第3章 信息系统技术基础

信息系统的技术基础是指支持信息系统构建和运行的设备和技术，主要包括覆盖信息系统中信息处理生命周期所需的信息输入、输出、存储、处理和传输等各种软硬件设备，以及支撑信息系统构建的开发方法。

3.1 计算机系统

计算机系统可划分为硬件和软件两部分。硬件由主机（中央处理器、高速缓冲存储器、主存储器）和外设（输入设备、输出设备、辅助存储器、网络设备、声卡、显卡等）等物理实体构成。软件是一系列按照特定顺序组织的数据和指令，并控制硬件完成指定的功能。计算机软件可以进一步分为系统软件和应用软件，系统软件是指支持应用软件的运行，为用户开发应用软件提供平台支撑的软件，而应用软件是指计算机用户利用计算机的软、硬件资源为某一专门的应用目的而开发的软件。典型的计算机系统组成如图 3-1 所示。

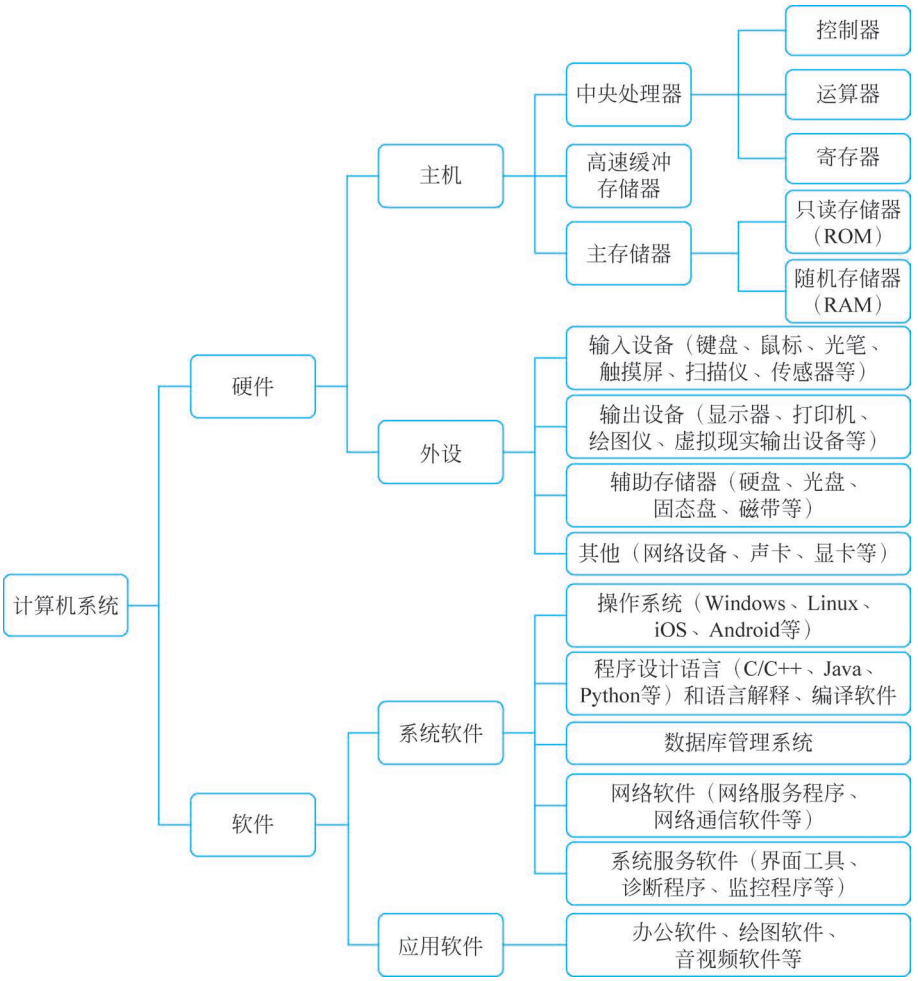


图 3-1 计算机系统组成

3.1.1 计算机硬件

计算机系统中的各硬件主要功能如下。

1. 中央处理器

中央处理器(Central Processing Unit,CPU)作为计算机系统的运算和控制核心,是信息处理、程序运行的最终执行单元。CPU 由运算器和控制器两个主要部分组成。运算器是计算机的运算单元。主要用于完成算术运算和逻辑运算。运算器内部结构还可细分为算术逻辑单元、累加器、状态寄存器和通用寄存器。算术逻辑单元主要用于完成算术、逻辑操作;累加器用于暂存操作数或运算结果;状态寄存器用于存放运算中产生的状态信息;通用寄存器用于暂存操作数与数据地址。算术逻辑单元、累加器及通用寄存器的位数决定了 CPU 的字长,即计算机中作为一个整体被传送和运算的二进制位数。因此,如果一台计算机以 32 位为一个整体进行传送与运算,则称为 32 位机。控制器是计算机的神经中枢,它按照主频的节拍发出各种控制信息,以指挥整个计算机工作。

CPU 的运算速度是决定计算机系统性能的重要指标。由于微电子技术的飞速发展,芯片性能的日益提高,目前 CPU 已经能够集成在单片集成电路上形成微处理器芯片。

Intel 公司的 x86 系列处理器占据了笔记本/台式机/服务器的市场,1978 年,Intel 推出 16 位处理器 8086,后续不断加入新特性,2023 年发布了 13 代酷睿系列处理器,采用的是 7nm 制程工艺。x86 是复杂指令集计算机(Complex Instruction Set Computer,CISC),具有许多不同格式的指令,然而,操作系统一般只会用到小部分。与之对比,精简指令集计算机(Reduced Instruction Set Computer,RISC)采用“很少”的指令,同时每个指令的模式也“很少”,代表性的架构有 ARM(Advanced RISC Machines)、RISC-V 等。AMD 是 Intel 的主要竞争对手,2022 年 AMD 推出的 Zen 4 架构采用了 5nm 制程工艺。我国的 CPU 近几年发展非常迅速,主要产品有中国科学院计算所研发的龙芯系列,采用的是 MIPS 架构(Microprocessor without Interlocked Piped Stages architecture),属于 RISC 的处理器架构。最新款龙芯 3A4000/3B4000 4 核处理器采用 28nm 制程工艺,主频为 1.8~2.0GHz。华为公司研发的海思处理器采用 ARM 架构,最新款为麒麟 990 5G 版,采用 7nm 制程工艺,集成了 5G 基带。国防科技大学研发的飞腾系列,架构从 SPARC 转向 ARM,最新款 FT-2000/4 4 核处理器,采用 16nm 制程工艺,主频 3.0GHz,功耗仅 10W。

2. 存储器

存储器通常分为主存储器和辅助存储器两类,主存储器又称为内存储器,简称内存或主存,是在计算机运行过程中用来存储数据和程序指令的。辅助存储器又称为外部存储器,简称外存或辅存,用于数据和程序的持久化存储。

内存包括只读存储器(ROM)和随机存储器(RAM)。只读存储器是指只能从中读出信息,不能写入信息的存储器。常用它存放计算机的启动程序、自检程序及磁盘引导程序等。随机存储器是指可以在任意时刻,从存储器的任意单元读出信息,或将信息写入

任意存储单元，而读写信息所需时间与存储单元的位置无关的存储器。常用它存放计算机运行过程中所需的程序和数据。当运行结束，程序和数据将保存在辅助存储器内。断电后机器内内存中的信息自动消失。内存的最小数据存储单元是字节(byte)，每个字节可以存储一个数值、字符或符号，内存的容量是决定计算机处理速度和处理能力的重要指标，早期的微机通常有 640KB 内存，现在的主流的微机内存容量已超过 16GB。

在计算机运行过程中，可以将内存中的数据分批写入外存存储，也可以将外存中的程序和数据分批读入内存进行操作和处理。外存的特点是容量很大，价格较低，但由于外存需要通过机械部件的运动在大容量的存储设备上存取数据，所以相对于内存中的电子运动而言存取速度要慢得多。外存的主要类型有磁带、磁盘、光存储器、固态硬盘等。其中，磁带主要用于备份和归档。磁盘具备容量大、价格低等优点，是存储数据和程序最可靠、最经济和最快捷的外存设备之一，目前主流的磁盘容量可达 2TB。光存储器又分为只读的光盘和可擦除光盘，普通光盘的存储容量约为 650MB，DVD 盘片单面可达 GB 级，而蓝光 DVD 容量可达 25GB。固态硬盘是随着 Flash 存储器技术的发展，产生的一种新型的存储介质，具有速度快、体积小、重量轻等特点。

为提高存储器的性能，通常把各种不同存储容量、存取速度和价格的存储器按层次结构组成多层存储器，并通过管理软件和辅助硬件有机组合成统一的整体，使所存放的程序和数据按层次分布在各存储器中。一般采用多级层次结构来构成存储系统，由寄存器、高速缓冲存储器 Cache、内存和外存组成。图 3-2 中自上向下容量逐渐增大，速度逐级降低，成本逐级减少。

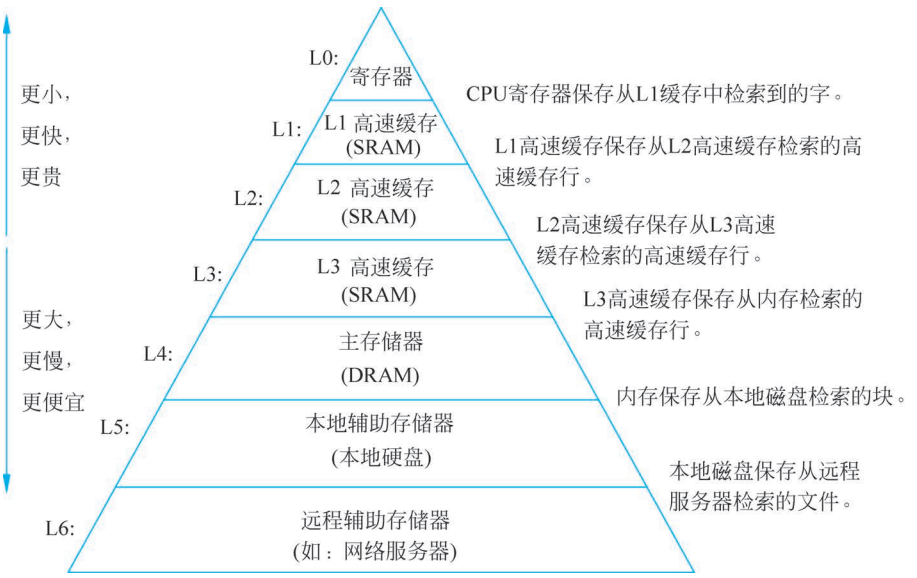


图 3-2　存储器层次

整个结构可看成主存-辅存和 Cache-主存两个层次。在辅助硬件和计算机操作系统的管理下，可把主存-辅存作为一个存储整体，形成的可寻址存储空间比主存储器空间大得多。由于辅存容量大，价格低，使得存储系统的整体平均价格降低。Cache-主存层次可

以缩小主存和CPU之间的速度差距，从整体上提高存储器系统的存取速度。一个较大的存储系统由各种不同类型的存储设备构成，形成具有多级层次结构的存储系统。该系统既有与CPU相近的速度，又有极大的容量，而价格又是较低的。可见，采用多级层次结构的存储器系统可有效地解决存储器的速度、容量和价格之间的矛盾。

3. 输入输出设备

人与计算机之间以及计算机各个部件之间进行通信都必须使用输入输出系统。

输入设备主要有键盘、鼠标、输入笔或光笔、触摸屏、图文扫描仪、条码阅读器、数码相机、手写输入设备、语音输入设备，以及声、光、电、磁等传感器。输入技术包括传感器技术、人机交互技术等。

