3）时间-费用优化。在编制网络计划过程中，研究如何使得工程完工时间短、费用少；或者在保证既定工程完工时间的条件下，所需要的费用最少；或者在限制费用的条件下，工程完工时间最短。这就是时间-费用优化所要研究和解决的问题。为完成一项工程，所需要的费用可分为直接费用和间接费用。同时，项目有不可压缩的最短时间，也称为极限时间，它是指为了缩短各活动的持续时间而采取一切可能的技术和组织措施后，可能达到的最短的

工作时间和完成项目的最短时间。在进行时间-费用优化时，需要计算活动的直接费用变动率（简称为直接费用率）：直接费用率 = (极限时间的活动直接费用-正常时间的活动直接费用)/(正常时间-极限时间)

3．综合实例

下面通过一个综合实例，帮助读者理解本节的知识。

假设某信息系统开发工程合同工期为 25 个月，承建单位编制的网络计划图如图 20-2 所示。

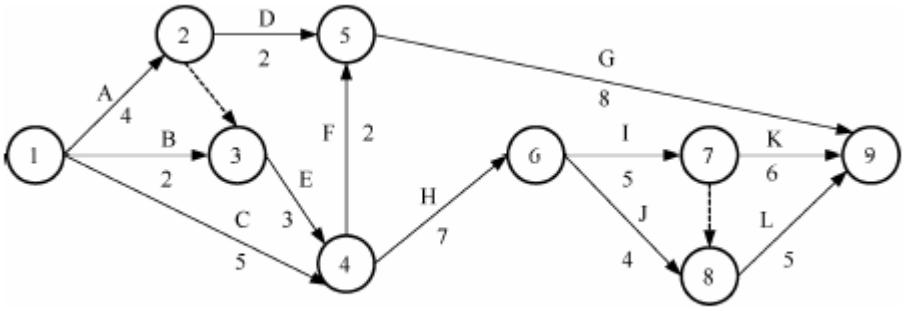


图 20-2　某工程网络计划图

（1）该网络计划能否满足合同工期要求？为确保工程按期完工，哪些工作应作为重点对象？

（2）当该计划执行 7 个月后，经监理工程师检查发现，工作 C 和 D 已完成，而 E 将拖后 2 个月。当计划执行到第 7 个月后，工作 E 的实际进度是否影响总工期？如果实际进度确定影响到总工期，为保证总工期不延长对原进度计划的调整方案有哪些？

（3）如果承建单位提出采用压缩某些工作持续时间，对原计划进行调整以保证工期不延长，各工作的直接费用率与极限时间如表 20-2 所示。在不改变各工作逻辑关系的前提下，原进度计划的最优调整方案是什么？此时直接费用将增加多少万元？

表 20-2　直接费用率与极限时间

工程过程	F	G	H	I	J	K	L
直接费用率	—	10.0	6.0	4.5	3.5	4.0	4.5
极限时间（月）	2	6	5	3	1	4	3

【解】一般利用所给出的图形找出关键路径和计算工期，从而确定重点工作。

（1）在图 20-2 中，有 2 条虚线弧，它表示虚活动，即不需要任何资源（时间、费用等），只表示逻辑关系的活动。从图 20-2 中可以看出，关键路径为 A→E→H→I→K，长度为 25，也就是说，项目的计算工期为 25 个月，正好等于合同工期，因此，该网络计划能满足合同工期要求。为了确保工程按期完工，A、E、H、I、K 工作应作为重点控制对象，因

为它们为关键工作。

（2）分析拖延工作是否在关键路径上，拖延的时间是否超过工作的总时差来衡量与判断是否影响工期。因为工作 E 为关键工作，其总时差为 0。所以，E 拖延 2 个月将影响总工期 2 个月。由于工作 E 拖延了 2 个月，使总工期延长了 2 个月，为了保证总工期不延长，对原计划的调整方法有 2 种，一是改变某些工作之间的逻辑关系，二是缩短某些工作的持续
时间。

（3）要调整计划，使之不延长时间，则需要调整关键路径上的工作，即 A、E、H、I、K。但题目已经告诉我们，是从第 7 个月开始，这时 A 已经完成了。因此，只能选择 E、H、I、K。从表 20-2 中可以看出，E 是不可以压缩的。所以，只能压缩 H、I、K。

再看表 20-2，压缩直接费用率最小的为工作 K，K 原计划时间为 6 个月，极限时间为 4 个月，可以压缩 2 个月，正好可以满足要求。但是要注意，如果 K 压缩 2 个月，则会引起关键路径的变化。图 20-3 是 K 压缩 1 个月后的网络计划图。

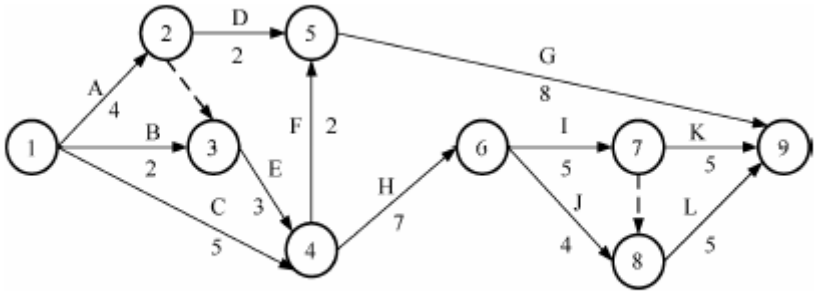


图 20-3　K 压缩 1 个月后的网络计划图

从图 20-3 中可以看出，这时关键路径有 2 条，分别是 A→E→H→I→K 和 A→E→H→I→L。因此，需要把 I 压缩 1 个月，费用为 4.5 万元。

综上所述，应该压缩 K、I 各 1 个月，增加费用为 4.0+4.5=8.5（万元）。

20.1.2 线性规划

线性规划是研究在有限的资源条件下，如何有效地使用这些资源达到预定目标的数学方法。用数学的语言来说，也就是在一组约束条件下寻找目标函数的极值问题。

求极大值（或极小值）的模型表达如下：

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leqslant b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leqslant b_2 \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leqslant b_n \end{cases}, \quad x_i \geqslant 0, \quad 1 \leqslant i \leqslant n$$

在上述条件下，求解 x1,x2,- - - ,xn,使满足下列表达式的 z 取极大值（或极小值）：

z=c1x1+c2x2+ +cnxn

1．图解法解线性规划问题的方法有很多，最常用的有图解法和单纯形法。图解法简单直观，有助于了解线性规划问题求解的基本原理，下面，通过一个例子来说明图解法的应用。

某工厂在计划期内要安排生产 I、II 两种产品，已知生产单位产品所需的设备台时及 A、B 两种原料的消耗，如表 20-3 所示。

表 20-3 产品及原料表

	I	II	总数
设 备	1	2	8 台时
原材料 A	4	0	16kg
原材料 B	0	4	12kg

该工厂每生产一件产品 I 可获利 2 元，每生产一件产品 II 可获利 3 元，问应该如何安排计划使该工厂获利最多？

【解】该问题可用以下数学模型来描述，设 x1，x2 分别表示在计划期内产品 I、II 的产量，因为设备的有效台时是 8，这是一个限制产量的条件，所以在确定产品 I、II 的产量时，要考虑不超过设备的有效台时数，即可用不等式表示为：

x1 +2x2≤8

同理，因原料 A、B 的限量，可以得到以下不等式：

4 x1 ≤16

4 x2 ≤12

该工厂的目标是在不超过所有资源限制的条件下，如何确定产量 x1，x2 ，以得到最大的利润。若用 z 表示利润，这时 z =2x1 +3x2 。综上所述，该计划问题可用数学模型表示为：目标函数

maxz = 2x1 + 3x2

满足约束条件

x1 +2x2 ≤8

4x1 ≤16

4x2 ≤12

x1、x2≥ 0

在以 x1，x2 为坐标轴的直角坐标系中，非负条件 x1， x2 ≥0 是指第一象限。上述每个约束条件都代表一个半平面。例如，约束条件 x1 + 2x2 ≤8 代表以直线 x1 + 2x2 ≤ 8 为边界的左下方的半平面。若同时满足 x1， x2 ≥0， x1 + 2x2 ≤8， 4x1 ≤16 和 4x2 ≤ 12 的约束条件的点，必然落在由这三个半平面相交组成的区域内，如图 20-4 中的阴影部分所示。阴影区域中的每一个点（包括边界点）都是这个线性规划问题的解（称可行解），因而此区域是本题的线性规划问题的解的集合，称它为可行域。

