

全国计算机技术与软件专业资格（水平）考试

系统架构设计师

wen老师系统架构设计师课程内部辅导资

料 **一本通精华知识点**

目录

一	计算机硬件	9
1	校验码	9
1.1	奇偶校验码	9
1.2	循环冗余校验码 CRC	9
2	计算机硬件和指令	10
3	指令系统 CISC 和 RISC	11
4	指令的流水处理	12
4.1	流水线原理	12
4.2	流水线时间计算	12
5	存储系统	13
5.1	计算机存储结构层次图	13
5.2	局部性原理	14
5.3	高速缓存 Cache	14
5.4	虚拟存储器	15
5.5	磁盘	15
6	输入输出技术	15
7	总线	16
二	操作系统知识	17
1	操作系统概述	17
2	进程管理	18
2.1	进程的组成和状态	18
2.2	前趋图和进程资源图	19
2.3	进程间的同步与互斥	20
2.4	信号量操作	20
2.5	进程调度	21
2.6	死锁问题	22
2.7	线程	22
3	存储管理	22
3.1	分区存储管理	23

3.2	分页存储管理	23
3.3	段式存储管理	25
3.4	段页式存储管理	26
4	设备管理	26
4.1	概述	26
4.2	I/O 软件	27
4.3	虚设备和 SPOOLING 技术	27
5	文件管理	28
5.1	概述	28
5.2	索引文件结构	29
5.3	文件目录	29
5.4	文件存储空间管理	30
三	数据库系统	31
1	三级模式—两级映像	31
2	数据库的设计	32
3	E-R 模型	33
4	关系代数运算	34
5	关系数据库的规范化	36
5.1	函数依赖	36
5.2	键和约束	36
5.3	范式	37
5.4	模式分解	40
5.5	并发控制基本概念图	43
5.6	事务管理	43
5.7	封锁协议	44
6	数据故障与备份	46
6.1	安全措施	46
6.2	数据故障	46
6.3	数据备份	46
7	分布式数据库	47

8	数据仓库与数据挖掘	48
9	反规范化技术	49
10	SQL 语言	49
11	NoSQL 数据库	50
四	嵌入式技术	51
1	第二版教材 2.4 嵌入式内容补充	51
2	嵌入式微处理器	54
2.1	嵌入式微处理器体系结构	54
2.2	嵌入式微处理器分类	55
2.3	多核处理器结构	56
3	嵌入式软件与操作系统	57
3.1	嵌入式软件基础	57
3.2	嵌入式操作系统	59
3.3	嵌入式实时操作系统	59
4	嵌入式软件设计	60
4.1	开发流程	60
4.2	开发工具	61
五	计算机网络	62
1	网络概述和模型	62
2	传输介质	67
3	通信方式和交换方式	69
4	IP 地址	71
4.1	分类地址格式	71
4.2	子网划分	72
4.3	无分类编址	73
5	IPv6	73
6	网络规划和设计	74
6.1	层次化局域网模型	74
6.2	建筑物综合布线系统 PDS	74
7	网络管理命令	75
8	网络存储技术	75

	8.1 廉价磁盘冗余阵列	75
	8.2 网络存储	77
	9 其他考点汇总	78
六	其他计算机系统基础知识	79
	1 计算机语言	79
	2 多媒体	80
	3 系统工程	82
七	系统配置与性能评价	85
	1 性能指标	85
	2 性能评价方法	87
	3 阿姆达尔解决方案	87
八	信息系统基础知识	88
	1 信息系统概述	88
	2 业务处理系统（TPS）	92
	3 管理信息系统（MIS）	93
	4 决策支持系统（DSS）	94
	5 专家系统（ES）	96
	6 办公自动化系统（OAS）	97
	7 企业资源规划（ERP）	98
	8 典型信息系统架构模型	100
	9 信息化战略体系	101
	10 信息系统战略规划	103
	11 企业信息化与电子商务	103
	11.1 客户关系管理 CRM	103
	11.2 供应链管理 SCM	104
	11.3 企业应用集成 EAI	105
	11.4 电子商务	107
九	系统安全	107
	1 信息安全基础知识	107
	2 信息安全系统的组成框架	108
	3 信息安全技术	109

4	信息安全的抗攻击技术	111
5	信息安全的保证体系与评估方法	115
6	网络安全技术	117
7	网络安全协议	118
十	软件工程基础知识	119
1	软件工程	119
1.1	软件过程模型	120
1.2	能力成熟度模型	123
1.3	逆向工程	125
2	需求工程	125
2.1	软件需求	125
2.2	需求获取	126
2.3	需求分析	127
2.4	需求定义	130
2.5	需求验证	130
2.6	需求管理	131
3	系统设计	132
3.1	处理流程设计	132
3.2	系统设计	133
3.3	人机界面设计	135
4	测试基础知识	135
4.1	测试基础	135
4.2	测试阶段	136
4.3	测试模型	137
4.4	测试用例设计	139
4.5	调试	141
5	系统运行与维护	142
5.1	系统转换	142
5.2	系统维护概述	143
6	净室软件工程	143