输出设备主要有显示器(CRT、LCD、等离子体)、投影仪、打印机、绘图仪、声音输出设备、存储介质等。输出技术包括显示器件技术、信息可视化技术等。

3.1.2 计算机软件

软件主要分为系统软件和应用软件两种类型。

1. 系统软件

系统软件执行计算机的所有用户都会用到的基本任务，这些任务与硬件有关。使用现代计算机就不可能不使用它的系统软件，系统软件通常由硬件生产商(供应商)或由专门生产软件的公司提供(软件供应商)。系统软件有3种基本类型：操作系统、实用程序和语言处理器。

操作系统是计算机硬件和用户的接口。目前主流操作系统主要有Windows、UNIX和Linux操作系统。操作系统可以实现以下基本功能。

- 任务调度。操作系统能确定任务执行的顺序。同时使用字处理软件和电子表格软件时，操作系统能确切地跟踪哪一个程序正在使用处理器，哪一部分外存中数据正在被使用，计算机屏幕上需要显示什么信息等。
- 管理硬件和软件资源。操作系统执行程序时，把用户的应用程序装入内存，使不同的硬件按应用程序说明执行相应的操作。
- 维护系统安全性。操作系统要求用户输入密码进行身份认证，判断用户是否有权访问系统。这种安全性能够确保任何用户都无法越过密码检查而访问外存或其他资源。
- 多用户资源共享。大型计算机和联网的微型机允许一个以上的用户同时访问计算机资源，这个特征称为并发性(concurrency)。并发性使得多个用户虽然不是同时刻访问资源，但操作系统在每个用户小的子任务之间迅速切换，使得每个用户感觉计算机资源在唯一地为他服务。操作系统能同时处理多个用户的应用程序并做出调度，这个特征称为多任务(multitasking)。
- 处理中断。中断技术是指操作系统暂时把一个程序的执行挂起，转而执行另一个程序。当程序请求输入、输出时，或者程序超出了设置的时间限制时，就会发生中断。

实用程序是计算机里的例行程序。用户可以通过它完成基本的计算机数据处理操作，这些操作不是针对特殊用户的特殊需求的。用户可以使用实用程序进行复制文件、删除文件、整理文件内容、合并两个或多个文件等文件管理操作。还有一些实用程序允许计算机操作管理员恢复丢失或损坏的文件、监视系统性能，甚至是控制用户和计算机之间的数据流动。在计算机发展的早期，这些实用程序不是标准的。计算机程序员需要自己编写程序代码来复制文件，或完成其他功能。随着计算机生产商之间竞争的日益激烈，操作系统中许多实用程序开始标准化。

第一台计算机问世的时候，还没有出现面向程序员的程序设计语言。程序员需要将0、1组成的指令和数据序列装入计算机内存以控制计算机的运行，这是很低效的，尤其是当执行的许多任务中都有一些相似的工作（如查找数据记录和把数据装入内存等）时。为了提高效率，人们规定一套新的指令，称为高级语言，其中每一条指令完成一项操作，这种操作相对于软件总的功能而言是简单而基本的，而相对于 CPU 的操作而言又是复杂的。用这种高级程序语言来编写程序（称为源程序）就像用预制板代替砖块来造房子，效率要高得多。但 CPU 并不能直接执行这些新的指令，需要编写一个软件，专门用来将源程序中的每条指令翻译成一系列 CPU 能接受的基本指令（也称机器语言），使源程序转化成能在计算机上运行的程序。完成这种翻译的软件就是语言处理器。语言处理器提供了一种更为友好地编写计算机程序的方式。C、C++、Java 和 Python 都是常用的高级程序语言。

2. 应用软件

应用软件是直接面向最终用户的具体软件。应用软件以操作系统为基础，用程序设计语言编写，或用数据库管理系统构造，用于满足用户的各种具体要求。应用软件主要可分为通用应用软件和专用应用软件两大类。

通用应用软件是指某些通用信息处理功能的商品化软件。它的特点是具有通用性，因此可以为许多有类似应用需求的用户所使用。它所提供的功能往往可以由用户通过选择、设置和调配来满足特定需求。比较典型的有文字处理软件、表格处理软件、数值统计分析软件、财务核算软件、人事档案管理软件等。通用应用软件一般由计算机软件开发商开发和发售，用户购买该类软件后，要经过一定的配置过程才能满足用户的特定需求。某些大型和复杂的通用软件的配置、安装和调试工作也由专业技术人员来完成。而大多数通用应用软件，尤其是微型计算机的应用软件，其安装和调配往往较简单，最终用户只要按照软件说明书或经过简单培训就能独立进行。

专用应用软件也称为用户定制软件。在许多场合下，用户对数据处理的功能需求有很大的差异，当通用软件不能满足要求时，就需要由专业人士采取单独开发的方法，为用户开发具有特定功能的专用软件。

3.1.3 计算机系统类型

计算机系统的分类方式很多，按大小可以划分为微型机、工作站、小型机、大型机；按用途可以划分为科学计算系统、事务处理系统、实时控制系统等；按数据类型可以划分为

定点机、浮点机、向量机、堆栈机等；按处理机个数和种类划分为单处理机、并行处理机、多处理机、分布处理机、阵列处理机、超标量处理机、超流水线处理机、SMP(对称多处理机)、MPP(大规模并行处理机)、集群(Cluster)系统等；Flynn 提出的按指令流、数据流及其多倍性可以划分为 SISD(在一个时钟周期内，CPU 仅执行一条指令、处理一个数据流，如串行单处理机)、SIMD(在所有 CPU 上执行相同的指令，但在不同的数据流上运行，如阵列处理机)、MISD(多个指令部件对同一数据的各个处理阶段进行操作，这种机器很少见)、MIMD(多个独立或相对独立的处理机分别执行各自的程序、作业或进程，如多处理机)。

信息系统中常用的计算机系统主要有以下几种类型。

1. 微型机

微型机自 20 世纪 80 年代以后性能价格比飞速提高，这主要得益于其中的核心部件即微处理器、内存芯片以及硬盘等的进步。现在一台微型机的性能已超过了 20 世纪 60 年代大型机的水平。随着硬件性能的提高，大量微型机上可以使用的应用软件也迅速发展。而这种趋势促进了利用微型机和局域网作为信息系统硬件平台的解决方案的普及。与大型机相比，用微型机构成的信息系统投资低、收效快、使用方便。同时，由于微型机操作系统的发展，图形化用户界面使得微型机的操作更加简单方便。从系统的管理和维护方便性等方面，微型机也比大型机要容易。由于这些优点，以微型机为中心的信息系统结构逐渐取代了大型机的主导地位。

2. 工作站型计算机

工作站(workstation)型计算机是面向专业人员的一种高性能计算机，是一种高端的通用微型机。它提供比微型机更强大的性能，尤其是在图形处理、任务并行等方面的能力。工作站型计算机通常配有高分辨率的大屏、多屏显示器及容量很大的内存和外存，并且具有高性能的图形、图像处理功能。另外，连接到服务器的终端机也可称为工作站。由于工作站性能和图形处理功能强，可以快速画出极其复杂的图形，所以对一些需要高性能快速进行图形渲染功能的应用比较适合。以擅长图形处理的 SGI 工作站为例，其内部有专用于图形处理的大规模集成电路，同时配有图形设计专用软件。用户使用它们可以快速、方便地进行作图、色彩处理、光线处理以及各种明暗、纹理、过滤处理等。工作站型计算机适合应用在需要高速计算机和高速图形显示的应用领域，如科学和工程计算、软件开发、计算机辅助分析、计算机辅助制造、工程设计和应用、图形和图像处理、过程控制和信息管理等。同时可以用于智能应用的研究开发，可以高效地运行人工智能算法进行模型(深度学习模型)的学习和训练。

3. 小型机

一般小型机上的操作系统多为专用系统，比较具有代表性的小型机有 HP 公司的小型机和 DEC 公司的 VAX 小型机。小型机常采用多 CPU 结构，具有较大容量的内存和多台大容量硬盘，数据处理功能较强，可供数百个用户连接使用。小型机还可以同时连接局域网，或同时连接数十条通信线路。小型机的用途主要是作为联机事务处理系统的

服务器，或作为有较大数据流量的局域网服务器。在这类系统中，企业中的数据资源集中存放在小型机服务器上，由各个工作站共享。小型机的实时处理性能比较好，对网络用户的要求能迅速做出响应，对多台工作站的要求能及时处理，因此这种方案适用于大流量的数据处理。

4. 大型机

大型机具有很强的数据传输和信息处理能力，可供数百至数千个终端同时工作。大型机可作为中央计算机对机构的大量数据进行集中快速的处理。在这类机器上，更广泛使用的数据库是关系数据库。大型计算机可以作为联机计算机，也可以作为进行批处理的计算机。在大型商场、证券公司、银行、航空公司等机构的订票处理系统中，一般都需要采用大型机进行后台服务处理。2023 年，为面对未来人工智能、安全性、混合云和开源等工作负载，满足企业发展需求，IBM 正式公布了 z16 大型机。IBM z16 采用了 IBM Telum 双处理器芯片，有 16 个 CPU 内核，运行频率为 5.2 GHz，包括 40TB 的独立内存冗余阵列，并专为人工智能应用程序进行了优化。

计算机系统的选择是建设信息系统时的一个关键问题。目前，在信息系统建设中计算机设备仍是投资最大的一项。计算机的价格、配置、功能、外观等有许多不同，这直接关系到连接的网络、配置的操作系统以及开发的应用软件等，而且计算机设备一旦购入，就要在一个较长的时期中一直被使用。因此，在计算机选型时，应当在基本原则的指导下，与专家或咨询公司进行深入讨论，以制定出合理的计算机选型方案。

在计算机选型时，应掌握的一些基本原则如下。

（1）根据摩尔定律（Intel 公司创始人 Gordon Moore 从统计数据中发现，计算机的性能价格比每过 18 个月就翻一番），一个企业应该根据自己机构的需要和预算，选择具有较高性价比的计算机。一般微型机使用期限为 5 年，如果是过时的旧型号，则生命周期更短。而最先进的计算机价格往往过高，所以对于性能不必要求太超前。一方面，可根据需要加以扩充；另一方面，计算机发展速度很快，再先进的计算机使用周期也很短。但也不宜选择明显落后的计算机，因为落后的计算机很难找到合适的配件支持和技术服务。

（2）现在购买计算机一般应与网络设备联系起来一同考虑，而网络设备和企业的组织结构、工作流程、环境等因素紧密相连，必须统一起来考虑。例如，考虑能否与当前的计算机网络互联、能否与行业计算机网络相联。

（3）充分考虑计算机的效益，即是否能够满足企业的需求。根据计算机用于何处、进行何种业务来考虑。如在办公室中使用的计算机主要业务是文字处理，就不需要太高档，也无需多媒体功能。应从业务需要出发考虑性能最适合的计算机系统。

常用的选型方法是：通过征集计算机方案来选择。首先选择一些计算机厂商或系统集成商，要求他们根据本机构的情况提出系统方案，然后从经济上、技术上对方案进行综合评价，选出少数方案，最后细节经过技术谈判决定。对于数量多、金额大的方案，应通过招标选择。

3.2 计算机网络

3.2.1 计算机网络发展阶段

计算机网络就是利用通信设备和线路将地理位置不同的、功能独立的多个计算机系统互连起来，以功能完善的网络软件（即网络通信协议、信息交换方式、网络操作系统等）实现网络中资源共享和信息传递的系统。