再分析目标函数 z = 2x1 +3x2 ，在坐标平面上，它可表示以 z 为参数，-2/3 为斜率的一族平行线：

$$x_2 = -(\frac{2}{3})x_1 + \frac{z}{3}$$

位于同一直线上的点，具有相同的目标函数值，因此称为等值线。当 z 值由小变大时，直线沿其法线方向向右上方移动。当移动到 Q2 点时，使 z 值在可行域边界上实现最大化，这就得到了本题的最优解 Q2，Q2 点的坐标为（4，2）。经过计算，可以得出 z=14。

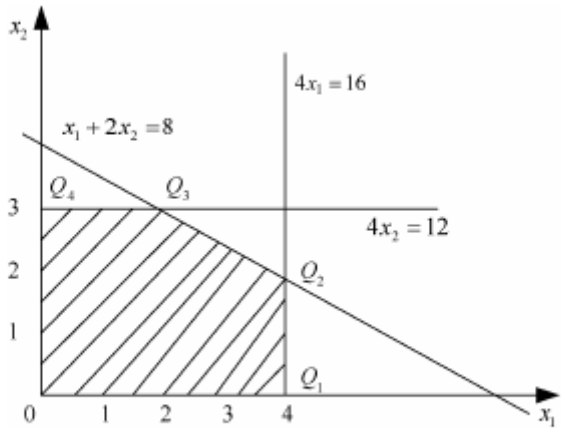


图 20-4 图解法

这说明该厂的最优生产计划方案是：生产 4 件产品 I，2 件产品 II，可得最大利润为 14 元。

2．关于解的讨论

在上述例题中，得到的最优解是唯一的，但对一般线性规划问题而言，求解结果还可能出现以下几种情况：无穷多最优解（多重解），无界解（无最优解），无可行解。当求解

结果出现后两种情况时，一般说明线性规划问题的数学模型有错误。无界解源于缺乏必要的约束条件，无可行解源于矛盾的约束条件。

从图解法中直观地看到，当线性规划问题的可行域非空时，它是有界或无界凸多边形。若线性规划问题存在最优解，它一定在可行域的某个顶点得到；若在两个顶点同时得到最优解，则它们连线上的任意一点都是最优解，即有无穷多最优解。

3．单纯形法

图解法虽然直观，但当变量数多于三个以上时，它就无能为力了，这时需要使用单纯形法。

单纯形法的基本思路是：根据问题的标准，从可行域中某个可行解（一个顶点）开始，转换到另一个可行解（顶点），并且使目标函数达到最大值时，问题就得到了最优解。限于篇幅，本书不再介绍单纯形法的详细求解过程。

4．线性规划的适用性

线性规划模型用在原材料单一、生产过程稳定不变、分解型生产类型的企业是十分有效的，例如，石油化工厂等。对于产品结构简单，工艺路线短，或者零件加工企业，有较大的应用价值。需要注意的是，对于机电类企业用线性规划模型只适用于作年度的总生产计划，而不宜用来作月度计划。这主要与工件在设备上的排序有关，计划期太短，很难安排过来。

一般来说，一个经济管理问题符合以下条件时，才能建立线性规划的模型：

（1）要求解问题的目标函数能用数值指标来反映，且为线性函数。

（2）存在着多种方案。

（3）要求达到的目标是在一定约束条件下实现的，这些约束条件可用线性等式或不等式描述。

20.1.3 决策论

决策就是决定的意思，大至国家经济、政治，小到个人生活，凡是在有选择的地方就有决策。关于决策的重要性，诺贝尔奖金获得者西蒙有一句名言“管理就是决策”。这就是说，管理的核心是决策。

1．决策过程和模型构造决策行为的模型主要有两种，分别为面向结果的方法和面向过程的方法。面向决策结果的方法程序比较简单，其过程为“确定目标→收集信息→提出方案→方案选择→决策”。面向决策过程的方法一般包括“预决策→决策→决策后”三个阶段，

其中决策阶段又可分为分部决策和最终决策两个子阶段。

任何决策问题都由以下要素构成决策模型：

（1）决策者。可以是个人、委员会或某个组织，一般指领导者或领导集体。

（2）可供选择的方案（替代方案）、行动或策略。

（3）衡量选择方案的准则。包括目的、目标、属性、正确性的标准，在决策时有单一准则和多准则。

（4）事件：不为决策者所控制的、客观存在的、将发生的状态。

（5）每一事件的发生将会产生的某种结果。例如，获得收益或损失。

（6）决策者的价值观。例如，决策者对货币额或不同风险程度的主观价值观念。

2．不确定型决策

不确定型决策（非确定型决策）是指决策者对环境情况一无所知，决策者根据自己的主观倾向进行决策。根据决策者的主观态度不同，可分为 5 种准则，分别为悲观主义准则、乐观主义准则、折中主义准则、等可能准则和后悔值准则。下面通过一个例题，具体介绍这些准则的含义和求解方法。

某公司需要根据下一年度宏观经济的增长趋势预测决定投资策略。宏观经济增长趋势有不景气、不变和景气 3 种，投资策略有积极、稳健和保守 3 种，各种状态的收益如表 20-4 所示。

表 20-4　各种状态的收益

预计收益（单位：百万元人民币）		经济趋势预测		
		不景气	不变	景气
投资策略	积极	50	150	500
	稳健	150	200	300
	保守	400	250	200

【解】在本题中，由于下一年度宏观经济的各种增长趋势的概率是未知的，所以是一个不确定型决策问题。

（1）乐观主义准则。乐观主义准则也称为最大最大准则（maxmax 准则），其决策的原则是“大中取大”。持这种准则思想的决策者对事物总抱有乐观和冒险的态度，他决不放弃任何获得最好结果的机会，争取以“好中之好”的态度来选择决策方案。决策者在决策表中各个方案对各个状态的结果中选出最大者，记在表的最右列，再从该列中选出最大者。在表 20-4 中，积极方案的最大结果为 500，稳健方案的最大结果为 300，保守方案的最大结果为 400。三者的最大值为 500，因此，选择其对应的积极投资方案。

（2）悲观主义准则。悲观主义准则也称为最大最小准则（maxmin 准则），其决策的原则是“小中取大”。这种决策方法的思想是对事物抱有悲观和保守的态度，在各种最坏的可能结果中选择最好的。决策时从决策表中各方案对各个状态的结果中选出最小者，记在表的最右列，再从该列中选出最大者。在表 20-4 中，积极方案的最小结果为 50，稳健方案的最小结果为 150，保守方案的最小结果为 200。三者的最大值为 200，因此，选择其对应的保守投资方案。

（3）折中主义准则。折中主义准则也称为赫尔威斯（Harwicz）准则，这种决策方法的特点是对事物既不乐观冒险，也不悲观保守，而是从中折中平衡一下，用一个系数 α（称为折中系数）来表示，并规定 0≤α≤1，用以下公式计算结果：

$cvi=\alpha\times\max\{aij\}+(1-\alpha)\times\min\{aij\}$

即用每个决策方案在各个自然状态下的最大效益值乘以 α，再加上最小效益值乘以 1−α。然后再比较 cvi，从中选择最大者。显然，折中主义准则的结果取决于 α 的选择。α 接近于 1，则偏向于乐观；α 接近于 0，则偏向于悲观。