	7 基于构件的软件工程	144
十一	面向对象技术	145
	1 面向对象基础	145
	2 UML	148
	2.1 事物	148
	2.2 关系	148
	2.3 图	149
	3 SysML	155
十二	项目管理	162
	1 进度管理	162
	2 软件配置管理	165
	3 质量管理	166
	4 风险管理	167
十三	系统架构设计	168
	1 软件架构的概念	168
	2 基于架构的软件开发方法	170
	3 软件架构风格	172
	3.1 基本架构风格	172
	3.2 层次结构风格	176
	3.3 面向服务的架构 SOA	179
	4 软件架构复用	181
	5 特定领域软件架构 DSSA	182
	6 系统质量属性与架构评估	183
	6.1 软件系统质量属性	183
	6.2 系统架构评估	185
	6.3 基于场景的评估方法	186
	7 中间件技术	189
十四	软件可靠性基础知识	192
	1 软件可靠性基本概念	192
	2 软件可靠性建模	194
	3 软件可靠性管理	195

	4 软件可靠性设计	196
	5 软件可靠性测试与评价	199
十五	软件架构的演化和维护	200
	1 软件架构演化和定义	200
	2 面向对象软件架构演化	200
	3 软件架构演化方式的分类	201
	4 软件架构演化原则	203
	5 软件架构演化评估方法	204
	6 大型网站系统架构演化实例	205
	7 软件架构维护	206
十六	未来信息综合技术	206
	1 信息物理系统技术	206
	2 人工智能技术	208
	3 机器人技术	209
	4 边缘计算	210
	5 数字孪生体技术	212
	6 云计算和大数据技术	212
十七	补充：知识产权和标准化	214
	1 知识产权基础知识	214
	2 标准化基础知识	216
十八	补充：数学与经济管理	216
	1 图论应用	217
	1.1 最小生成树	217
	1.2 最短路径	217
	1.3 网络与最大流量	217
	2 运筹方法	217
	2.1 线性规划	217
	2.2 动态规划	218
	2.3 伏格尔法	218
	2.4 博弈论	219
	2.5 状态转移矩阵	219

2.6	排队论	219
2.7	决策论	219
3	数学建模	220

一 计算机硬件

1 校验码

码距：就单个编码 A: 00 而言，其码距为 1，因为其只需要改变一位就变成另一个编码。在两个编码中，从 A 码到 B 码转换所需要改变的位数称为码距，如 A: 00 要转换为 B: 11，码距为 2。一般来说，码距越大，越利于纠错和检错。

1.1 奇偶校验码

在编码中增加 1 位校验位来使编码中 1 的个数为奇数（奇校验）或者偶数（偶校验），从而使码距变为 2。奇校验可以检测编码中奇数个数据位出错，即当合法编码中的奇数位发生了错误时，即编码中的 1 变成 0 或者 0 变成 1，则该编码中 1 的个数的奇偶性就发生了变化，从而检查出错误。但无法纠错。

1.2 循环冗余校验码 CRC

CRC 只能检错，不能纠错，其原理是找出一个能整除多项式的编码，因此首先要将原始报文除以多项式，将所得的余数作为校验位加在原始报文之后，作为发送数据发给接收方。

使用 CRC 编码，需要先约定一个生成多项式 $G(x)$ 。生成多项式的最高位和最低位必须是 1。假设原始信息有 m 位，则对应多项式 $M(x)$ 。生成校验码思想就是在原始信息位后追加若干校验位，使得追加的信息能被 $G(x)$ 整除。接收方接收到带校验位的信息，然后用 $G(x)$ 整除。余数为 0，则没有错误；反之则发生错误。

例：假设原始信息串为 10110，CRC 的生成多项式为 $G(x)=x^4+x+1$ ，求 CRC 校验码。

(1) 在原始信息位后面添 0，假设生成多项式的阶为 r ，则在原始信息位后添加 r 个 0，本题中， $G(x)$ 阶为 4，则在原始信息串后加 4 个 0，得到的新串为 101100000，作为被除数。

(2) 由多项式得到除数，多项中 x 的幂指数存在的位置 1，不存在的位置 0。本题中， x 的幂指数为 0,1,4 的变量都存在，而幂指数为 2,3 的不存在，因此得到串 10011。