到目前为止，计算机网络的发展经历了4个阶段。

第一个阶段（20世纪60年代）。计算机网络是以单个计算机为中心的远程联机系统，也称为面向终端的计算机网络。20世纪60年代初期美国航空公司的Sabre订票系统就是典型代表。在这种系统中，主机既要进行数据处理，又要承担终端间的通信，随着终端的增加，主机的运行效率下降，为此，在主机前增设一个前端处理机负责通信工作。另外，如果每个终端都利用专线与主机连接，那么不但线路的利用率低，而且费用高，为此，在终端比较集中的地方，增设一些终端控制器，以减少通信费用，可以使用比较便宜的小型计算机或微机作为前端处理机和终端控制器。

第二个阶段（20世纪60年代至70年代）。计算机网络由多个主计算机通过通信线路互连起来，为用户提供服务，这样多个主计算机互联的网络即是目前所称的计算机网络。20世纪60年代后期美国国防部开发的ARPA网就是典型代表。这种网络中多个主机之间是通过接口报文处理机（Interface Message Processor，IMP）和通信线路一起完成通信任务的。IMP和它们之间互联的通信线路一起构成了通信子网（communication subnet）。通过通信子网互联的主机负责运行用户的应用程序，向网络用户提供可供共享的软件和硬件资源，它们组成了资源子网（resource subnet）。ARPA网就是一种两级子网的结构。

第三个阶段（20世纪80年代）。该阶段是现代计算机网络互连阶段，特征是网络体系结构的形成和网络协议的标准化。在计算机通信系统的基础之上，重视网络体系结构和协议标准化的研究，建立全网统一的通信规则，用通信协议软件来实现网络内部及网络与网络之间的通信，通过网络操作系统，对网络资源进行管理，极大地简化了用户的使用，使计算机网络对用户提供透明服务。1984年，国际标准化组织（International Organization for Standardization，ISO）颁布了一套开放系统互连的参考模型（Open System Interconnection，OSI），成为了新一代网络体系结构的基础。TCP/IP协议就是支持OSI模型实现的网络协议栈。

第四个阶段（20世纪90年代以来）是互联网络与高速网络阶段。这个阶段的计算机网络技术进入新的发展阶段，其特点是互联、高速和智能化。发展了以Internet为代表的互联网，1993年美国政府公布了“国家信息基础设施”行动计划，即采用数字化大容量光纤通信网络的信息高速公路计划，对政府机构、企业、大学、科研机构和家庭的计算机进行联网。美国政府在1997年开始研究发展更加快速可靠的下一代互联网（Next Generation Internet）。可以说，网络互联和高速计算机网络正成为最新一代计算机网络

的发展方向。随着网络规模的增大与网络服务功能的增多，各国正在开展智能网络(Intelligent Network，IN)的研究，以提高通信网络开发业务的能力，并更加合理地进行网络各种业务的管理，真正以分布和开放的形式向用户提供服务。在这个阶段，以我国华为公司为代表的移动网络 5G 技术逐步进入国际领先，全球著名权威咨询公司 GlobalData 发布的 2022 年《5G 移动核心网竞争力报告》显示，华为的 5G 核心网解决方案和成熟的商用案例排名全球第一，且领先优势扩大。

3.2.2 计算机网络系统组成

计算机网络系统由以下基本元素组成，如图 3-3 所示。

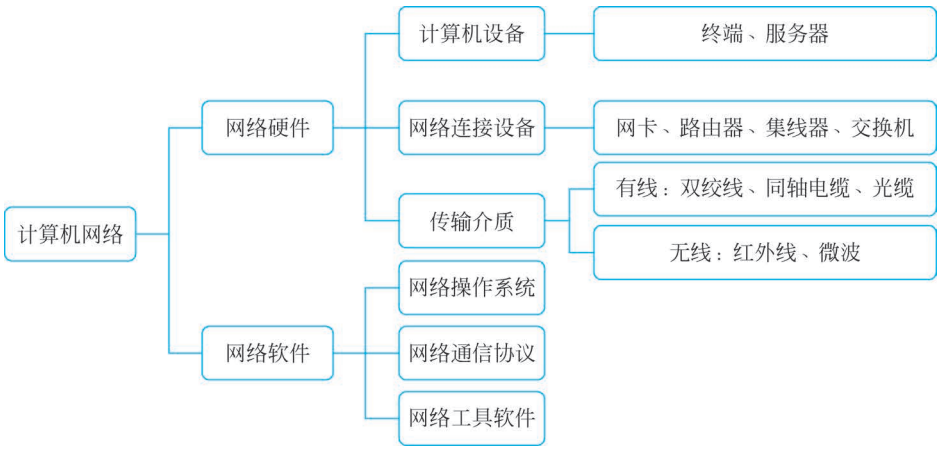


图 3-3 计算机网络系统组成

(1) 计算机设备。主要包括终端和服务器等。网络系统中往往包含多种类型的终端。一种是哑终端，它们的处理能力和存储容量有限，依赖服务器进行信息的存储和处理，哑终端通常只为用户提供键盘和显示器。另一种是智能终端，不但有键盘和显示器，而且自身有存储和处理数据的能力。微机就是网络系统中常用的一种智能终端。另外，其他一些类型的输入输出设备都可能成为网络系统中的终端，如电话、打印机等办公设备。网络中的服务器可以是微机、小型机或主机。服务器为网络中的其他用户提供各种服务，如应用程序的处理、打印服务、访问数据库管理系统等。有时一台服务器提供各种服务，在大型网络中，多台服务器往往各自提供不同的服务，如打印服务器、电子邮件服务器、传真服务器、数据库服务器、应用程序服务器等。

(2) 网络连接设备。将多个终端接入网络系统或服务器的连接设备。如网卡、路由器、集线器、交换机等，这些设备的作用如图 3-4 所示。网卡可以将两台主机直接连接，例如，小张和小王可以通过网卡连接并进行协同工作。但是如果小李的主机也要加入网络，网卡就满足不了需求了，这时就需要具有多个网口的集线器将不同主机的网线集结起来，实现最初级的网络互通。然而集线器在网络传播信息的方式是广播式的，也就是小张向小李发送的信息，小李和小王都会收到。因此，需要用交换机来根据网口地址传送信息。最后，当其他房间也需要连接到这个工作网络，就需要路由器来实现按照相同

的协议来寻址，这样，即使是不同操作系统的主机也可使实现互联了。

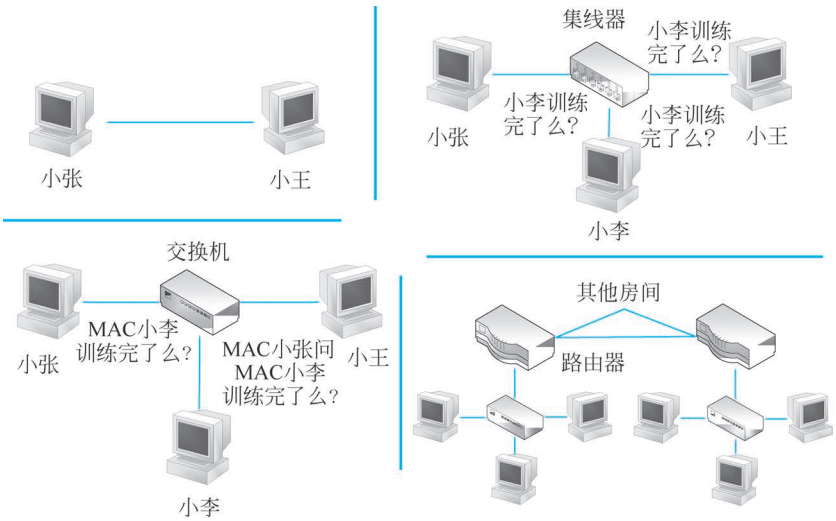


图 3-4　网络连接设备示意图

（3）传输介质。用于网络中数据或声音发射以及接收的各设备之间连接的材料，实现数据传输功能。通常介质有多种类型，每种都有自己的特性。

双绞线通常由两对或多对铜质双绞线组成，绞合的电线彼此间是绝缘的，双绞线的带宽约为 4000Hz，是理想的传输声音的媒介，主要是用来传送模拟信号的。其主要优点是价格便宜、安装方便，但双绞线不支持高速传输。

同轴电缆比双绞线贵，但比光纤便宜，能够快速传送数据，所传输的信号比双绞线清晰，但是由于比较重，难以在多个建筑物间拉线，不支持模拟信号传输。一般用于局域网内部计算机的连接，如在一个办公室、一层楼、一个建筑物及校园内部。

光缆以调制的光脉冲形式传送数字数据信号。光缆上没有电信号，因此这种传送数据的方式比较安全。由于信号纯净，衰减小，所以光缆是实现高速、大容量数据传输的好方法。光缆传输不受电信号的干扰，而且传输速率可以达到 100Mbps，理论上速率可以达到 1Gbps。光缆制造和安装复杂，成本高，目前主要用来作为网络的主干网。

无线网络指不需要铺设连接线路的传输介质，常见的方式有微波传输、蜂窝传输和红外线传输等。微波是通过大气和空间传送的，虽然使用微波无须铺设电缆，但其所需要的传输设备十分昂贵。微波是直线传输，也就是说，发送器与接收器之间必须是直线，不能有障碍物。卫星通信就是一种典型的微波传输方式，卫星的优点是可在一个比较大的地理区域内接收和传播信息，受地球曲率、高山和其他一些障碍的影响较小。蜂窝传输是将一个区域划分成若干小区，采用无线电波进行信号的双向传播的通信方式。移动通信技术采用蜂窝传输方式，具有使用频段广、系统扩展方便、移动性强的优势，但因为蜂窝传输使用的是无线电波，可使用特殊的接收器来窃听蜂窝电话中的通话，所以安全性难以保障。红外线传输是借助光波通过空气发送信号，是短距离传输。红外线传输可用于连接各类小设备和计算机。例如，利用红外线传输将手提计算机的数据和信息传送

给同一房间内的较大型计算机。某些特殊类型的电话也可使用红外线传输。

（4）网络操作系统。网络操作系统是网络上各计算机能方便而有效地共享网络资源，为网络用户提供所需的各种服务的软件和有关规程的集合。网络操作系统与通常的操作系统有所不同，它除了应具有通常操作系统应具有的处理器管理、存储器管理、设备管理和文件管理外，还应具备对网络的管理功能，能够提供高效、可靠的网络通信能力；提供多种网络服务功能，如远程作业录入并进行处理的服务功能、文件传输服务功能、电子邮件服务功能、远程打印服务功能等。通用操作系统如 Windows、UNIX、Linux 等都提供网络操作系统版本的支持。

（5）网络通信协议。网络通信协议是一种网络通用语言，为连接不同操作系统和不同硬件体系结构的互联网络提供通信支持。网络通信协议是控制网络中两个节点间信息传送的一套规则和程序。同一网络中的每个设备都应能够解释其他设备的协议。协议在网络通信中的主要功能有：识别网络系统中的每个设备、确保正确接收传送的信息、确定需要重新发送的不完全或错误的信息、保证系统中设备的安全以及有一定的恢复功能。OSI 模型定义了网络互联的 7 层框架（物理层、数据链路层、网络层、传输层、会话层、表示层和应用层），明确区分了服务、接口和协议三者的概念，各个层的作用如图 3-5 所示。由于 OSI 把服务层考虑得过于完备，因此通常 OSI 参考模型作为理论的模型。TCP/IP 模型由于有较少的层次，因而显得更简单，与 OSI 模型的对应关系如图 3-5 所示，并且作为从 Internet 上发展起来的协议，已经成了网络互联的实际上的标准。其他常用的网络协议包括 IBM 提出的系统网络体系结构（Systems Network Architecture，SNA）、常用于局域网的以太网以及 X.400、X.500 等。