（4）等可能准则。等可能准则也称为拉普拉斯（Laplace）准则。当决策者无法事先确定每个自然状态出现的概率时，就可以把每个状态出现的概率定为 1/n（n 是自然状态数），然后按照最大期望值准则决策。也就是说，把一个不确定型决策转换为风险决策。

（5）后悔值准则。后悔值（遗憾值）准则也称为萨维奇（Savage）准则、最小机会损失准则。决策者在制定决策之后，如果不能符合理想情况，必然有后悔的感觉。这种方法的特点是每个自然状态的最大收益值（损失矩阵取为最小值），作为该自然状态的理想目标，并将该状态的其他值与最大值相减所得的差作为未达到理想目标的后悔值。这样，从收益矩阵就可以计算出后悔值矩阵。最后按照最大后悔值达到最小的方法进行决策，因此，也称为最小最大后悔值（minmax）。在本题中，根据表 20-4 可以得出后悔值矩阵。如表 20-5 所示为各种状态的后悔值。

表 20-5　各种状态的后悔值

预计收益（单位：百万元人民币）		经济趋势预测		
		不景气	不变	景气
投资策略	积极	350	100	0
	稳健	250	50	200
	保守	0	0	300

在表 20-5 中，积极方案的最大后悔值为 350，稳健方案的最大后悔值为 250，保守方案的最大后悔值为 300。三者的最小值为 250，因此，选择其对应的稳健投资方案。

4．风险决策

风险决策是指决策者对客观情况不甚了解，但对将发生各事件的概率是已知的。在风险决策中，一般采用期望值作为决策准则，常用的有最大期望收益决策准则（Expected Monetary Value，EMV）和最小机会损失决策准则（Expected Opportunity Loss，EOL）。

（1）最大期望收益决策准则。决策矩阵的各元素代表“策略-事件”对的收益值，各事件发生的概率为 pj ，先计算各策略的期望收利益值 Σpjaij'，i=1,2, ,n，然后从这些期望收益值中选取最大者，以它对应的策略为决策者应选择的决策策略。

（2）最小机会损失决策准则。决策矩阵的各元素代表“策略-事件”对的损失值，各事件发生的概率为 pj ，先计算各策略的期望损失值 1.png 然后从这些期望收益值中选取最小者，以它对应的策略为决策者应选择的决策策略。

当 EMV 为最大时，EOL 便为最小。因此，在决策时用这两个决策准则所得的结果是一致的。

某电子商务公司要从 A 地向 B 地的用户发送一批价值为 90 000 元的货物。从 A 地到 B 地有水、陆两条路线。走陆路时比较安全，其运输成本为 10 000 元；走水路时一般情况下的运输成本只要 7000 元，不过一旦遇到暴风雨天气，则会造成相当于这批货物总价值的 10%的损失。根据历年情况，这期间出现暴风雨天气的概率为 1/4，那么该电子商务公司该如何选择呢？

【解】这是一个风险决策问题，其决策树如图 20-5 所示。

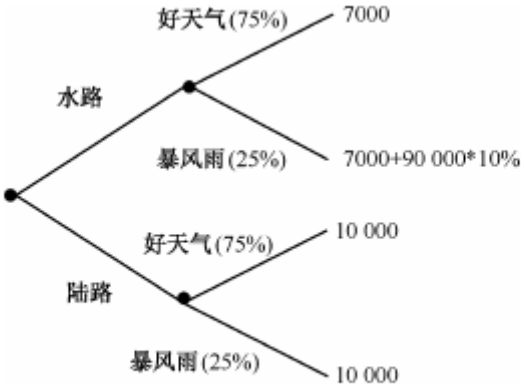


图 20-5 运输问题的决策树

根据图 20-5，走水路时，成本为 7000 元的概率为 75%，成本为 16 000 元的概率为 25%，因此，走水路的期望成本为(7 000×75%)+(16 000×25%) =9250（元）；走陆路时，其成本为(10 000×75%)+(10 000×25%)=10 000（元）。所以，走水路的期望成本小于走陆路的成本，应该选择走水路。

20.1.4 对策论

对策论也称为竞赛论或博弈论，是研究具有竞争（或斗争）性质现象的数学理论和方法。大到国际间的谈判、各种政治力量的较量，小到日常生活中的“诡计”，都是对策论的研究对象。

具有竞争或对抗性质的行为称为对策行为。在这类行为中，参加竞争的各方各自具有不同的目标和利益。为了达到各自的目标和利益，各方必须考虑对手的各种可能的行动方案，并力图选取对自己最为有利或最为合理的方案。对策论就是研究对策行为中竞争各方是否存在最合理的行动方案，以及如何找到这个合理的行动方案的数学理论和方法。在此我们不必研究高深的理论，仅以一个实例来说明这类考题如何解决，因为这类试题往往是 “一通百通”的。

例：甲、乙两个独立的网站主要靠广告收入来支撑发展，目前都采用较高的价格销售广告。这两个网站都想通过降价争夺更多的客户和更丰厚的利润。假设这两个网站在现有策略下各可以获得 1000 万元的利润。如果一方单独降价，就能扩大市场份额，可以获得 1500 万元利润，此时，另一方的市场份额就会缩小，利润将下降到 200 万元。

如果这两个网站同时降价，则他们都将只能得到 700 万元利润。那么，这两个网站的主管各自经过独立的理性分析后，决定采取什么策略呢？

【解】这是一个比较简单但又常见的对策问题，可以表示为图 20-6 所示的赢得矩阵。

		甲网站	
		高价	低价
乙网站	高价	1000，1000	200，1500
	低价	1500，200	700，700

图 20-6　赢得矩阵

由图 20-6 可以看出，假设乙网站采用高价策略，那么甲网站采用高价策略得 1000 万元，采用低价策略得 1500 万元。因此，甲网站应该采用低价策略；如果乙网站采用低价策略，那么甲网站采用高价策略得 200 万元，采用低价策略得 700 万元，因此，甲网站也应该采用低价策略。采用同样的方法，也可分析乙网站的情况，也就是说，不管甲网站采取什么样的策略，乙网站都应该选择低价策略。因此，这个博弈的最终结果一定是两个网站都采用低价策略，各得到 700 万元的利润。

这个对策是一个非合作对策问题，且两个局中人都肯定对方会按照个体行为理性原则决策，因此，虽然双方采用低价策略的均衡对双方都不是理想的结果，但因为两个局中人都无法信任对方，都必须防备对方利用自己的信任（如果有的话）谋取利益，所以双方都会坚持采用低价，各自得到 700 万元的利润，各得 1000 万元利润的结构是无法实现的。即使两个网站都完全清楚上述利害关系，也无法改变这种结局。

20.2 数学建模

在前面几节的讨论中，多处提到了“数学模型”，但并未对其进行解释。那么，什么是数学模型，怎么建立数学模型呢？作为本章的结束，本节主要介绍数学建模相关知识。

当需要从定量的角度分析和研究一个实际问题时，人们就要在深入调查研究、了解对象信息、作出简化假设、分析内在规律等工作的基础上，用数学的符号和语言，把它表述为数学式子，也就是数学模型，然后用通过计算得到的模型结果来解释实际问题，并接受实际的检验。这个建立数学模型的全过程就称为数学建模。

数学建模是一种数学的思考方法，是运用数学的语言和方法，通过抽象和简化，建立能近似刻画并解决实际问题的模型的一种强有力的数学手段。

1．数学模型

数学模型是客观世界中的实际事物的一种数学简化，它常常是以某种意义上接近实际事物的抽象形式存在的，但它和真实的事物有着本质的区别。要描述一个实际现象可以有很多种方式，例如，录音、录像、比喻等。为了使描述更具科学性、逻辑性、客观性和可重复性，人们采用一种普遍认为比较严格的语言来描述各种现象，这种语言就是数学。使用数学语言描述的事物就称为数学模型。