(3) 生成 CRC 校验码，将前两步得出的被除数和除数进行模 2 除法运算（即不进位也不借位的除法运算）。除法过程如下图所示。

$$\begin{array}{r}
 10011 \overline{) 101100000} \\
 \underline{10011} \\
 10100 \\
 \underline{10011} \\
 11100 \\
 \underline{10011} \\
 1111
 \end{array}$$

得到余数 1111。

注意：余数不足 r ，则余数左边用若干个 0 补齐。如求得余数为 11， $r=4$ ，则补两个 0 得到 0011。

(4) 生成最终发送信息串，将余数添加到原始信息后。上例中，原始信息为 10110，添加余数 1111 后，结果为 10110 1111。发送方将此数据发送给接收方。

(5) 接收方进行校验。接收方的 CRC 校验过程与生成过程类似，接收方接收了带校验和的帧后，用多项式 $G(x)$ 来除。余数为 0，则表示信息无错；否则要求发送方进行重传。

注意：收发信息双方需使用相同的生成多项式。

2 计算机硬件和指令

计算机的硬件基本系统由五大部分组成：运算器、控制器、存储器、输入设备（如鼠标键盘）、输出设备（如显示器）。

其中，运算器和控制器合并称为中央处理单元，即 CPU；存储器分为内部存储器（即内存，容量小，速度快，临时存放数据）和外部存储器（即硬盘、光盘等，容量大，速度慢，长期保存数据）；输入设备和输出设备合并称为外设。

鼠标键盘等输入设备都是通过中断的原理来实现控制，点击后触发中断，首先进入中断处理程序。

主机：CPU（运算器、控制器）+主存储器。

CPU：由运算器、控制器、寄存器组和内部总线组成。实现程序控制、操作控制、时间控制、数据处理功能。

运算器：由**算术逻辑单元 ALU**（实现对数据的算术和逻辑运算）、**累加寄存器 AC**（运算结果或源操作数的存放区）、**数据缓冲寄存器 DR**（暂时存放内存的指令或数据）、和**状态条件寄存器 PSW**（保存指令运行结果的条件码内容，如溢出标志等）组成。执行所有的算术运算，如加减乘除等；执行所有的逻辑运算并进行逻辑测试，如与、或、非、比较等。

控制器：由**指令寄存器 IR**（暂存 CPU 执行指令）、**程序计数器 PC**（存放指令执行地址）、**地址寄存器 AR**（保存当前 CPU 所访问的内存地址）、**指令译码器 ID**（分析指令操作码）等组成。控制整个 CPU 的工作，最为重要。

CPU 依据**指令周期的不同阶段**来区分二进制的指令和数据，因为在指令周期的不同阶段，指令会命令 CPU 分别去取指令或者数据。

计算机指令的组成

一条指令由操作码和操作数两部分组成，操作码决定要完成的操作，操作数指参加运算的数据及其所在的单元地址。

在计算机中，操作要求和操作数地址都由二进制数码表示，分别称作操作码和地址码，整条指令以二进制编码的形式存放在存储器中。

计算机指令执行过程

可分为取指令——分析指令——执行指令三个步骤，首先将程序计数器 PC 中的指令地址取出，送入地址总线，CPU 依据指令地址去内存中取出指令内容存入指令寄存器 IR；而后由指令译码器进行分析，分析指令操作码；最后执行指令，取出指令执行所需的源操作数。

指令寻址方式

顺序寻址方式：由于指令地址在主存中顺序排列，当执行一段程序时，通常是一条指令接着一指令地顺序执行。从存储器取出第一条指令，然后执行这条指令；接着从存储器取出第二条指令，再执行第二条指令；……，以此类推。这种程序顺序执行的过程称为指令的顺序寻址方式。

跳跃寻址方式：所谓指令的跳跃寻址，是指下一条指令的地址码不是由程序计数器给出，而是由本条指令直接给出。程序跳跃后，按新的指令地址开始顺序执行。因此，指令计数器的内容也必须相应改变，以便及时跟踪新的指令地址。

指令操作数的寻址方式

立即寻址方式：指令的地址码字段指出的不是地址，而是操作数本身。

直接寻址方式：在指令的地址字段中直接指出操作数在主存中的地址。

间接寻址方式：与直接寻址方式相比，间接寻址中指令地址码字段所指向的存储单元中存储的不是操作数本身，而是操作数的地址。

寄存器寻址方式：指令中的地址码是寄存器的编号，而不是操作数地址或操作数本身。寄存器的寻址方式也可以分为直接寻址和间接寻址，两者的区别在于：前者的指令地址码给出寄存器编号，寄存器的内容就是操作数本身；而后者的指令地址码给出寄存器编号，寄存器的内容是操作数的地址，根据该地址访问主存后才能得到真正的操作数。