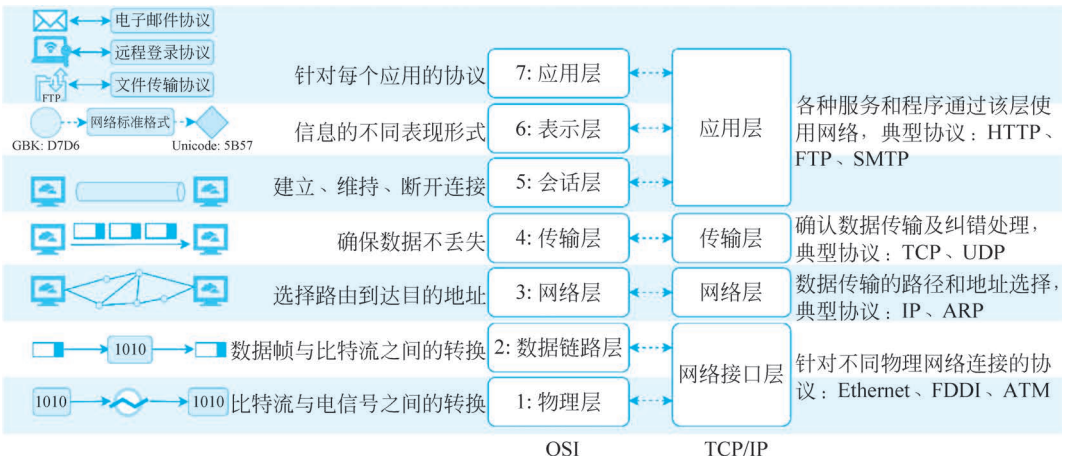


图 3-5　OSI 和 TCP/IP 对比

（6）网络工具软件。随着网络技术的发展，计算机网络工具软件在设计方面也在不断优化，网络工具软件主要包括应用级的提供网络服务功能的专用软件，大量地被应用到计算机系统中。在这个发展过程中，越来越多的网络建设者、设计者和使用者都需要工具软件辅助网络的管理和应用，尤其是对于企业、工程以及学校和银行这些领域。与

此同时，计算机网络软件的安全问题在一定程度上也关系到国家信息安全的发展和稳定。为此，在当前的社会信息体系条件下，网络软件工具软件将有着不可忽视的重要作用，常用的工具软件有网络浏览、网络监控、网络传输、电子邮件、远程控制、即时通信、电子商务、网络安全等方面的应用软件。

3.2.3 计算机网络类型

可从不同的角度对计算机网络系统进行不同的分类。这里主要从网络拓扑结构角度、网络中节点之间的物理距离以及网络提供的通信和服务角度讨论网络的分类。

1. 按网络拓扑结构分类

1）星状结构

这种结构的网络中多个节点以自己单独的链路（链路是指两个节点间的通信线路）与处理中心相连，任何两个节点间的通信都要通过中央节点来进行，如图 3-6(a)所示。一个节点在传送数据之前，首先向中央节点发出请求，要求与目标节点相连接，只有连接建立以后，该节点才能向目标节点发送数据。这种结构采用集中式访问的控制策略，所有通信均由中央节点控制，中央节点必须建立和维持许多条并行的数据通信线路。因此，中央节点的结构比较复杂，而每个节点的通信处理任务很轻。这种网络结构简单，便于管理，从终端到处理中心的时延小，但通信线路总长度长，因此花费在线路上的成本较高。中央节点的故障必然导致整个网络瘫痪。

2）总线型结构

这种网络中单个通信线路连接多个设备，如图 3-6(b)所示。主通信线路可以是双绞线、同轴电缆或光缆。所有信号在整个网络中都是双向传播的，由于没有中心主机控制网络，所以系统中必须安装特定的软件以判断各个信号的接收节点。网络中一个节点的失败不会影响网络中其他的任何节点。但是，系统中的通信信道一次只能处理一个信号，这样当传输信息量很大时系统的性能将会下降，当两个计算机同时发送信息时，就会发生冲突，必须重新发送。

3）环状结构

与总线型网络类似，这种网络没有起着中心控制作用的计算机，如图 3-6(c)所示，不会因为一个节点的故障而使整个网络停止运行。网络中的每台计算机都可直接与另一台计算机通信，各个计算机独立进行应用程序的运行。但是，在环状网络中，同轴电缆或光缆组成一个封闭的环，数据总是沿着环的一个方向进行传送。总线型和环状拓扑结构的网络是局域网中常用的结构，这两种网络中都是多个设备共享通信介质，比较节省通信线路，但是如果通道发生故障，所有设备都无法与系统中的服务器通信，另外，网络中的设备必须竞争网络介质和网络资源，如打印机、硬盘、调制解调器等。

4）树状结构

树状结构是从总线型和星状结构演变来的，像一棵倒置的树。如图 3-6(d)所示，顶端的主集线器是树根，树根以下带分支，每个分支还可带子分支。节点按层次进行连接，信息交换主要在上、下层节点之间进行，相邻及同层节点之间一般不进行数据交换。树

状拓扑结构虽有多个中心节点，但各个中心节点之间很少有信息流通。各个中心节点均能处理业务，但最上面的主节点有统管整个网络的能力。所谓统管，是指通过各级中心节点去分级管理，从这个意义上说，它是一个在分级管理基础上的集中式网络，适合进行各种管理工作。

5）网状结构

网状结构是指网络中两个节点之间不止一条连接通路，如图 3-6(e)所示。这种结构的网络往往具有较高的可靠性，并且能够保证具有较快的响应速度。

6）无线拓扑结构

通常使用无线网络时采用，如图 3-6(f)所示。初装费比采用其他结构高，但是对用户端，特别是需要经常移动的用户说，无线网络更加有效。

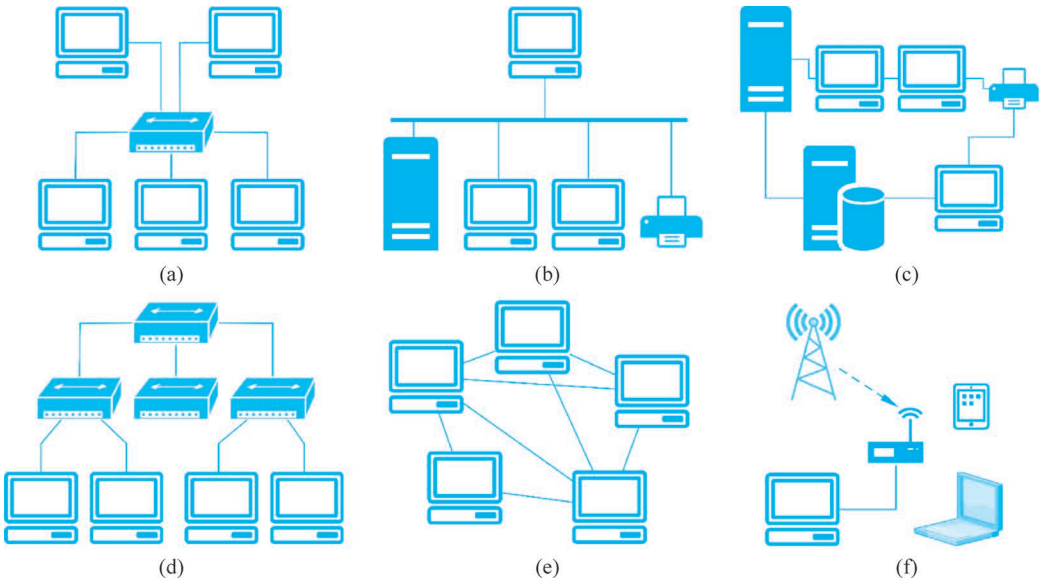


图 3-6　网络拓扑结构分类

网络拓扑结构的选择取决于可靠性、可扩充性及网络自身的特性等多种因素。总线型结构由于其价格、可靠性和可扩充性等性能比较好，因此得到较为广泛的应用。环状结构比其他网络结构具有更高的吞吐率，但可靠性较差，可以采用双环结构来解决这个问题。星状结构主要用于终端密集且网络管理集中于中央节点的场合，在这种结构中，中央节点的可靠性尤为重要。另外，可根据实际需要混合使用不同的拓扑结构。

2. 按节点间距离的网络分类

局域网(Local Area Network，LAN)又称局部网，是指在有限的地理范围内(如一个实验室、一幢大楼、一个单位)建立的计算机网络。局域网的覆盖范围一般不超过 10km。一般是私有的。这种网络的特点是：连接范围窄、用户数少、易于配置、连接速度快。目前，局域网中速度最快的是 10Gbps 以太网。IEEE 802 标准委员会定义了各种主要的 LAN 网络：以太网、令牌环、光纤分布式接口(FDDI)、异步传输模式(ATM)和无线局域网。

城域网(Metropolitan Area Network,MAN)是在一个城市范围内建立的计算机网络,它所覆盖的范围一般为10～100km。通常使用与局域网相似的技术,传输介质主要采用光纤,传输速率在100Mbps以上,既可能是私有的也可能是公用的。

广域网(Wide Area Network,WAN)又称为远程网,是一种跨越大地域的网络,它所覆盖的地理范围从几十到几千千米,可覆盖一个国家、地区,或跨越几个洲,形成国际性的远程网络。Internet国际互联网就地理范围和网络规模而言,是实现全球计算机互联互通的最大广域网。

3. 按网络提供的通信和服务分类

公用网:是多个不同企业与不同人群共同使用的网络。

专用网:为一个企业服务的网络。

增值网(Value-Added Network,VAN):是一种为企业提供增值服务的专用数据通信网络。它是由运营商提供的一种私有网络,用于在不同的地理位置间传输数据。VAN的主要功能包括数据加密、数据传输、数据管理等,为企业提供了高效、安全、可靠的数据通信服务。

虚拟专用网(Virtual Private Network,VPN):是一种基于公共网络的加密通信方式,通过在公共网络上建立加密通道,实现数据在不安全的公共网络上的安全传输。VPN的主要功能包括加密数据、隧道传输、身份认证等,为企业提供了安全、便捷、低成本的远程访问服务。

3.3 信息系统开发方法

信息系统的开发过程规模大、复杂性高,需耗费大量的人力、物力和财力等资源,因此选择一种合适的系统开发方法有着十分重要的意义。信息系统的开发方法有多种,每一种方法都有相应的模型、工具和技术,但每一种方法自身都有其特点和适用环境,没有任何一种方法或一种通用的方法适合所有的开发项目。目前常用的信息系统开发方法有结构化方法、面向对象方法、原型法、面向方面的方法和敏捷开发方法,它们构成了绝大多数开发方法的基础。当面对不同的问题时,可以有针对性地选用或参考借鉴相应的开发方法作为解决途径。

3.3.1 结构化方法

1. 产生背景

结构化,简单说就是“有组织、有规范、有规律的一种安排”。计算机科学中的结构化(structured)一词最早是作为一种程序设计技术而出现的,即结构化程序设计。结构化程序设计(structured programming)技术产生于20世纪60年代,其主要目的是提供一组约定的规则(rule)去提高程序的质量。因为一个质量好的程序不仅能在每次执行时产生正确的结果,而且应能让其他程序员方便地读懂它和修改它。在结构化程序设计技术产生之前,程序员更关心程序能否运行和结果是否正确,不关注所写的程序别人是否能看懂。

直到20世纪60年代末，由Dijkstra提出了结构化程序设计的理论，并由Bohn和Jacopini给出了证明。其基本思想是每一个程序都应按照一定的基本结构来组织，这些基本结构包括顺序结构、选择结构和循环结构，并且每一个程序都只能有一个入口和出口，如图3-7所示。结构化程序设计技术的这一简单的规程在很大程度上解决了程序可读性和可维护性差的问题，很快便成为一种事实上的工业标准，并被广大程序设计人员所接受。