模型的一般数学形式可用下列表达式描述。

目标的评价准则：$U = f(x_i,\ y_i,\ \xi_k)$

约束条件：$g(x_i,\ y_i,\ \xi_k) \geqslant 0$

其中， xi 为可控变量， yi 为已知参数； ξk 为随机因素。

目标的评价准则一般要求达到最佳（最小或最大）、适中、满意等。准则可以是单一的，也可以是多个的。约束条件可以没有也可有多个。

当 g 是等式时，即为平衡条件。当模型中无随机因素时，称它为确定性模型，否则为

随机模型。随机模型的评价准则可用期望值、方差表示，也可用某种概率分布来表示；当可控变量只取离散值时，称为离散模型，否则称为连续模型。也可按使用的数学工具，将模型分为代数方程模型、微分方程模型、概率统计模型、逻辑模型等；若用求解方法来命名时，有直接最优化模型、数字模拟模型、启发式模型等；也有按用途来命名的，例如，分配模型、运输模型、更新模型、排队模型、存储模型等；还可以用研究对象来命名，例如，能源模型、教育模型、军事对策模型、宏观经济模型等。

2．数学建模的过程

应用数学去解决各类实际问题时，建立数学模型是十分关键的一步，同时也是十分困难的一步。建立数学模型的过程，是把错综复杂的实际问题简化、抽象为合理的数学结构的过程。要通过调查、收集数据资料，观察和研究实际对象的固有特征和内在规律，抓住问题的主要矛盾，建立起反映实际问题的数量关系，然后利用数学理论和方法去分析和解决问题。这就需要深厚而扎实的数学基础，敏锐的洞察力和想象力，对实际问题的浓厚兴趣和广博的知识面。

虽然面临的各种实际问题不一样，但数学建模的基本过程基本上是一致的，可以遵循以下过程。

（1）模型准备：了解问题的实际背景，明确其实际意义，掌握对象的各种信息。用数学语言来描述问题。

（2）模型假设：根据实际对象的特征和建模的目的，对问题进行必要的简化，并用精确的语言提出一些恰当的假设。

（3）模型建立：在假设的基础上，利用适当的数学工具来刻画各变量之间的数学关系，建立相应的数学结构。只要能够把问题描述清楚，尽量使用简单的数学工具。

（4）模型求解：利用获取的数据资料，对模型的所有参数作出计算（估计）。

（5）模型分析：对所得的结果进行数学上的分析。

（6）模型检验：将模型分析结果与实际情形进行比较，以此来验证模型的准确性、合理性和适用性。如果模型与实际较吻合，则要对计算结果给出其实际含义，并进行解释。如果模型与实际吻合较差，则应该修改假设，再次重复建模过程。

（7）模型应用：应用方式因问题的性质和建模的目的而异。

3．数学建模的方法

构造模型是一种创造性劳动，成功的模型往往是科学与艺术的结晶，一般的建模方法和思路有以下 4 种：

（1）直接分析法：根据对问题内在机理的认识，直接构造出模型。

（2）类比法：根据类似问题的模型构造新模型。

（3）数据分析法：通过试验，获得与问题密切相关的大量数据，用统计分析方法进行建模。

（4）构想法：对将来可能发生的情况给出逻辑上合理的设想和描述，然后用已有的方法构造模型，并不断修正完善，直至用户比较满意为止。

第 21 章：虚拟化、云计算与物联网

虚拟化、云计算与物联网是一组新技术，这些技术目前都是业内的热点。对于这些技术需要读者了解其基本概念及相关的结构与应用。

21.1 虚拟化

虚拟化（Virtualization）技术最早出现在 20 世纪 60 年代的 IBM 大型机系统，在 20 世纪 70 年代的 System 370 系列中逐渐流行起来，这些机器通过一种叫虚拟机监控器（Virtual Machine Monitor，VMM）的程序在物理硬件之上生成许多可以运行独立操作系统软件的虚拟机（Virtual Machine）实例。随着近年多核系统、集群、网格甚至云计算的广泛部署，虚拟化技术在商业应用上的优势日益体现，不仅降低了 IT 成本，而且还增强了系统安全性和可靠性，虚拟化的概念也逐渐深入到人们日常的工作与生活中。

虚拟化是一个广义的术语，对于不同的人来说可能意味着不同的东西，这要取决他们所处的环境。在计算机科学领域中，虚拟化代表着对计算资源的抽象，而不仅仅局限于虚拟机的概念。例如对物理内存的抽象，产生了虚拟内存技术，使得应用程序认为其自身拥有连续可用的地址空间（Address Space），而实际上，应用程序的代码和数据可能被分隔成多个碎片页或段，甚至被交换到磁盘、闪存等外部存储器上，即使物理内存不足，应用程序也能顺利执行。

21.1.1 虚拟化技术的分类

虚拟化技术是一个非常广的概念，所以其具体内容可分成不同层面的几大类型。

1．平台虚拟化

平台虚拟化（Platform Virtualization）是针对计算机和操作系统的虚拟化。我们通常所说的虚拟化主要是指平台虚拟化技术，通过使用控制程序（Control Program，也称为 Virtual Machine Monitor 或 Hypervisor），隐藏特定计算平台的实际物理特性，为用户提供抽象的、统一的、模拟的计算环境。虚拟机中运行的操作系统称为客户机操作系统（Guest OS），运行虚拟机监控器的操作系统称为主机操作系统（Host OS），当然某些虚拟机监控器可以脱离操作系统直接运行在硬件之上（如 VMWARE 的 ESX 产品）。运行虚拟机的真实系统称为主机系统。

平台虚拟化技术又可以细分为如下几个子类：

（1）全虚拟化

全虚拟化（Full Virtualization）是指虚拟机模拟了完整的底层硬件，包括处理器、物理内存、时钟、外设等，使得为原始硬件设计的操作系统或其他系统软件完全不作任何修改就可以在虚拟机中运行。操作系统与真实硬件之间的交互可以看成是通过一个预先规定的硬件接口进行的。全虚拟化 VMM（Virtual Machine Monitor）以完整模拟硬件的方式提供全部接口（同时还必须模拟特权指令的执行过程）。举例而言，x86 体系结构中，对于操作系统切换进程页表的操作，真实硬件通过提供一个特权 CR3 寄存器来实现该接口，操作系统只需执行 "mov pgtable,%%cr3" 汇编指令即可。全虚拟化 VMM 必须完整地模拟该接口执行的全过程。如果硬件不提供虚拟化的特殊支持，那么这个模拟过程将会十分复杂：一般而言，VMM 必须运行在最高优先级来完全控制主机系统，而 Guest OS 需要降级运行，从而不能执行特权操作。当 Guest OS 执行前面的特权汇编指令时，主机系统产生异常（General Protection Exception），执行控制权重新从 GuestOS 转到 VMM 手中。VMM 事先分配一个变量作为影子 CR3 寄存器给 Guest OS，将 pgtable 代表的客户机物理地址（Guest Physical Address）填入影子 CR3 寄存器，然后 VMM 还需要 pgtable 翻译成主机物理地址（Host Physical Address）并填入物理 CR3 寄存器，最后返回到 Guest OS 中。随后 VMM 还将处理复杂的 Guest OS 缺页异常（Page Fault）。比较著名的全虚拟化 VMM 有 Microsoft Virtual PC、VMware Workstation、SUN Virtual Box、Parallels Desktop forMac 和 QEMU。

（2）超虚拟化

超虚拟化（Paravirtualization）是一种修改 Guest OS 部分访问特权状态的代码以便直接与 VMM 交互的技术。在超虚拟化虚拟机中，部分硬件接口以软件的形式提供给客户机操作系统，这可以通过 Hypercall（VMM 提供给 Guest OS 的直接调用，与系统调用类似）的方式来提供。例如，Guest OS 把切换页表的代码修改为调用 Hypercall 来直接完成修改影子