基址寻址方式：将基址寄存器的内容加上指令中的形式地址而形成操作数的有效地址，其优点是可以扩大寻址能力。

变址寻址方式：变址寻址方式计算有效地址的方法与基址寻址方式很相似，它是将变址寄存器的内容加上指令中的形式地址而形成操作数的有效地址。

相对寻址方式：相对于当前的指令地址而言的寻址方式。相对寻址是把程序计数器 PC 的内容加上指令中的形式地址而形成操作数的有效地址，而程序计数器的内容就是当前指令的地址，所以相对寻址是相对于当前的指令地址而言的。

3 指令系统 CISC 和 RISC

CISC 是复杂指令系统，兼容性强，指令繁多、长度可变，由微程序实现；

RISC 是精简指令系统，指令少，使用频率接近，主要依靠硬件实现（通用寄存器、硬布线逻辑控制）。

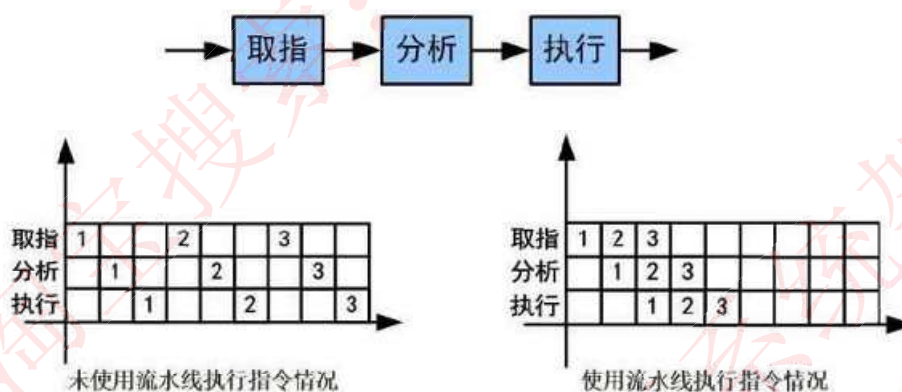
二者各方面区分如下图：

指令系统类型	指令	寻址方式	实现方式	其它
CISC（复杂）	数量多，使用频率差别大，可变长格式	支持多种	微程序控制技术（微码）	研制周期长
RISC（精简）	数量少，使用频率接近，定长格式，大部分为单周期指令，操作寄存器，只有Load/Store操作内存	支持方式少	增加了通用寄存器；硬布线逻辑控制为主；适合采用流水线	优化编译，有效支持高级语言

4 指令的流水处理

4.1 流水线原理

将指令分成不同段，每段由不同的部分去处理，因此可以产生叠加的效果，所有的部件去处理指令的不同段，如下图所示：



RISC 中的流水线技术：超流水线（在每个机器周期内能完成一个甚至两个浮点操作，以时间换空间）、超标量（内装多条流水线同时执行多个处理，以空间换时间）、超长指令字 VLIW（同时执行多条指令，发挥软件作用）。

4.2 流水线时间计算

流水线周期：指令分成不同执行段，其中执行时间最长的段为流水线周期。

流水线执行时间： $1 \text{ 条指令总执行时间} + (\text{总指令条数} - 1) * \text{流水线周期}$ 。

单缓冲区和双缓冲区：此类题型不给出具体流水线执行阶段，需要考生自己区分出流水线阶段，一般来说，能够同时执行的阶段就是流水线的独立执行阶段；只能独立执行的阶段应该合并为流水线中的一个独立执行阶段。

例如有三个阶段即读入缓冲区+送入用户区+数据处理，在单缓冲区中，缓冲区和用户区都只有一个，一个盘块必须执行完前两个阶段，下一个盘块才能开始，因此前两个阶段应该合并，整个流水线为送入用户区+数据处理；