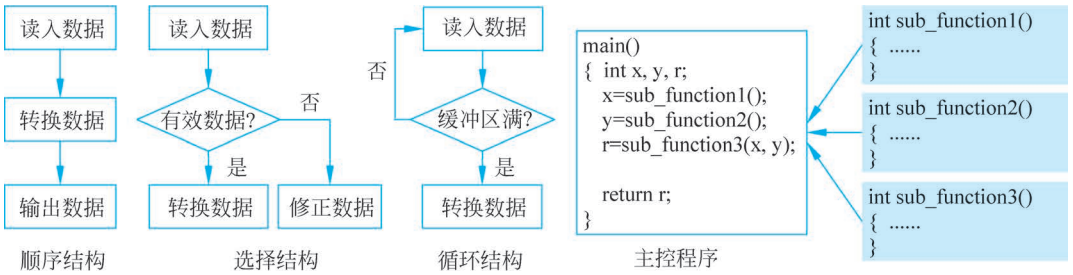


图3-7　结构化程序基本结构

20世纪70年代以来，随着计算机应用的发展，计算机程序所解决问题的复杂性越来越高，而结构化程序设计技术本身在解决大型复杂问题时，实际上采用的是一种自底向上(bottom up)的设计策略，即先设计好每一个具体的功能模块，然后再将这些设计好的程序模块组装成一个应用软件系统。显然这种解决问题的方法没有能够从全局的角度去考虑软件系统中各个功能模块之间的关系，缺乏系统总体结构的规划。所以说这样的软件系统在灵活性、可维护性、可靠性等方面都不理想。在此基础上，20世纪70年代中期，Larry Constantine在*IBM System Journal*上发表了奠基性的文章《结构化设计》，其中提出了结构化设计(structured design)的思想。结构化系统设计的目标是对一个表达清楚的问题，运用一组规范和准则指导系统开发人员首先从确定系统的总体结构着手，然后进行每一个功能模块的具体设计，选择和组织模块与模块接口，求得所述问题的最优解。这种先整体后局部、先设计后实现的策略是一种自顶向下(top down)的开发策略。

结构化设计的两个最基本的原则是高内聚(highly cohesive)、低耦合(loose coupled)。高内聚是指一个模块只完成一个明确的任务，这样不仅便于理解一个模块所实现的功能，而且一个模块本身的修改不会影响到其他模块的功能；低耦合则反映了模块间的相互依赖程度，即一个模块应尽可能地和其他模块保持相对独立，模块间的依赖性越小说明模块的独立性越强，这使得一个模块在设计和以后修改时无须涉及其他模块。工业社会最典型的高内聚、低耦合产品就是手机，手机将多种功能聚合在一起，是高内聚的；而手机的实现又是低耦合的，各种功能组件已经标准化，制造商可以从市场上选购成熟的组件进行集成。高内聚、低耦合是信息系统功能模块理想的实现效果。

结构化系统分析(structured analysis)技术的提出则是在20世纪70年代末期、80年代早期，1975年，Tom Demarco、Chris Gane、Edward Yourdon、Trish Sarson等专家提出了结构化分析的方法，以求清楚阐述要解决的问题，建立系统用户需求模型，它保证了系

统开发人员在设计系统总体结构和程序模块之前将系统的需求进一步明确化。系统设计的目标是建立在“一个表达清楚的问题”的基础上，因此结构化系统分析技术的关键便是如何以抽象的方式将求解的问题形式化地表示出来。从方法学的角度，根据方法和技术两者之间的关系不难看出，结构化方法是由结构化分析技术、结构化设计技术和结构化程序设计技术组成的。

结构化方法解决的问题和发展过程如图 3-8 所示。结构化程序通过 3 种基本结构解决程序可读性和可维护性差的问题，结构化设计方法解决模块间复杂的调用关系，从全局性角度来构建信息系统，结构化分析方法有助于更加清晰准确地表达信息系统需求。

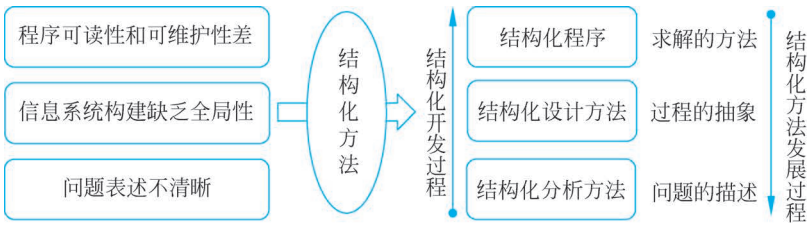


图 3-8　结构化方法解决的问题和发展过程

2. 结构化方法的特点

从开发方法学的观点看，结构化方法是用结构化分析技术、结构化设计技术以及结构化程序设计技术去开发一个信息系统的方法；从开发过程性的角度，结构化方法是分阶段实施，自顶向下、逐步求精的开发方法，是一种从具体（现实世界的物理系统）到抽象（用逻辑模型表示的系统需求）、再由抽象（由逻辑模型转换得到的物理模型）到具体（一个具体的信息系统）系统的（先整体后局部、由分析到综合）开发方法。使用结构化方法开发信息系统一般具有如下重要特征。

（1）抽象性。抽象是一种略去与某一阶段目标无关的细节的手段，抽象的目的在于描述最本质的内容。在用结构化方法开发信息系统的过程中，从结构化系统分析，到结构化系统设计，再到结构化程序设计，这期间有多个抽象级，这些抽象级是在不同层次上对同一问题的不同的抽象表示。

（2）面向过程。面向过程也可说成是过程驱动。在用结构化方法开发信息系统的过程中，始终从问题域中的业务功能（过程）这一角度考虑，而不是着重从信息（数据）的角度来考虑，数据只是作为过程的“属性”，即完成处理过程所需要的数据、所产生的数据、所需存储的数据等。数据在结构化方法中相对于过程来说只是处于“从属”地位。

（3）层次性、模块化、结构化。结构化方法体现了系统的思想，是一个“分而治之，由分而合”的过程。在求解问题的过程中，将复杂问题分解为一些较小的、比较容易理解的、相对独立的部分来求解，然后再将这些部分问题的解综合成复杂问题的解。功能的模块化、模块的层次化、程序的结构化实际上是分而治之、逐步求精思想的具体应用。

（4）逻辑独立性。逻辑独立性是结构化方法的又一重要特征，逻辑设计（系统分析）和物理设计（系统设计）分开进行，有利于开发人员更准确地抽象出系统的本质特征和功能。

3. 结构化方法的适用性分析

结构化方法以 SDLC 为基础，按工程学的原理组织和管理信息系统的开发。各阶段基本上是一种线性的顺序关系，它强调在设计阶段之前完成需求的确定，在实施之前一定要完成设计。结构化方法有如下优点。

（1）阶段的顺序性和阶段的依赖性。将系统开发过程分成若干阶段，每一阶段又分成若干个工作步骤，每一个阶段都有明确的目标和任务。上一阶段的工作成果是下一阶段的工作前提和依据，也就是说，后一阶段工作是在前一阶段工作内容的基础上进行的，后一阶段任务的完成又使前一阶段的成果在实现过程中更为具体。

（2）推迟实现。对于有一定规模和复杂程度的软件，软件危机的启示是编码越早，完成的时间反而越长，甚至会导致不可挽回的损失。结构化方法的逻辑设计和物理设计分开进行的特征，在确保系统需求正确性、一致性的基础上，大大提高了系统的可靠性。

（3）良好的文档支持。在系统开发过程中的每一阶段都必须建立相应的文档资料。文档是信息系统不可缺少的重要组成部分，不仅是一个阶段工作成果和结束的标志，也是各阶段之间、开发人员与用户之间的沟通桥梁，更是做好系统维护工作的保障手段。每个阶段对文档的复审就是对本阶段工作成果的评定，避免错误被带入下一阶段，错误被发现并纠正得越早，所造成的损失也就越少。

（4）有较多成熟的方法和工具可以使用。结构化方法采用的模型和工具如图 3-9 所示。

	结构化分析	结构化系统设计	结构化程序设计
模型	数据流程图（DFD）	模块结构图（MSC）	程序流程图
	数据字典 过程描述、结构化语言、判定树/表	伪码 系统流程图	N-S图（盒图） 问题分析图（PAD）
	实体-联系图		
结构化方法的相关工具	ER-Win	Visio	RTCASE	VS.NET

图 3-9　结构化方法采用的模型和工具

结构化方法虽然有许多优点，但该方法有两点不足是公认的。

（1）可变性差。一方面，因为该方法是一种预先定义需求的方法，也就是说，采用该方法的基本前提是必须能够在早期就冻结用户的需求，并且需求相对稳定。然而这种预先定义需求的策略，对那些随时间推移需求会调整变化或需求一时难以确定或项目参与者因种种原因而导致需求模糊、误解等情形来说，显然是不切实际的。按照这些预先指定的需求开发系统，当系统开发出来时，要么已过时而不符合当前的用户需要，要么系统存有隐患，需为此付出很高的修改代价，甚至根本不可能修改而造成负面影响。另一方面，结构化分析和结构化设计技术是围绕实现处理功能的“过程”来构造系统的，然而用户需求的变化大部分是针对功能的，因此这种变化对基于过程的设计来说是灾难性的，

用这种技术设计出的系统在结构上会不够稳定。

(2) 系统分析到系统设计的模型之间的转换不自然。结构化方法未能很好地解决系统分析与系统设计之间的衔接问题。结构化方法在这两个阶段中所用的概念、术语、模型结构等均有一定的差距和跨度,因而在阶段之间的过渡性上显得不够自然,不具有"同构性"。

由于结构化方法的这些问题,其他的信息系统开发方法便相继出现。

3.3.2 面向对象方法

1. 面向对象方法概述

面向对象(Object Oriented,OO)方法的出发点和基本原则是尽可能模拟人类习惯性的思维方式,使开发软件系统的方法和过程尽可能接近人类认识世界、解决问题的方法和过程,就是使描述问题的问题空间(或称为问题域)与实现求解问题的解空间(也称为求解域)在结构上尽可能一致。

"面向对象"是一种认识客观世界的观点,认为客观世界是由许多不同种类的对象构成的,每个对象有自己的内部状态和运动规律。不同对象之间相互联系、相互作用就构成了完整的客观世界。而信息系统是用计算机解决客观世界中的问题,从根本上来说,是借助某种程序设计语言的规则,对计算机中的实体施加某种处理,并用处理结果去映射。面向对象方法强调直接以问题域(现实世界)中的事物为中心来思考问题、认识问题,根据这些事物的本质特征,把它们抽象地表示为系统中的对象,作为系统的基本构成单位。

信息系统或软件系统中的数据及其处理是密切相关的,传统的开发方法人为地把数据和处理分离成两个独立的部分。与传统方法相反,面向对象方法把数据和处理过程相结合,把对象作为由数据及可以施加在这些数据上的操作所构成的统一体。面向对象方法中的对象与传统的数据有着本质的区别,它不是被动地等待外界对其施加操作;相反,它是操作处理的主体。只能通过发送消息(message)请求对象主动地执行某些操作,处理私有数据,而不能直接从外界对私有数据进行操作。

面向对象方法所提供的"对象"概念,是让软件开发人员自己定义或选取解空间对象,应该使得这些解空间对象与问题空间对象尽可能一致。软件系统可看作一系列离散的解空间对象的集合。这些解空间对象彼此通过发送消息而相互作用,从而得出问题的解。也就是说,面向对象方法不是把系统看作工作在数据集上的一系列过程或函数的集合,而是把系统看作既彼此独立又相互协作的一系列对象的集合,每个对象都有自己的数据、操作、功能和目的,系统是通过这些对象间的相互作用来完成任务的。