CR3 寄存器和翻译地址的工作。由于不需要产生额外的异常和模拟部分硬件执行流程，超虚拟化可以大幅度提高性能，比较著名的 VMM 有 Denali、Xen。

（3）硬件辅助虚拟化

硬件辅助虚拟化（Hardware-Assisted Virtualization）是指借助硬件（主要是主机处理器）的支持来实现高效的全虚拟化。例如有了 Intel-VT 技术的支持，Guest OS 和 VMM 的执行环境自动地完全隔离开来，Guest OS 有自己的“全套寄存器”，可以直接运行在最高级别。因此在上面的例子中，Guest OS 能够执行修改页表的汇编指令。Intel-VT 和 AMD-V 是目前 x86 体系结构上可用的两种硬件辅助虚拟化技术。

（4）部分虚拟化

部分虚拟化（Partial Virtualization）中，VMM 只模拟部分底层硬件，因此客户机操作系统不作修改是无法在虚拟机中运行的，其他程序可能也需要进行修改。在历史上，部分虚拟化是通往全虚拟化道路上的重要里程碑，最早出现在第一代的分时系统 CTSS 和 IBM M44/44X 实验性的分页系统中。

（5）操作系统级虚拟化

在传统操作系统中，所有用户的进程本质上是在同一个操作系统的实例中运行，因此内核或应用程序的缺陷可能影响到其他进程。操作系统级虚拟化（Operating System Level Virtualization）是一种在服务器操作系统中使用的轻量级的虚拟化技术，内核通过创建多个虚拟的操作系统实例（内核和库）来隔离不同的进程，不同实例中的进程完全不了解对方的存在。比较著名的有 Solaris Container、FreeBSD Jail 和 OpenVZ 等。

2．资源虚拟化

资源虚拟化（Resource Virtualization），针对特定的系统资源的虚拟化，比如内存、存储、网络资源等。

3．应用程序虚拟化

应用程序虚拟化（Application Virtualization），包括仿真、模拟、解释技术等。

虽然上面对虚拟化的分类已较为精确，但这种分类并不是绝对的，一个优秀的虚拟化软件往往融合了多项技术。例如 VMware Workstation 是一个著名的全虚拟化的 VMM，但是它使用了一种被称为动态二进制翻译的技术把对特权状态的访问转换成对影子状态的操作，从而避免了低效的 Trap-And-Emulate 的处理方式，这与超虚拟化相似，只不过超虚拟化是静态地修改程序代码。对于超虚拟化而言，如果能利用硬件特性，那么虚拟机的管理将会大大简化，同时还能保持较高的性能。

21.1.2 虚拟化的模式

虚拟化可以通过很多方法来证实。它不是一个单独的实体，而是一组模式和技术的集合，这些技术提供了支持资源的逻辑表示所需的功能，以及通过标准接口将其呈现给这些资源的消费者所需的功能。这些模式本身都是前面介绍过的各种不同虚拟形式的重复出现。

下面是在实现虚拟化时常常使用的一些模式和技术。

1．单一资源多个逻辑表示

这种模式是虚拟化最广泛使用的模式之一。它只包含一个物理资源，但是它向消费者呈现的逻辑表示却仿佛它包含多个资源一样。消费者与这个虚拟资源进行交互时就仿佛自己是唯一的消费者一样，而不会考虑他正在与其他消费者一起共享资源。

2．多个资源单一逻辑表示

这种模式包含了多个组合资源，以便将这些资源表示为提供单一接口的单个逻辑表示形式。在利用多个功能不太强大的资源来创建功能强大且丰富的虚拟资源时，这是一种非常有用的模式。存储虚拟化就是这种模式的一个例子。在服务器方面，集群技术可以提供这样的幻想：消费者只与一个系统（头节点）进行交互，而集群事实上可以包含很多的处理器或节点。实际上，这就是从 IT 技术设施的角度看到的网格可以实现的功能。

3．在多个资源之间提供单一逻辑表示

这种模式包括一个以多个可用资源之一的形式表示的虚拟资源。虚拟资源会根据指定的条件来选择一个物理资源实现，例如资源的利用、响应时间或邻近程度。尽管这种模式与上一种模式非常类似，但是它们之间有一些细微的差别。首先，每个物理资源都是一个完整的副本，它们不会在逻辑表示层上聚集在一起。其次，每个物理资源都可以提供逻辑表示所需要的所有功能，而不是像前一种模式那样只能提供部分功能。这种模式的一个常见例子是使用应用程序容器来均衡任务负载。在将请求或事务提交给应用程序或服务时，消费者并不关心到底是几个容器中执行的哪一个应用程序的副本为请求或事务提供服务。消费者只是希望请求或事务得到处理。

4．单个资源单一逻辑表示

这是用来表示单个资源的一种简单模式，就仿佛它是别的什么资源一样。启用 Web 的企业后台应用程序就是一个常见的例子。在这种情况下，我们不是修改后台的应用程序，而是创建一个前端来表示 Web 界面，它会映射到应用程序接口中。这种模式允许通过对后台应用程序进行最少的修改（或根本不加任何修改）来重用一些基本的功能。也可以根据无法

修改的组件，使用相同的模式构建服务。

5．复合或分层虚拟

这种模式是前面介绍的一种或多种模式的组合，它使用物理资源来提供丰富的功能集。信息虚拟化是这种模式一个很好的例子。它提供了底层所需要的功能，这些功能用于管理对资源、包含有关如何处理和使用信息的元数据，以及对信息进行处理的操作的全局命名和引用。例如 Open Grid Services Architecture （ OGSA ）或者 Grid Computing Components，实际上都是虚拟化的组合或虚拟化的不同层次。

21.2 云计算

云计算是一种基于互联网的计算方式，通过这种方式，共享的软硬件资源和信息可以按需提供给计算机和其他设备。云其实是网络、互联网的一种比喻说法。云计算的核心思想，是将大量用网络连接的计算资源统一管理和调度，构成一个计算资源池向用户按需服务。提供资源的网络称为云。狭义云计算指 IT 基础设施的交付和使用模式，指通过网络以按需、易扩展的方式获得所需资源；广义云计算指服务的交付和使用模式，指通过网络以按需、易扩展的方式获得所需服务。这种服务可以是 IT 和软件、互联网相关，也可是其他服务。

通俗一点来说，提供资源的网络称为云。云中的资源在使用者看来是可以无限扩展的，并且可以随时获取，按需使用，随时扩展，按使用付费。这种特性经常被称为像水电一样使用 IT 基础设施。这就好比是从古老的单台发电机模式转向了电厂集中供电的模式。它意味着计算能力也可以作为一种商品进行流通，就像煤气、水电一样，取用方便，费用低廉。最大的不同在于，它是通过互联网进行传输的。

21.2.1 云计算的特点

云计算是一项交叉学科技术，所以它有许多的特点：

（1）计算资源集成提高设备计算能力。云计算把大量计算资源集中到一个公共资源池中，通过租用的方式共享计算资源。虽然单个用户在云计算平台获得服务水平受到网络带宽等各因素影响，未必获得优于本地主机所提供的服务，但是从整个社会资源的角度而言，整体的资源调控降低了部分地区峰值荷载，提高了部分荒废的主机的运行率，从而提高了资源利用率。

（2）分布式数据中心保证系统容灾能力。分布式数据中心可将云端的用户信息备份到

地理上相互隔离的数据库主机中，甚至用户自己也无法判断信息的确切备份地点。该特点不仅仅提供了数据恢复的依据，也使得网络病毒和网络黑客的攻击失去目的性而变成徒劳，大大提高系统的安全性和容灾能力。

（3）软硬件相互隔离减少设备依赖性。虚拟化层将云平台上方的应用软件和下方的基础设备隔离开来。技术设备的维护者无法看到设备中运行的具体应用。同时对软件层的用户而言基础设备层是透明的，用户只能看到虚拟化层中虚拟出来的各类设备。这种架构减少了设备依赖性，也为动态的资源配置提供可能。