而在双缓冲区中，盘块可以交替读入缓冲区，但用户区只有一个，因为缓冲区阶段可以同时进行，流水线前两个阶段不能合并，就是读入缓冲区+送入用户区+数据处理三段。

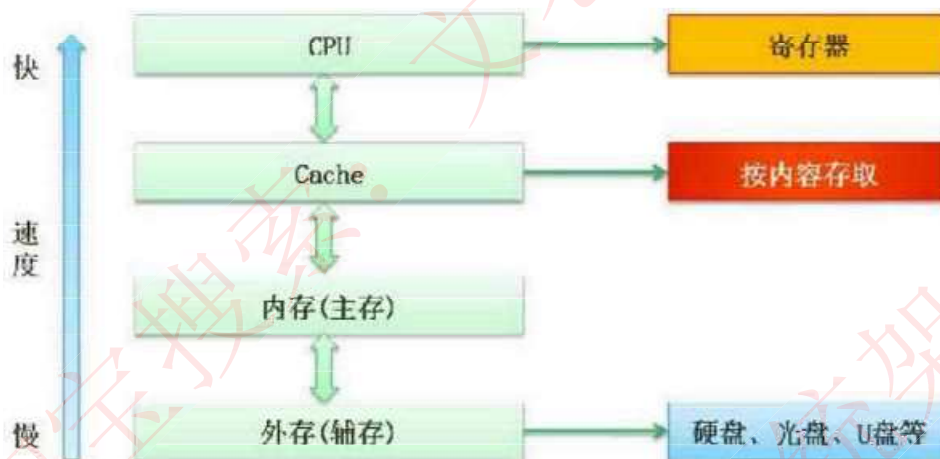
划分出真正的流水线阶段后，套用流水线时间计算公式可以轻易得出答案。

流水线吞吐率计算：吞吐率即单位时间内执行的指令条数。公式：指令条数/流水线执行时间。

流水线的加速比计算：加速比即使用流水线后的效率提升度，即比不使用流水线快了多少倍，越高表明流水线效率越高，公式：不使用流水线执行时间/使用流水线执行时间。

5 存储系统

5.1 计算机存储结构层次图



计算机采用分级存储体系的主要目的是为了解决存储容量、成本和速度之间的矛盾问题。

两级存储映像为：Cache-主存、主存-辅存（虚拟存储体系）。

存储器的分类：

按存储器所处的位置：内存、外存。

按存储器构成材料：磁存储器（磁带）、半导体存储器、光存储器（光盘）。

按存储器的工作方式：可读可写存储器（RAM）、只读存储器（ROM 只能读，PROM 可写入一次，EPROM 和 EEPROM 既可以读也可以写，只是修改方式不用）。

按存储器访问方式：按地址访问、按内容访问（相联存储器）。

按寻址方式：随机存储器（访问任意存储单元所用时间相同）、顺序存储器（只能按顺序访问，如磁带）、直接存储器（二者结合，如磁盘，对于磁道的寻址是随机的，在一个磁道内则是顺序的）。

5.2 局部性原理

总的来说, 在 CPU 运行时, 所访问的数据会趋向于一个较小的局部空间地址内。(例如循环操作, 循环体被反复执行)

时间局部性原理: 如果一个数据项正在被访问, 那么在近期它很可能会被再次访问, 即在**相邻的时间里会访问同一个数据项**。

空间局部性原理: 在最近的将来会用到的数据的地址和现在正在访问的数据地址很可能是相近的, 即**相邻的空间地址会被连续访问**。

5.3 高速缓存 Cache

高速缓存 Cache 用来存储当前最活跃的程序和数据, 直接与 CPU 交互, 位于 CPU 和主存之间, 容量小, 速度为内存的 5-10 倍, 由半导体材料构成。其内容是主存内存的副本拷贝, 对于程序员来说是透明的。

Cache 由控制部分和存储器组成, 存储器存储数据, 控制部分判断 CPU 要访问的数据是否在 Cache 中, 在则命中, 不在则依据一定的算法从主存中替换。

地址映射方法

在 CPU 工作时, 送出的是主存单元的地址, 而应从 Cache 存储器中读/写信息。这就需要将主存地址转换为 Cache 存储器地址, 这种地址的转换称为地址映像, 由**硬件自动完成映射**, 分为下列三种方法:

直接映像: 将 Cache 存储器等分成块, 主存也等分成块并编号。主存中的块与 Cache 中的块的对应关系是固定的, 也即二者块号相同才能命中。地址变换简单但不灵活, 容易造成资源浪费。

全相联映像: 同样都等分成块并编号。主存中任意一块都与 Cache 中任意一块对应。因此可以随意调入 Cache 任意位置, 但地址变换复杂, 速度较慢。因为主存可以随意调入 Cache 任意块, 只有当 Cache 满了才会发生块冲突, 是最不容易发生块冲突的映像方式。

组组相连映像: 前面两种方式的结合, 将 Cache 存储器先分块再分组, 主存也同样先分块再分组, 组间采用直接映像, 即主存中组号与 Cache 中组号相同的组才能命中, 但是组内全相联映像, 也即组号相同的两个组内的所有块可以任意调换。

命中率及平均时间

Cache 存储器的大小一般为 K 或者 M 单位, 很小, 但是最快, 仅次于 CPU 中的寄存器, 而寄存器一般不算作存储器, CPU 与内存之间的数据交互, 内存会先将数据拷贝到 Cache 里, 这样, 根据局部性原理, 若 Cache 中的数据被循环执行, 则不用每次都去内存中读取数据, 会加快 CPU 工作效率。