面向对象方法的核心理念是:客观世界是由各种对象构成的;信息系统中所有对象都划分为各种对象类;按照子类-父类关系,将若干对象类组成具有层次结构的系统;对象间通过消息进行相互联系和交互。例如,用户到 ATM 机取钱,采用面向对象方法可以描述为如图 3-10 所示的过程。

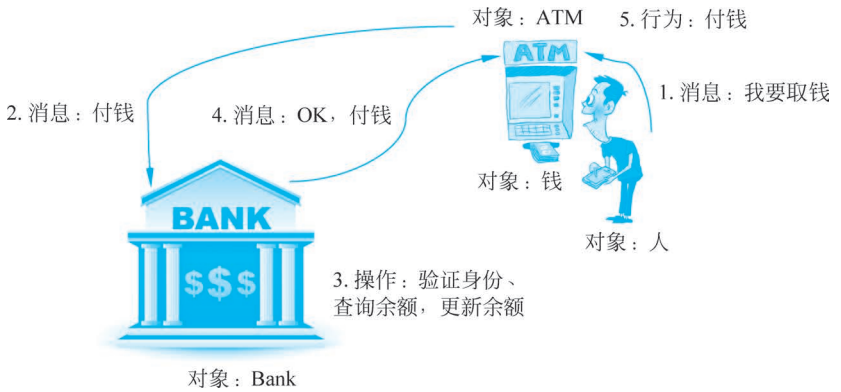


图 3-10　面向对象方法描述示例

2. 面向对象的基本概念

在应用领域中有意义的、与所有解决的问题有关系的任何事物都可以作为对象，它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有边界和意义的东西。

定义 3-1　对象是由描述该对象属性的数据以及可以对这些数据施加的所有操作封装在一起构成的统一体。

对象由属性和方法组成。

定义 3-2　属性反映事物的信息特征，如特点、值、状态等。

定义 3-3　方法用来改变属性状态的各种操作。

例如，“人”是一个对象，属性有姓名、性别、出生日期、出生地等，方法有行走、乘车、用餐、就寝等。“卫星”也可以是一个对象，属性有轨道参数、型号、载荷类型等，方法有开机、关机、测摆、变轨等。

定义 3-4　消息是对象之间建立的一种通信机制，它统一了数据流和控制流。

对象之间的联系主要是通过消息来实现的，消息传递的方式是通过消息模式和方法所定义的操作来实现。当一个消息发送给某个对象时，包含要求接收对象去执行某些活动的信息。接收到消息的对象经过解释，然后予以响应。传递消息的对象称为发送者，接收消息的对象称为接收者。发送者不需要知道接收者如何对请求予以响应。接收者响应消息的过程是先选择符合消息要求的操作，然后执行该操作，最后将控制权返回调用者。

通常一个消息由 3 部分组成：接收消息的对象；消息标识符（即消息名）；零个或多个变元。访问一个方法的过程称为向这个对象发送一个消息。

例如，MyCircle. Show(Green)，其中，MyCircle 是接收消息的对象的名字，Show 是消息名，Green 是消息的变元。

定义 3-5　对象类是具有相同数据结构和相同方法的一组相似对象的抽象，简称类，表示某些对象在属性和操作方面的共同特征。

定义 3-6 类实例是指由某个特定的类所描述的一个具体的对象，类实例可简称为实例。

例如，“东风 湘 A F3207”是类“卡车”的实例；“风云一号”是类“气象卫星”的实例。

定义 3-7 继承是指能够直接获得已有的性质和特征，而不必重复定义它们。

在面向对象技术中，继承是子类自动地共享基类中定义的数据和方法的机制。继承使得相似对象可以共享程序代码和数据结构，提高开发效率，使程序结构清晰。

例如，卡车类就是继承汽车类的子类，除了具有汽车类的属性外，还具有卸货和载重量属性，如图 3-11 所示。

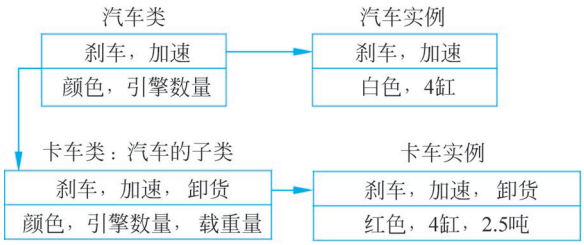


图 3-11 继承示例

定义 3-8 多态指在类层次结构中，不同层次上的类可以共享一个方法的名称，但可以按照各自的方式来实现这个方法。

多态使得一种操作可以被不同的对象/类以不同的方式执行。例如，多边形类中的“计算边数”操作，三角形和矩形都是调用 getSides()方法，但是三角形返回结果 3，矩形返回结果 4。

面向对象方法中重载的概念与多态不同，有两种类型的重载：函数重载是指在同一作用域内的若干参数特征不同的方法可以使用相同的函数名称；运算符重载是指同一运算符可以施加于不同类型的操作数上。当然，当参数特征不同或操作数的类型不同时，实现方法或运算符的语义是不同的。重载进一步提高了面向对象系统的灵活性和可读性。

3. 面向对象的技术要点

首先，面向对象方法的优势在于抽象能力。面向对象方法采用对象来表达一切事情，将其静态属性和动态行为抽象为数据结构以及在数据结构上所施加的一组操作，并把它们封装成一个统一体，使对象状态变成对象属性值的集合，对象行为变成改变对象状态的操作方法的集合。在对象抽象的基础上，面向对象方法进一步提出了对象类这一独特的概念，对象类实现了更高一级的抽象，它把具有相同或共性语义特性（即数据结构特性和操作特性及其有关约束特性）的一组对象组成为对象类，将数据结构上的抽象与功能上的抽象结合起来，并加以统一说明，实现了传统方法所不具备的更高级的抽象。正是由于对象这种广泛的、高度的抽象表达能力，使该方法具有很强的建模能力。

其次，面向对象的封装是保证软件部件具有优良的模块性的基础。所谓封装，是指所有软件部件的内部有明确的范围以及清楚的外部边界，且每个软件部件都具有友好的

外部接口，用于说明各部件之间的相互作用和相互关系，同时应当完全保护软件的内部实现，使用户不必了解如何具体实现。这样，封装一方面有利于用户集中精力去考虑所开发的系统、各模块之间的关系等重大问题；另一方面也有利于编程人员对软件部件进行精心雕琢，确保模块质量的可靠性。

再次，面向对象技术提供几种不同层次的共享。数据结构和行为的继承允许在若干个相似子类间共享公共结构。面向对象技术不仅允许在某个具体应用范围内支持共享，在更通用的普遍化的设计中，它还提供了未来项目中可重用设计的可能，面向对象管理组织(OMG)已提供了这方面的标准和工具，诸如用抽象、封装和继承来建立可重用组件库。

最后，面向对象技术关注对象是什么，而不是如何使用对象的过程，也就是说，在面向对象的设计方法中，计算机的观点不是重要的，最重要的是现实世界的模型。在开发期间，一个对象的使用高度依赖于应用的细节和变化的频率，由于需求在变化，用一个对象提供的特征比它的使用方式更加稳定，建立在对象结构上的软件系统在长时间运作中也会更为稳定。传统的面向过程是基于控制的过程调用机制，系统按功能划分模块，功能的抽象即为过程，是一种将重点放在过程结构上的开发方法，当系统功能需求发生变化时将引起软件结构的整体修改。面向对象的系统是基于消息机制的对象间的相互作用，是一种将重点放在对象结构上的开发方法。

4. 面向对象方法的适用性分析

通过对面向对象技术要点的分析，可以看出，面向对象方法具有以下优点：

(1) 与人类思维方法一致。面向对象方法强有力的抽象机制，体现了从特殊到一般的归纳思维过程；通过建立类的层次结构而获得的继承特征，则体现了从一般到特殊的演绎思维过程。通过先设计出抽象类来构成系统框架，随着认识深入和具体化再逐步形成更具体的派生类。这样的开发过程符合人们认识客观世界解决复杂问题时逐步深化的渐进过程。

(2) 可重用性好。面向对象方法为软件重用提供了新的手段，基于类和对象的重用显得比传统方法更为容易。一方面，对象类本身就是一种比较理想的重用“部件”；另一方面，继承机制使得父类的数据结构和程序代码可以被子类重用，或者派生新类。面向对象方法不同粒度的重用如图 3-12 所示。

图 3-12　面向对象方法不同粒度的重用

(3) 稳定性好。类的独立性强，修改一个类通常很少会涉及其他类。若仅修改一个类的内部实现部分而不改变该类的外部接口，则软件系统的其他部分完全不受影响。同

时，继承机制使得系统扩展方便，只需从已有类派生出一些新类而无须修改系统的原有部分。再者，多态机制使得扩展软件功能时很容易重用原有代码。

（4）可维护性好。面向对象的开发方法符合人们习惯的思维方式，软件系统更易于阅读和理解，这就使得在软件维护过程中易于测试和调试。

面向对象方法与结构化方法的对比如表 3-1 所示。

表 3-1　面向对象方法与结构化方法的对比

	结构化方法	面向对象方法
系统模型	过程、分析和设计分离	对象、分析即设计
系统实现	程序＝算法＋数据结构	对象＝算法＋数据结构 程序＝对象＋方法＋消息
设计思想	自顶向下、逐步求精	由小而大、自底向上
稳定	过程建模，不适应动态变化	对象建模，相对较为稳定
复用	代码级复用，复用支持差，缺乏机制和标准，分析和设计复用困难	继承、多态、重载等机制就是用于复用

3.3.3　原型法

1. 原型法产生的背景

导致信息系统开发过程缓慢的原因很多，影响大多数项目进度的因素主要包括需求的变动、返工以及开发工具的不足或不正确。

造成许多系统开发延期的原因之一是在开发过程中需求发生改变。需求的变动要求设计和构造工作也要进行相应的变化。从软件工程的角度来看，需求变动发生得越晚，实现需求变动所付出的代价就越大。以房屋装修为例，在正式施工前的房屋设计的改动（如修改水电管线走向、增加电源等），其代价就相当小，而在施工之后的改动，就需要重新绘制蓝图并修改物料清单和建设进度，甚至是返工。软件需求的变动会造成开发成本的增加，大致的估算方法如图 3-13 所示，如果项目规划阶段的改动需要成本 1 美元，那么系统分析阶段的改动需要成本为 10 美元，系统设计阶段的改动需要成本为 100 美元，详细设计阶段的改动需要成本为 1000 美元，系统实施期间的改动需要成本为 10 000 美元。

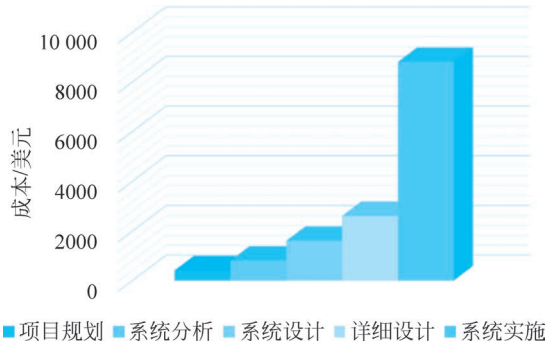


图 3-13　生命周期各个阶段的需求变动所需的代价

因此，一种新的信息系统开发思路是用较小代价构造一个工作演示模型，并与需求对照，及早发现所设计系统的缺陷和难点。

2. 原型法的思想

原型(prototype)是指用来模拟客体的原始模型，其结构、大小、功能等都与客体类似，而在信息系统开发中，则用“原型”来形象地表示一个系统的早期的可运行版本或模型。原型法从常规的程序设计方法和呆板的“瀑布”模型开发模式，发展到一种快速、灵活、交互式的软件开发方法学，这种方法通常能够证明所建立起来的应用系统是符合用户需要的。原型法利用交互、快速建立起来的原型取代了形式化的需求规格说明，用户通过在计算机上实际运行和试用原型系统而向开发人员提供真实、客观的反馈意见或建议。分析设计人员通过构建和运行原型，向用户展示系统的功能，并获取用户的反馈，从而快速实现对系统的修正。原型法是一种基于迭代(iterative)的开发模式，也是目前比较流行和实用的一种开发方法。常用的原型设计工具有 Axure RP、Mockplus、POP(Prototyping on Paper)、Proto.io 等。