（4）平台模块化设计体现高可扩展性。目前主流的云计算平台均根据 SPI 架构在各层集成功能各异的软硬件设备和中间件软件。大量中间件软件和设备提供针对该平台的通用接口，允许用户添加本层的扩展设备。部分云与云之间提供对应接口，允许用户在不同云之间进行数据迁移。类似功能更大程度上满足了用户需求，集成了计算资源，是未来云计算的发展方向之一。

（5）虚拟资源池为用户提供弹性服务。云平台管理软件将整合的计算资源根据应用访问的具体情况进行动态调整，包括增大或减少资源的要求。因此云计算对于在非恒定需求的应用，如对需求波动很大、阶段性需求等，具有非常好的应用效果。在云计算环境中，既可以对规律性需求通过事先预测事先分配，也可根据事先设定的规则进行实时公台调整。弹性的云服务可帮助用户在任意时间得到满足需求的计算资源。

（6）按需付费降低使用成本。作为云计算的代表按需提供服务、按需付费是目前各类云计算服务中不可或缺的一部分。对用户而言，云计算不但省去了基础设备的购置运维费用，而且能根据企业成长的需要不断扩展订购的服务，不断更换更加适合的服务，提高了资金的利用率。

21.2.2 云计算的类型

云计算包括三种基本类型。

1．软件即服务

软件即服务（Software-as-a-Service，SaaS）是基于互联网提供软件服务的软件应用模式。作为一种在 21 世纪开始兴起的创新的软件应用模式，SaaS 是软件科技发展的最新趋势。

SaaS 提供商为企业搭建信息化所需要的所有网络基础设施及软件、硬件运作平台，并负责所有前期的实施、后期的维护等一系列服务，企业无须购买软硬件、建设机房、招聘 IT

人员，即可通过互联网使用信息系统。就像打开自来水龙头就能用水一样，企业根据实际需要，从 SaaS 提供商租赁软件服务。

2．平台即服务

平台即服务（Platform-as-a-Service，PaaS）是把服务器平台或者开发环境作为一种服务提供的商业模式，如将软件研发的平台作为一种服务，以 SaaS 的模式提交给用户。因此，PaaS 也是 SaaS 模式的一种应用。但是，PaaS 的出现可以加快 SaaS 的发展，尤其是加快 SaaS 应用的开发速度。早在 2007 年，国内外 SaaS 厂商就先后推出了自己的 PaaS 平台。

PaaS 之所以能够推进 SaaS 的发展，主要在于它能够提供企业进行定制化研发的中间件平台，同时涵盖数据库和应用服务器等。PaaS 可以提高在 Web 平台上利用的资源数量。

3．基础设施即服务

基础设施即服务（Infrastructure as a Service，IaaS）是指消费者通过 Internet 可以从完善的计算机基础设施获得服务，如《纽约时报》就使用成百上千台 Amazon EC2 实例在 36 小时内处理 TB 级的文档数据。如果没有 EC2，《纽约时报》处理这些数据将要花费数天或者数月的时间。

21.2.3 云计算的应用

云计算目前已应用到各个领域，大多大型电子商务企业近几年也将云计算的布局作为自己战略目标中的一个方面。下面将谈一谈具体的应用场景。

1．云安全

云安全（Cloud Security）是一个从“云计算”演变而来的新名词。云安全的策略构想是：使用者越多，每个使用者就越安全，因为如此庞大的用户群，足以覆盖互联网的每个角落，只要某个新木马病毒出现，就会立刻被截获。

“云安全”通过网状的大量客户端对网络中软件行为的异常监测，获取互联网中木马、恶意程序的最新信息，推送到 Server 端进行自动分析和处理，再把病毒和木马的解决方案分发到每一个客户端。

2．云存储

应用云存储是在云计算概念上延伸和发展出来的一个新的概念，是指通过集群应用、网格技术或分布式文件系统等功能，将网络中大量各种不同类型的存储设备通过应用软件集合起来协同工作，共同对外提供数据存储和业务访问功能的一个系统。当云计算系统运算和处

理的核心是大量数据的存储和管理时，云计算系统中就需要配置大量的存储设备，那么云计算系统就转变成为一个云存储系统，所以云存储是一个以数据存储和管理为核心的云计算系统。

3．云呼叫

应用云呼叫中心是基于云计算技术而搭建的呼叫中心系统，企业无须购买任何软、硬件系统，只需具备人员、场地等基本条件，就可以快速拥有属于自己的呼叫中心，软硬件平台、通信资源、日常维护与服务由服务器商提供。具有建设周期短、投入少、风险低、部署灵活、系统容量伸缩性强、运营维护成本低等众多特点；无论是电话营销中心还是客户服务中心，企业只需按需租用服务，便可建立一套功能全面、稳定、可靠、座席可分布全国各地，全国呼叫接入的呼叫中心系统。

4．云会议

应用云会议是基于云计算技术的一种高效、便捷、低成本的会议形式。它是视频会议与云计算的完美结合，带来了最便捷的远程会议体验。使用者只需通过互联网界面，进行简单易用的操作，便可快速高效地与全球各地团队及客户同步分享语音、数据文件及视频，而会议中数据的传输、处理等复杂技术由云会议服务商帮助使用者进行操作。目前国内云会议大多以 SaaS 模式为主体，其服务内容包括电话、网络、视频等形式。

21.3 物联网

顾名思义，物联网（The Internet of Things，IoT）是实现物物相连的互联网络。其内涵包含两个方面：第一，物联网的核心和基础仍然是互联网，是在互联网基础上延伸和扩展的网络；第二，其用户端延伸和扩展到了任何物体与物体之间，使其进行信息交换和通信。

物联网是将无处不在的末端设备和设施，包括具备“内在智能”的传感器、移动终端、工业系统、楼控系统、家庭智能设施、视频监控系统等和“外在使能”的，如贴上 RFID 的各种资产、携带无线终端的个人与车辆等“智能化物件或动物”或“智能尘埃”，通过各种无线、有线的长距离/短距离通信网络实现互联互通、应用大集成，以及基于云计算的 SaaS 营运等模式。提供安全可控乃至个性化的实时在线监测、定位追溯、报警联动、调度指挥、预案管理、远程控制、安全防范、远程维保、在线升级、统计报表、决策支持等管理和服务功能。实现对“万物”的“高效、节能、安全、环保”的“管、控、营”一体化。

21.3.1 物联网的层次结构

物联网可以分为三个层次，底层是用来感知数据的感知层，即利用传感器、二维码、RFID等设备随时随地获取物体的信息。第二层是数据传输处理的网络层，即通过各种传感网络与互联网的融合，将物体当前的信息实时准确地传递出去。第三层则是与行业需求结合的应用层，即通过智能计算、云计算等对物体进行智能化控制。

1．感知层

感知层用于识别物体、采集信息。感知层包括二维码标签和识读器、RFID 标签和读写器、摄像头、GPS、传感器、M2M 终端、传感器网关等，主要功能是识别物体、采集信息，与人体结构中皮肤和五官的作用类似。

感知层解决的是人类世界和物理世界的数据获取问题。它首先通过传感器、数码相机等设备，采集外部物理世界的数据，然后通过 RFID、条码、工业现场总线、蓝牙、红外等短距离传输技术传递数据。感知层所需要的关键技术包括检测技术、短距离无线通信技术等。

对于目前关注和应用较多的 RFID 网络来说，附着在设备上的 RFID 标签和用来识别 RFID 信息的扫描仪、感应器都属于物联网的感知层。在这一类物联网中被检测的信息就是 RFID 标签的内容，现在的电子不停车收费系统（Electronic Toll Collection，ETC）、超市仓储管理系统、飞机场的行李自动分类系统等都用到了这个层次的设备。

2．网络层

网络层用于传递信息和处理信息。网络层包括通信网与互联网的融合网络、网络管理中心、信息中心和智能处理中心等。网络层将感知层获取的信息进行传递和处理，类似于人体结构中的神经中枢和大脑。