因此, Cache 有一个命中率的概念, 即当 CPU 所访问的数据在 Cache 中时, 命中, 直接从 Cache 中读取数据, 设读取一次 Cache 时间为 1ns, 若 CPU 访问的数据不在 Cache 中, 则需要从内存中读取, 设读取一次内存的时间为 1000ns, 若在 CPU 多次读取数据过程中, 有 90%命中 Cache, 则 CPU 读取一次的平均时间为 $(90\%*1 + 10\%*1000)ns$, 很容易理解。

5.4 虚拟存储器

虚拟存储器技术是将很大的数据分成许多较小的块，全部存储在外存中。运行时，将用到的数据调入主存中，马上要用到的数据置于缓存中，这样，一边运行一边进行所需数据块的调入/调出。对于应用程序员来说，就好像有一个比实际主存空间大得多的虚拟主存空间，基本层级为：**主存——缓存——外存**。与 **CPU——高速缓存 Cache——主存** 的原理类似。但虚拟存储器中程序员无需考虑地址映像关系，由系统自动完成，因此对于程序来说是透明的。

其管理方式分为页式、段式、段页式，在 2.2 存储管理中详细介绍。

5.5 磁盘

磁盘结构和参数

磁盘有正反两个盘面，每个盘面有多个同心圆，每个同心圆是一个磁道，每个同心圆又被划分为多个扇区，数据就被存放在一个个扇区中。

磁头首先要寻找到对应的磁道，然后等待磁盘进行周期旋转，旋转到指定的扇区，才能读取到对应的数据，因此，会产生寻道时间和等待时间。公式为：

存取时间=寻道时间+等待时间（平均定位时间+转动延迟）。

注意：寻道时间是指磁头移动到磁道所需的时间；等待时间为等待读写的扇区转到磁头下方所用的时间。

磁盘调度算法

之前已经说过，磁盘数据的读取时间分为寻道时间+旋转时间，也即先找到对应的磁道，而后再旋转到对应的扇区才能读取数据，其中寻道时间耗时最长，需要重点调度，有如下调度算法：

先来先服务 FCFS：根据进程请求访问磁盘的先后顺序进行调度。

最短寻道时间优先 SSTF：请求访问的磁道与当前磁道最近的进程优先调度，使得每次的寻道时间最短。会产生“饥饿”现象，即远处进程可能永远无法访问。

扫描算法 SCAN：又称“电梯算法”，磁头在磁盘上双向移动，其会选择离磁头当前所在磁道最近的请求访问的磁道，并且与磁头移动方向一致，磁头永远都是从里向外或者从外向里一直移动完才掉头，与电梯类似。

单向扫描调度算法 CSCAN：与 SCAN 不同的是，其只做单向移动，即只能从里向外或者从外向里。

6 输入输出技术

计算机系统中存在多种内存与接口地址的编址方法，常见的是下面两种：内存与接口地址独立编址和内存与接口地址统一编址。

1) 内存与接口地址独立编址方法

在内存与接口地址独立编址方法下，内存地址和接口地址是完全独立的两个地址空间，它们是完全独立的并且是相互隔离的。访问数据时所使用的指令也完全不同，用于接口的指令只用于接口的读/写，其余的指令全都是用于内存的。因此，在编程序或读程序时很易使用和辨认。这种编址方法的缺点是用于接口的指令太少、功能太弱。

2) 内存与接口地址统一编址方法

在这种编址方法中，内存地址和接口地址统一在一个公共的地址空间里，即内存单元和接口共用地址空间。在这些地址空间里划分出一部分地址分配给接口使用，其余地址归内存单元使用。这种编址方法的优点是原则上用于内存的指令全都可以用于接口，这就大大地增强了对接口的操作功能，而且在指令上也不再区分内存或接口指令。该编址方法的缺点就在于整个地址空间被分成两部分，其中一部分分配给接口使用，剩余的为内存所用，这经常会导致内存地址不连续。

程序控制（查询）方式：CPU 主动查询外设是否完成数据传输，效率极低。

程序中断方式：外设完成数据传输后，向 CPU 发送中断，等待 CPU 处理数据，效率相对较高。中断响应时间指的是从发出中断请求到开始进入中断处理程序；中断处理时间指的是从中断处理开始到中断处理结束。中断向量提供中断服务程序的入口地址。多级中断嵌套，使用堆栈来保护断点和现场。