原型化是构建一个可以模仿真实系统的部分或者全部功能的系统模型的过程。原型化是一个建立原型的过程，而原型则是该过程的结果。一个原型是一个自我独立的系统，其过程如图 3-14 所示。对原型化这一过程，一般包括两种主要途径：抛弃式原型和演化式原型。

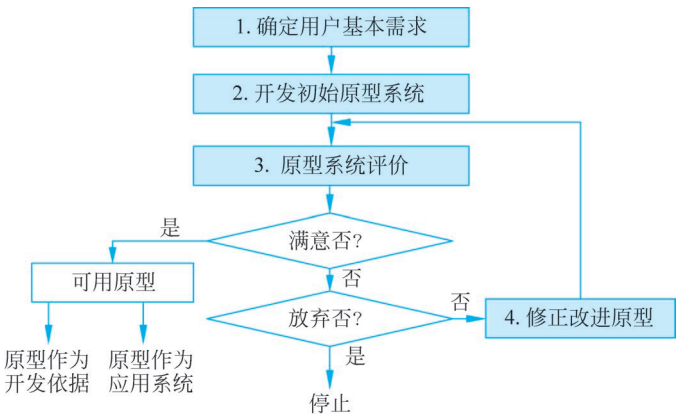


图 3-14　原型法的主要工作过程

抛弃式原型一旦目的达到就被抛弃，原型不作为最终产品。首先，快速构造，或引用一个初始原型给用户体验，并就系统的各方面展开讨论。然后，根据讨论结果，构造符合用户要求的系统，而原来的原型被抛弃。抛弃式原型的实质是作为用户与开发人员之间进行通信的媒介，并不打算把它作为实际系统运行。目的是对最终系统进行研究，使用户和开发人员借助这个原型进行交流，共同明确新系统的需求。使用这种方法，原型开发过程可以作为传统 SDLC 的需求定义阶段，用于确定需求。其特点是需要利用软件工具和开发环境，开发费用低、速度快。

演化式原型系统的形成和发展是逐步完成的，每次迭代都要对系统重新进行规格说

明、设计、实现和评价。首先也是快速构造一个功能较简单的初始原型给用户体验，并就系统的各方面展开讨论。然后，逐步完善各项功能，最终形成完整系统。由于用户的要求及系统的功能可能随时都在发生变化，所以可以先构造系统，出现问题随时修改。系统开始可以完成一项或几项任务，随着用户的使用，进一步对系统进行修改。系统功能的修改在演化式原型中十分频繁。其特点是最终系统能很好满足用户需求，但必须加强管理和控制，围绕系统的初始需求进行。

3. 原型法适用性分析

原型化方法主要有如下优点。

(1) 便于需求定义。有了原型的概念，用户在系统开发过程中起主导作用，陈述需求时更加直观、简单和具体，对一些动态需求或不易于用简单的语言文字、图表规范来辨认和描述的需求更为明显。

(2) 系统可靠性好。因为原型法让用户自始至终有机会参与整个开发过程，用户、开发人员以及原型系统之间可直接交互，所以需求确定、有效性好、可操作性强。

(3) 系统开发效率高、风险小、费用低。原型法运用了“迭代模型”的原型技术以及其他大量的辅助技术和工具，不仅使系统分析、设计和实施的时间大为缩短，还减少了开发人员对用户需求的误解，从而降低了系统的开发风险，减少了开发费用，提高了劳动生产率。

但原型法也有其不足的地方。首先表现为系统分析和设计的深度不够，系统是在逐步补充和细化中完善的，缺乏整体性，从而易导致系统的局部性能优而整体性能差，以及不易于扩充和维护；其次系统开发过程不易于管理：原型法 SDLC 的阶段性不够明显，系统的开发进程不如结构化方法那样易于组织、管理和控制，而且缺乏相应的文档资料；再者原型法要求训练有素的、有经验的开发人员参加且必须有一些自动化的高效辅助工具和开发环境作支持。

因此，原型法适用的场景包括：一部分需求不能单独确定；一些系统功能的技术可行性不可知或不确定；开发工具有足够能力去构造一个功能完备的原型系统；不适用的场景包括：没有交互性的信息系统(如资源动态监控的程序)；功能内部复杂(如利用复杂算法规划多模式最优路径)；有严格的性能要求和安全需要(如每小时处理上万个信息检索的程序)。

3.3.4 面向方面的方法

面向方面的(Aspect Oriented Programming，AOP)方法是面向对象方法的补充和完善。面向对象方法引入封装、继承、多态等概念来建立一种对象层次结构，用于模拟公共行为的一个集合。不过面向对象方面允许开发者定义纵向的关系，但并不适合定义横向的关系。如图 3-15 所示，在应用系统中的日志记录、权限验证、事务控制等功能，有可能横向地散布在所有对象层次中，而又与对应的对象的核心功能并无关系，这种散布在各处的无关的代码称为横切(cross cutting)，在面向对象设计中，它导致了大量代码的重复，而不利于各个模块的重用。

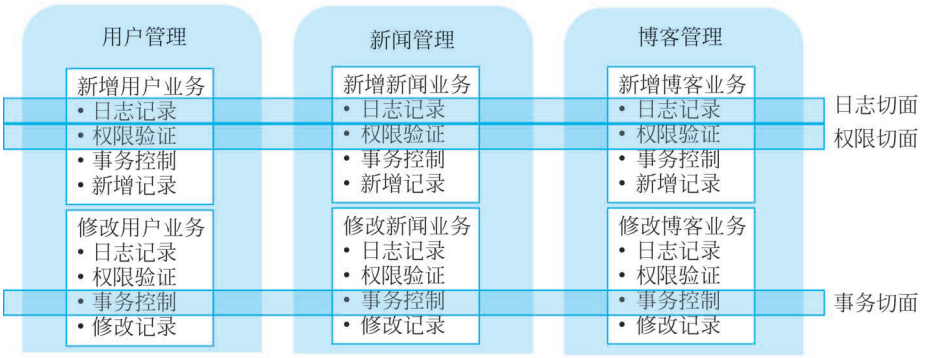


图 3-15　应用系统中的横切面

AOP 方法则利用“横切”，剖解开封装的对象内部，并将那些影响了多个类的公共行为封装到一个可重用模块，并将其命名为“方面”。方面就是指将那些与业务无关，却为业务模块所共同调用的逻辑或责任封装起来，便于减少系统的重复代码，降低模块之间的耦合度，并有利于未来的可操作性和可维护性。

AOP 把软件系统分为两个部分：核心关注点和横切关注点。业务处理的主要流程是核心关注点，如图 3-15 中的用户管理、新闻管理、博客管理等，与之关系不大的部分是横切关注点。横切关注点的一个特点是：它们经常发生在核心关注点的多处，而各处基本相似，比如权限验证、日志记录、事务控制。

AOP 的核心内容是分离关注点，是思考和构建软件系统的重要方法。在 AOP 中将关注点划分为各自独立的关注点，要求程序中的每方面（类、方法、过程等）只为实现一个目的，进而降低修改和复用方面的成本，甚至不用思考关注点之间的相互影响。当用关注点来表示一个需求或者一组需求的时候，可以很容易地在组件中跟踪需求。如果需求发生改变，那么研发人员可以快速定位到需要修改的代码，并且不需要考虑方面之间的相互影响，从而快速实现需求改变。AOP 的开发过程如图 3-16 所示。

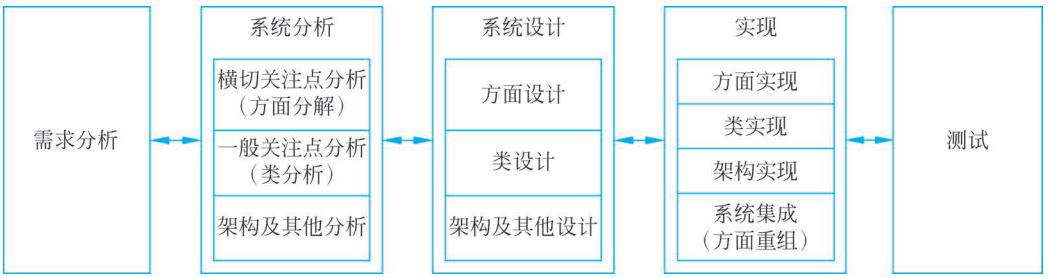


图 3-16　AOP 的开发过程

3.3.5　敏捷开发方法

1. 敏捷开发方法概述

敏捷开发方法是以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经

过测试，具备可视、可集成和可运行使用的特征。换言之，就是把一个大项目分为多个相互联系，但可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

敏捷开发采用的是迭代式开发，迭代是指把一个复杂且开发周期很长的开发任务分解为很多小周期可完成的任务，这样的一个周期就是一次迭代的过程；同时每一次迭代都可以生产或开发出一个可以交付的软件产品。与传统方法按照 SDLC 瀑布式推进和原型法少量迭代不同，敏捷开发方法的迭代会多次频繁进行，如图 3-17 所示。

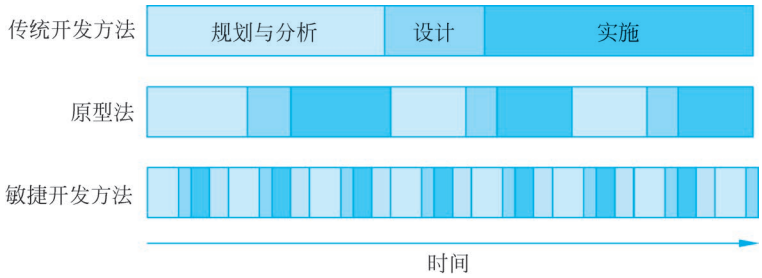


图 3-17　敏捷开发方法与其他开发方法的区别

在敏捷开发方法中，有的专注于管理工作流程，如 Scrum；有的专注于实践，如极限编程；有的则涵盖整个开发生命周期，如持续集成方法(Dev&Ops)。

2. Scrum

1995 年，美国的 Jeff Sutherlan 和 Ken Schwaber 提出了 Scrum 的概念，成为敏捷开发的一种典型实践，这种软件开发过程以英式橄榄球争球队形(Scrum)为名，因此可以想象，整个团队是高效而富有激情的。Scrum 开发特别强调沟通，要求团队所有人员都在一起工作，通过高效的沟通解决问题。敏捷开发的主要过程如图 3-18 所示。