网络层解决的是传输和预处理感知层所获得数据的问题。这些数据可以通过移动通信网、互联网、企业内部网、各类专网、小型局域网等进行传输。特别是在三网融合后，有线电视网也能承担物联网网络层的功能，有利于物联网的加快推进。网络层所需要的关键技术包括长距离有线和无线通信技术、网络技术等。

物联网的网络层将建立在现有的移动通信网和互联网基础上。物联网通过各种接入设备与移动通信网和互联网相连，例如，手机付费系统中由刷卡设备将内置手机的 RFID 信息采集上传到互联网，网络层完成后台鉴权认证，并从银行网络划账。

网络层中的感知数据管理与处理技术是实现以数据为中心的物联网的核心技术，包括传感网数据的存储、查询、分析、挖掘和理解，以及基于感知数据决策的理论与技术。云计算

平台作为海量感知数据的存储、分析平台，将是物联网网络层的重要组成部分，也是应用层众多应用的基础。在产业链中，通信网络运营商和云计算平台提供商将在物联网网络层占据重要的地位。

3．应用层

应用层实现广泛智能化。应用层是物联网与行业专业技术的深度融合，结合行业需求实现行业智能化，这类似于人们的社会分工。

物联网应用层利用经过分析处理的感知数据，为用户提供丰富的特定服务。物联网的应用可分为监控型（物流监控、污染监控）、查询型（智能检索、远程抄表）、控制型（智能交通、智能家居、路灯控制）和扫描型（手机钱包、高速公路不停车收费）等。

应用层解决的是信息处理和人机交互的问题。网络层传输而来的数据在这一层进入各类信息系统进行处理，并通过各种设备与人进行交互。这一层也可按形态直观地划分为两个子层。一个是应用程序层，进行数据处理，它涵盖了国民经济和社会的每一领域，包括电力、医疗、银行、交通、环保、物流、工业、农业、城市管理、家居生活等，其功能可包括支付、监控、安保、定位、盘点、预测等，可用于政府、企业、社会组织、家庭、个人等。这正是物联网作为深度信息化的重要体现。另一个是终端设备层，提供人机接口。 物联网虽然是“物物相连的网”，但最终要以人为本，还是需要人的操作与控制，不过这里的人机界面已远远超出现实中人与计算机交互的概念，而是泛指与应用程序相连的各种设备与人的交互。

应用层是物联网发展的体现，软件开发、智能控制技术将会为用户提供丰富多彩的物联网应用。各种行业和家庭应用的开发将会推动物联网的普及，也给整个物联网产业链带来丰厚的利润。

21.3.2 物联网的相关领域与技术

正如前文所述，物联网是一种综合应用型技术，其发展离不开相关基础技术的进步。下面将介绍与物联网相关的一些技术。

1．射频识别技术

射频识别技术（Radio Frequency Identification，RFID），又称电子标签，是一种通信技术，可通过无线电信号识别特定目标并读写相关数据，而无须识别系统与特定目标之间建立机械或光学接触。该技术是物联网的一项核心技术，很多物联网应用都离不开它。

最初在技术领域，应答器是指能够传输信息、回复信息的电子模块，近些年，由于射频

技术发展迅猛，应答器有了新的说法和含义，又称为智能标签或标签。RFID 电子阅读器（读写器）通过天线与 RFID 电子标签进行无线通信，可以实现对标签识别码和内存数据的读出或写入操作。典型的阅读器包含有高频模块（发送器和接收器）、控制单元及阅读器天线。

RFID 采用的是非接触的自动识别技术，它通过射频信号自动识别目标对象并获取相关数据，识别工作无须人工干预，可工作于各种恶劣环境。RFID 技术可识别高速运动物并可同时识别多个标签，操作快捷方便。这种系统一般由一个询问器（或阅读器）和很多应答器（或标签）组成。

RFID 的基本组成部分通常包括标签、阅读器和天线。

（1）标签（Tag）：由耦合元件及芯片组成，每个标签具有唯一的电子编码，附着在物体上标识目标对象。

（2）阅读器（Reader）：读取（有时还可以写入）标签信息的设备，可设计为手持式或固定式。

（3）天线（Antenna）：在标签和读取器间传递射频信号。

RFID 技术的基本工作原理并不复杂：标签进入磁场后，接收解读器发出的射频信号，凭借感应电流所获得的能量发送出存储在芯片中的产品信息（Passive Tag，无源标签或被动标签），或者由标签主动发送某一频率的信号（Active Tag，有源标签或主动标签），解读器读取信息并解码后，送至中央信息系统进行有关数据处理。

一套完整的 RFID 系统，是由阅读器与电子标签（即应答器）及应用软件系统三个部分组成的，其工作原理是 Reader 发射一特定频率的无线电波能量给 Transponder，用以驱动 Transponder 电路将内部的数据送出，此时 Reader 便依序接收解读数据，送给应用程序作相应的处理。

以 RFID 卡片阅读器及电子标签之间的通信及能量感应方式来看大致上可以分成：感应耦合（Inductive Coupling）及后向散射耦合（Backscatter Coupling）两种。一般低频的 RFID 大都采用第一种方式，而较高频大多采用第二种方式。

阅读器根据使用的结构和技术不同可以是读或读/写装置，也是 RFID 系统信息控制和处理中心。阅读器通常由耦合模块、收发模块、控制模块和接口单元组成。阅读器和应答器之间一般采用半双工通信方式进行信息交换，同时阅读器通过耦合给无源应答器提供能量和时序。在实际应用中，可进一步通过 Ethernet 或 WLAN 等实现对物体识别信息的采集、处理及远程传送等管理功能。应答器是 RFID 系统的信息载体，目前应答器大多由耦合原件（线圈、微带天线等）和微芯片组成无源单元。

2．二维码技术

二维码（2-dimensional bar code），如图 21-1 所示。它是用某种特定的几何图形按一定规律在平面（二维方向上）分布的记录数据符号信息的黑白相间的图形。在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图像输入设备或光电扫描设备自动识读以实现信息自动处理。

图 21-1　二维码

21.3.2 物联网的相关领域与技术

正如前文所述，物联网是一种综合应用型技术，其发展离不开相关基础技术的进步。下面将介绍与物联网相关的一些技术。

1．射频识别技术

射频识别技术（Radio Frequency Identification，RFID），又称电子标签，是一种通信技术，可通过无线电信号识别特定目标并读写相关数据，而无须识别系统与特定目标之间建立机械或光学接触。该技术是物联网的一项核心技术，很多物联网应用都离不开它。

最初在技术领域，应答器是指能够传输信息、回复信息的电子模块，近些年，由于射频技术发展迅猛，应答器有了新的说法和含义，又称为智能标签或标签。RFID 电子阅读器（读写器）通过天线与 RFID 电子标签进行无线通信，可以实现对标签识别码和内存数据的读出或写入操作。典型的阅读器包含有高频模块（发送器和接收器）、控制单元及阅读器天线。

RFID 采用的是非接触的自动识别技术，它通过射频信号自动识别目标对象并获取相关数据，识别工作无须人工干预，可工作于各种恶劣环境。RFID 技术可识别高速运动物并可同时识别多个标签，操作快捷方便。这种系统一般由一个询问器（或阅读器）和很多应答器（或标签）组成。

RFID 的基本组成部分通常包括标签、阅读器和天线。

（1）标签（Tag）：由耦合元件及芯片组成，每个标签具有唯一的电子编码，附着在物体上标识目标对象。

（2）阅读器（Reader）：读取（有时还可以写入）标签信息的设备，可设计为手持式或固定式。

（3）天线（Antenna）：在标签和读取器间传递射频信号。

RFID 技术的基本工作原理并不复杂：标签进入磁场后，接收解读器发出的射频信号，凭借感应电流所获得的能量发送出存储在芯片中的产品信息（Passive Tag，无源标签或被动标签），或者由标签主动发送某一频率的信号（Active Tag，有源标签或主动标签），解读器读取信息并解码后，送至中央信息系统进行有关数据处理。