DMA 方式（直接主存存取）：CPU 只需完成必要的初始化等操作，数据传输的整个过程都由 DMA 控制器来完成，在主存和外设之间建立直接的数据通路，效率很高。在一个总线周期结束后，CPU 会响应 DMA 请求开始读取数据；CPU 响应程序中断方式请求是在一条指令执行结束时；区分指令执行结束和总线周期结束。

7 总线

从广义上讲，任何连接两个以上电子元器件的导线都可以称为总线，通常分为以下三类：

内部总线：内部芯片级别的总线，芯片与处理器之间通信的总线。

系统总线：是板级总线，用于计算机内各部分之间的连接，具体分为数据总线（并行数据传输位数）、地址总线（系统可管理的内存空间的大小）、控制总线（传送控制命令）。代表的有 ISA 总线、EISA 总线、PCI 总线。

外部总线：设备一级的总线，微机和外部设备的总线。代表的有 RS232（串行总线）、SCSI（并行总线）、USB（通用串行总线，即插即用，支持热插拔）。

并行总线适合近距离高速数据传输，串行总线适合长距离数据传输，专用总线在设计上可以与连接设备实现最佳匹配。

总线计算：总线的时钟周期=时钟频率的倒数；总线的宽度（传输速率）=单位时间内传输的数据总量/单位时间大小。

二 操作系统知识

1 操作系统概述

操作系统定义：能有效地组织和管理系统中的各种软/硬件资源，合理地组织计算机系统工作流程，控制程序的执行，并且向用户提供一个良好的工作环境和友好的接口。

操作系统有三个重要的作用：第一，管理计算机中运行的程序和分配各种软硬件资源；第二，为用户提供友善的人机界面。第三，为应用程序的开发和运行提供一个高效率的平台。

操作系统的 4 个特征是并发性、共享性、虚拟性和不确定性。

操作系统的功能：

(1) 进程管理。实质上是对处理机的执行“时间”进行管理，采用多道程序等技术将 CPU 的时间合理地分配给每个任务，主要包括进程控制、进程同步、进程通信和进程调度。

(2) 文件管理。主要包括文件存储空间管理、目录管理、文件的读/写管理和存取控制。

(3) 存储管理。存储管理是对主存储器“空间”进行管理，主要包括存储分配与回收、存储保护、地址映射（变换）和主存扩充。

(4) 设备管理。实质是对硬件设备的管理，包括对输入/输出设备的分配、启动、完成和回收。

(5) 作业管理。包括任务、界面管理、人机交互、图形界面、语音控制和虚拟现实等。

操作系统的分类：

批处理操作系统：单道批处理和多道批处理（主机与外设可并行）。

分时操作系统：一个计算机系统与多个终端设备连接。分时操作系统是将 CPU 的工作时间划分为许多很短的时间片，轮流为各个终端的用户服务。

实时操作系统：实时是指计算机对于外来信息能够以足够快的速度进行处理，并在被控对象允许的时间范围内做出快速反应。实时系统对交互能力要求不高，但要求可靠性有保障。为了提高系统的响应时间，对随机发生的外部事件应及时做出响应并对其进行处理。

网络操作系统：是使联网计算机能方便而有效地共享网络资源，为网络用户提供各种服务的软件和有关协议的集合。功能主要包括高效、可靠的网络通信；对网络中共享资源的有效管理；提供电子邮件、文件传输、共享硬盘和打印机等服务；网络安全管理；提供互操作能力。三种模式：集中模式、客户端/服务器模式、对等模式。

分布式操作系统：分布式计算机系统是由多个分散的计算机经连接而成的计算机系统，系统中的计算机无主、次之分，任意两台计算机可以通过通信交换信息。通常，为分布式计算机系统配置的操作系统称为分布式操作系统。是网络操作系统的更高级形式。

微型计算机操作系统：简称微机操作系统，常用的有 Windows、Mac OS、Linux。

嵌入式操作系统：运行在嵌入式智能芯片环境中，对整个智能芯片以及它所操作、控制的各种部件装置等资源进行统一协调、处理、指挥和控制。其主要特点如下：

(1) 微型化。从性能和成本角度考虑，希望占用的资源和系统代码量少，如内存少、字长短、运行速度有限、能源少（用微型电池）。

(2) 可定制。从减少成本和缩短研发周期考虑，要求嵌入式操作系统能运行在不同的微处理器平台上，能针对硬件变化进行结构与功能上的配置，以满足不同应用需要。

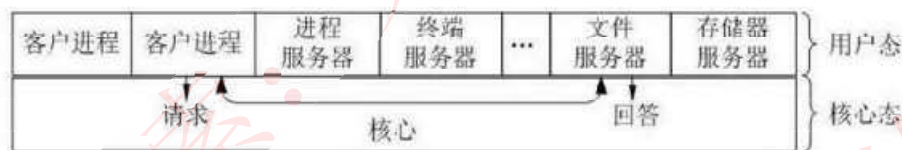
(3) 实时性。嵌入式操作系统主要应用于过程控制、数据采集、传输通信、多媒体信息及关键要害领域需要迅速响应的场合，所以对实时性要求较高。