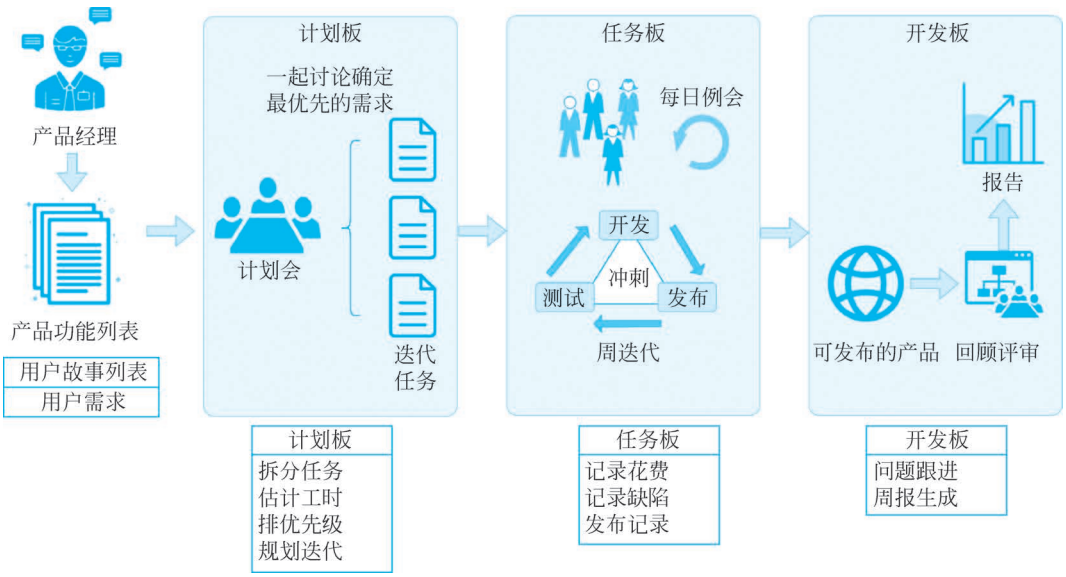


图 3-18　敏捷开发的主要过程

第一步：需求规划。确定完成产品需要做的事情。以用户故事描述的所有需求都到达产品经理(Product Owner,PO)这里，整理形成产品需求池，每次的迭代开发都是产品经理从产品需求池里挑出需要开发的部分需求。

第二步：每次迭代开发的需求集合。决定当前迭代(冲刺)需要解决的事情。团队一起开计划会，讨论确定最优先的需求及交付日期。

第三步：冲刺(sprint)。接下来利用2～4周的时间进行开发和测试，冲刺期间要开每日立会(scrum meeting)，每日立会是指站着开会，一般不超过15分钟，只描述状态和任务，不讨论技术细节，这样既可以保持高效，也使得团队中每个人都应该了解其他人在做的事情，以及当前团队的进展和风险。

第四步：交付评审。最终交付后，团队一起开迭代评审会议和迭代回顾会议。得到软件的一个增量版本，发布给用户，展现本次迭代的功能增量，全体成员一起回顾此次迭代做得好的地方，以及需要改进的地方，并对这些需要改进的点提出改进措施。然后在此基础上进一步计划增量的新功能和改进。

3. 极限编程

极限编程(eXtreme Programming,XP)由Kent Beck在1996年提出。XP是一个轻量级的、灵巧的、近螺旋式的开发方法，它将复杂的开发过程分解为一个个相对比较简单的小周期。通过积极的交流、反馈以及其他一系列的方法，开发人员和客户可以了解开发进度、变化、待解决的问题和潜在的困难等，并根据实际情况及时地调整开发过程。

极限编程的主要思想有以下几点。

- 连续自动测试。包括开始写程序之前先写单元测试；做任何的代码修改、复核、整合，都要运行单元测试；集成测试、负荷测试和系统测试等。
- 用户积极参与。用户也是开发队伍中的一员。每发布一个系统(经过一个开发周期)，系统都应可用，并实现所有计划需求。
- 团体系统实现。采用结对编程(Pair Programming)，任何人都可以修改其他人写的程序，修改后要确定通过单元测试。

XP方法裁剪了“臃肿”的长期分析和设计阶段，通过连续测试和及时系统修正来抵消由于不严格的快速分析和设计所带来的消极影响，快速开发和及时交付能够激发开发团队的工作热情。这种方式对于小型项目的开发很高效，但是对于大型项目，随着规模增大效率有可能降低。由于不太关注系统分析和体系结构设计，因此XP开发的系统在优化方面较弱。

4. 持续集成方法

持续集成(Continuous Integration,CI)是一组信息系统开发过程、方法和系统的总称。CI也是一种敏捷开发方法的软件开发实践，指团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，这意味着每天可能会发生多次集成。每次集成都通过自动化的构建(包括编译、发布、自动化测试)来验证，从而尽快地发现集成错误。

CI包括3个核心概念：持续集成、持续交付(Continuous Delivery)和持续部署(Continuous Deployment)。

持续集成强调开发人员提交了新代码之后，立刻进行构建和(单元)测试。根据测试结果，来确定新代码和原有代码能正确地集成在一起。持续交付在持续集成的基础上，将集成后的代码部署到更贴近真实的运行环境(类生产环境)中。比如，完成单元测试后，可以把代码部署到连接数据库的环境中进行更多的测试。如果代码没有问题，则可以继续手动部署到生产环境中。持续部署是持续交付的下一步，指代码在任何时刻都是可部署的，最后将部署到生产环境的过程自动化。

持续集成需要用到很多软件工具的支持，比如CI引擎Jenkins、静态代码分析工具Sonarqube、单元测试工具Junit、软件镜像构建工具Docker、项目进度和任务管理工具JIRA、软件版本管理工具GitHub等。因此，CI方法对项目开发和管理人员工具使用要求较高。

CI主要过程如图3-19所示。

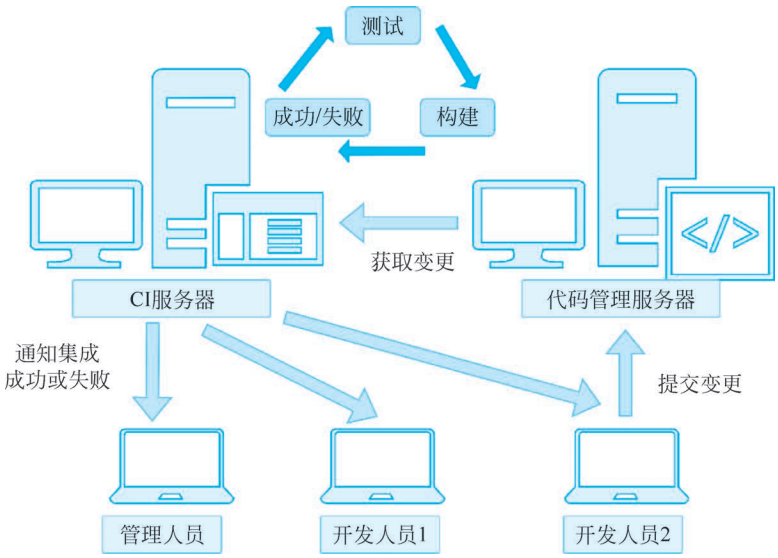


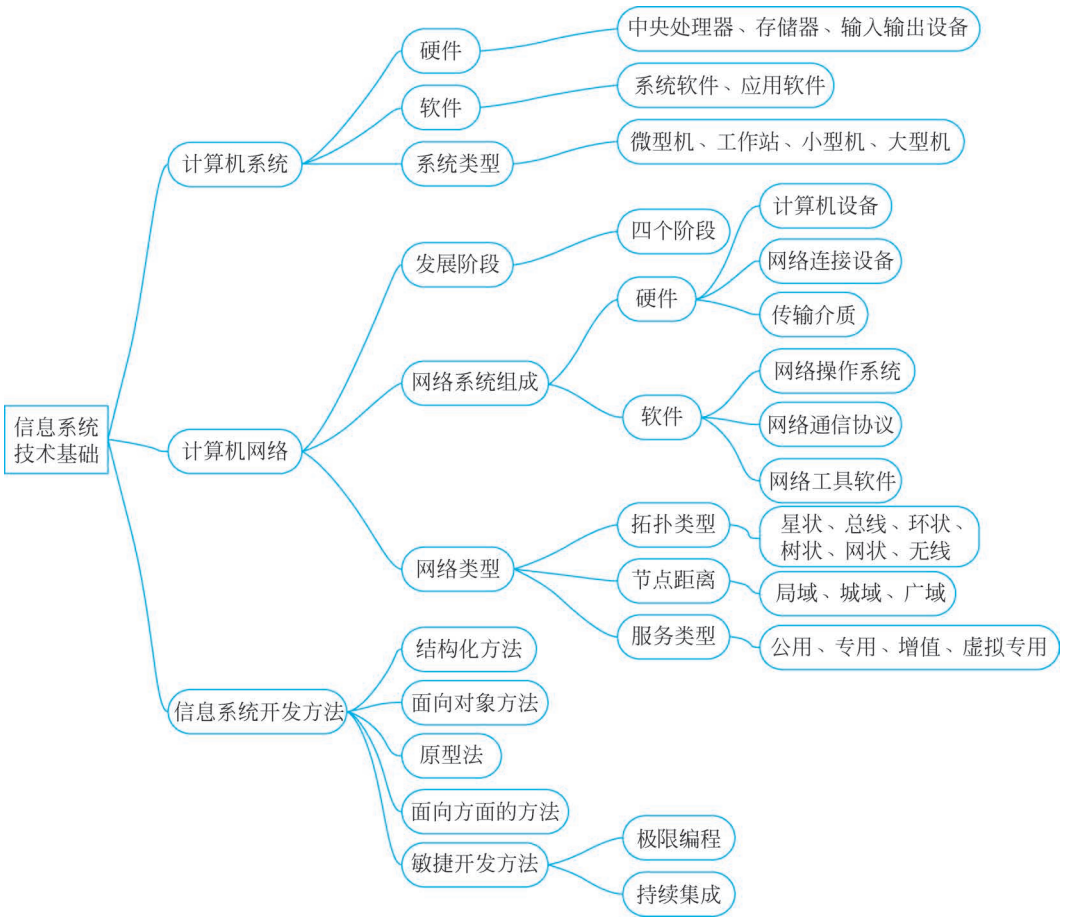
图3-19 CI主要过程

5. 敏捷开发方法的适用性分析

由于敏捷开发所具有的迭代式过程、增量交付、及时反馈、持续集成、自我管理等特点，一般适合的场景包括：开发小组人数少于10人；开发人员非常熟练，具备建模、工具使用、测试、交流和系统实现的广泛能力；所开发系统是相对独立的系统、新系统或者是与现有系统之间接口很少的系统。

敏捷开发不适用于大量开发人员参与的大型系统开发，或者是对旧系统进行改造的项目，而且旧系统并非当前团队所开发。

本章思维导图



习题 3

1. 按照计算机系统规模的大小，一般分为哪几种类型？

2. 说明为何要采用多级层次结构来构成存储系统？

3. 计算机网络发展经过了哪几个阶段？

4. 信息通信部门进行信息化升级，需要将原来独立的房间通过网络连接起来，需要采购的网络设备是什么？结合实际应用场景说明最好采用什么样的网络拓扑结构？

5. 结合 OSI 模型和 TCP/IP 模型，简述数据是如何在网络中转换为信号传输的。TCP/IP 是通过哪些协议来实现这个过程的？

6. 信息系统有哪些开发方法？简单描述每种方法的适用场景。

7. “结构化设计就是结构化程序设计的简称”，这种说法对吗？为什么？

8. 原型化方法是在什么情况下产生的？为什么这种方法会成为一种实用的系统开

发方法？

9. 面向对象方法有什么优势？

10. 与同学探讨一个最近由于开发缓慢而被迫放弃的项目。这个项目采用了什么样的开发方法？一个不同的开发方法可以使开发进程加快吗？

11. 收集一些在网络上或者校园里招聘信息工程专业毕业生的公司的信息，总结一下目前公司用来开发系统的有关开发方法的资料，招聘广告中是否涉及对 SDLC 的描述？提到了哪些 CASE 工具？

12. 简述极限编程的主要思想。