一套完整的 RFID 系统，是由阅读器与电子标签（即应答器）及应用软件系统三个部分组成的，其工作原理是 Reader 发射一特定频率的无线电波能量给 Transponder，用以驱动 Transponder 电路将内部的数据送出，此时 Reader 便依序接收解读数据，送给应用程序作相应的处理。

以 RFID 卡片阅读器及电子标签之间的通信及能量感应方式来看大致上可以分成：感应耦合（Inductive Coupling）及后向散射耦合（Backscatter Coupling）两种。一般低频的 RFID 大都采用第一种方式，而较高频大多采用第二种方式。

阅读器根据使用的结构和技术不同可以是读或读/写装置，也是 RFID 系统信息控制和处理中心。阅读器通常由耦合模块、收发模块、控制模块和接口单元组成。阅读器和应答器之间一般采用半双工通信方式进行信息交换，同时阅读器通过耦合给无源应答器提供能量和时序。在实际应用中，可进一步通过 Ethernet 或 WLAN 等实现对物体识别信息的采集、处理及远程传送等管理功能。应答器是 RFID 系统的信息载体，目前应答器大多由耦合原件（线圈、微带天线等）和微芯片组成无源单元。

2．二维码技术

二维码（2-dimensional bar code），如图 21-1 所示。它是用某种特定的几何图形按一定规律在平面（二维方向上）分布的记录数据符号信息的黑白相间的图形。在代码编制上巧妙地利用构成计算机内部逻辑基础的“0”、“1”比特流的概念，使用若干个与二进制相对应的几何形体来表示文字数值信息，通过图像输入设备或光电扫描设备自动识读以实现信息自动处理。

1.png

二维码具有条码技术的一些共性：每种码制有其特定的字符集、每个字符占有一定的宽度、具有一定的校验功能等。同时还具有对不同行的信息自动识别功能及处理图形旋转变化等特点。在许多种类的二维条码中，常用的码制有 Data Matrix、Maxi Code、Aztec、 QR Code、

Vericode、PDF417、Ultracode、Code 49、Code 16K 等，QR 码是 1994 年由日本 Denso-Wave 公司发明的。QR 来自英文 Quick Response 的缩写，即快速反应的意思，源自发明者希望 QR 码可让其内容快速被解码。QR 码最常见于日本、韩国，是目前日本最流行的二维空间条码。

二维条码/二维码能够在横向和纵向两个方位同时表达信息，因此能在很小的面积内表达大量的信息。其信息量远远超过原来的条码技术，原来的条码技术仅能存储十多个字符，而二维码存储容量可达数千字符。以 PDF417 编码格式为例：若采用扩展的字母数字压缩格式，可容纳 1850 个字符；若采用二进制/ASCII 格式，可容纳 1108 字节；若采用数字压缩格式，可容纳 2710 个数字。

3．传感网

传感网是由随机分布的，集成有传感器（传感器有很多种类型，包括温度、湿度、速度、气敏等）、数据处理单元和通信单元的微小节点，通过自组织的方式构成的无线网络。

传感网借助于节点中内置的传感器测量周边环境中的热、红外、声呐、雷达和地震波信号，从而探测包括温度、湿度、噪声、光强度、压力、土壤成分、移动物体的大小、速度和方向等物质现象。它给我们的生活带来了深刻的变化。然而在目前，网络功能再强大，网络世界再丰富，也终究是虚拟的，它与我们所生活的现实世界还是相隔的，在网络世界中，很难感知现实世界，很多事情还是不可能的，时代呼唤着新的网络技术。传感网络正是在这样的背景下应运而生的全新网络技术，它综合了传感器、低功耗、通信及微机电等技术，将现实的世界与虚拟的网络世界联系起来，达到很多意想不到的效果。目前传感网技术已广泛应用于石油、化工、电力、医药、生物、航空、航天、国防、能源、冶金、电子等众多行业。可以预见，在不久的将来，传感网络将给我们的生活方式带来革命性的变化。

4．M2M

简单地说，M2M 是将数据从一台终端传送到另一台终端，也就是机器与机器（Machine to Machine）的对话。但从广义上讲 M2M 可代表机器对机器（Machine to Machine）、人对机器（Man to Machine）、机器对人（Machine to Man）、移动网络对机器（Mobile to Machine）之间的连接与通信，它涵盖了所有实现在人、机器、系统之间建立通信连接的技术和手段。

M2M 强调的是在商业活动中通过移动通信技术和设备的应用变革既有商务模式或创造出新商务模式，是机器设备间的自动通信。现在，M2M 应用遍及电力、交通、工业控制、零售、公共事业管理、医疗、水利、石油等多个行业，对于车辆防盗、安全监测、自动售货、机械维修、公共交通管理等，M2M 可以说是无所不能。

M2M 不是简单的数据在机器和机器之间的传输，更重要的是，它是机器和机器之间的

一种智能化、交互式的通信。也就是说，即使人们没有实时发出信号，机器也会根据既定程序主动进行通信，并根据所得到的数据智能化地作出选择，对相关设备发出正确的指令。可以说，智能化、交互式成为了 M2M 有别于其他应用的典型特征，这一特征下的机器也被赋予了更多的“思想”和“智慧”。

M2M 的发展前景非常好，因为在当今世界上，机器的数量至少是人的数量的 4 倍以上，机器将替代人做更多的事情，这意味着巨大的市场潜力。

在国内，也有一些企业很早就开始应用 M2M 技术。三一重工对 M2M 的应用比较成功，三一重工在其销往全球各地的工程机械（关键部位或关键部件）上加装数据采集终端。机械的运行数据通过电信运营商网络汇总到三一集团企业控制中心（Enterprise Control Center，ECC），实现对工程设备作业状况的实时监控。这样，企业控制中心可以随时发现设备运行中存在的问题（如工程机械设备上智能设备控制器检测到的油温、转速、工作压力等运行数据信息异常），并就问题立即指导客户排除故障或派出维修人员上门服务。

21.3.3 物联网的应用

物联网可以看作人类与应用系统现有互动方式的延伸，而这种延伸是通过物体通信与集成的新层面实现的。物联网将对传统的数据采集系统和局部自动辨识系统的性能有所要求，进而提升各类应用系统的价值。具体地说，就是把感应器嵌入和装备到电网、铁路、桥梁、隧道、公路、建筑、供水系统、大坝、油气管道等各种物体中，然后将物联网与现有的互联网整合起来，实现人类社会与物理系统的整合，在这个整合的网络当中，存在能力超级强大的中心计算机群，能够对整合网络内的人员、机器、设备和基础设施实施实时的管理和控制，在此基础上，人类可以以更加精细和动态的方式管理生产和生活，达到“智慧”状态，提高资源利用率和生产力水平，改善人与自然间的关系。

物联网用途广泛，遍及智能交通、环境保护、政府工作、公共安全、平安家居、智能消防、工业监测、老人护理、个人健康等多个领域。在生产生活中的应用举不胜举，下面简述几个比较典型的应用。

目前食品安全是一个被大众所关注的主题，即便是超市的食品，人们也很难弄清楚这些食品的来源，以及相关情况。当物联网体系建立好以后，超市里销售的禽、肉、蛋、奶，在包装上可以嵌入微型感应器，顾客只需用手机扫描，就能了解食品的产地和转运、加工的时间地点，甚至还能显示加工环境的照片，是否绿色安全，一目了然。

在医疗方面，也可以应用物联网。将传感器嵌入到家人的手表里，即使用户在千里之外，也可以随时掌握家人的体征。用这种方法，医生也可以随时随地了解病人的体征，为病人诊断看病。

如果在汽车和汽车钥匙上都植入微型感应器，酒后驾车现象就可能被杜绝。当喝了酒的司机掏出汽车钥匙时，钥匙能通过气味感应器察觉到酒气，并通过无线信号通知汽车“不要发动”，汽车会自动罢工，并能够“命令”司机的手机给其亲友发短信，通知他们司机所在的位置，请亲友们来处理。