(4) 可靠性。系统构件、模块和体系结构必须达到应有的可靠性，对关键要害应用还要提供容错和防故障措施。

(5) 易移植性。为了提高系统的易移植性，通常采用硬件抽象层（Hardware Abstraction Level, HAL）和板级支撑包（Board Support Package, BSP）的底层设计技术。

嵌入式系统**初始化过程**按照自底向上、从硬件到软件的次序依次为：片级初始化→板级初始化→系统初始化。芯片级是微处理器的初始化，板卡级是其他硬件设备初始化，系统级初始化就是软件及操作系统初始化。

微内核操作系统：微内核，顾名思义，就是尽可能的将内核做的很小，只将最为核心必要的东西放入内核中，其他能独立的东西都放入用户进程中，这样，系统就被分为了用户态和内核态，如下图所示：



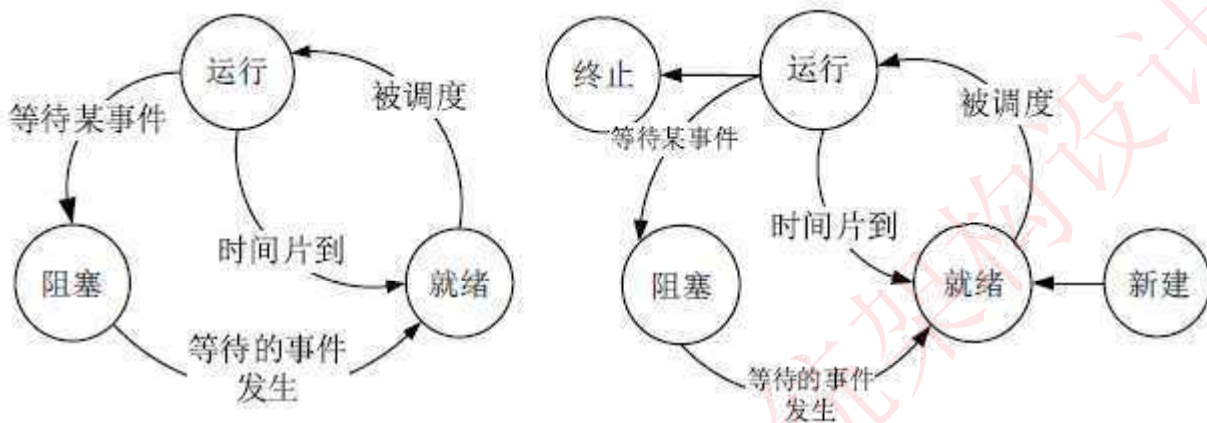
	实质	优点	缺点
单体内核	将图形、设备驱动及文件系统等全部在内核中实现，运行在内核状态和同一地址空间。	减少进程间通信和状态切换的系统开销，获得较高的运行效率。	内核庞大，占用资源较多且不易剪裁。 系统的稳定性和安全性不好。
微内核	只实现基本功能，将图形系统、文件系统、设备驱动及通信功能放在内核之外。	内核精练，便于剪裁和移植。 系统服务程序运行在用户地址空间，系统的可靠性、稳定性和安全性较高。 可用于分布式系统	用户状态和内核状态需要频繁切换，从而导致系统效率不如单体内核。

2 进程管理

2.1 进程的组成和状态

进程的组成：进程控制块 PCB（唯一标志）、程序（描述进程要做什么）、数据（存放进程执行时所需数据）。

进程基础的状态是下左图中的**三态图**。需要熟练掌握左下图中的进程三态之间的转换。

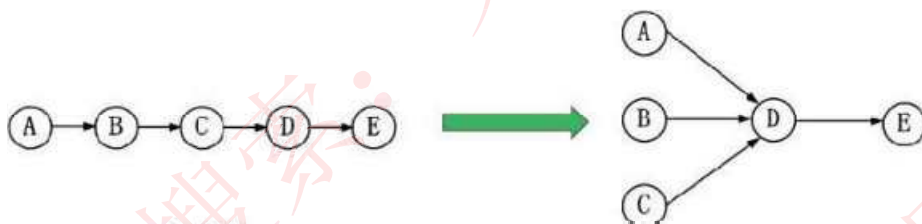


新建态对应于进程刚刚被创建时没有被提交的状态，并等待系统完成创建进程的所有必要信息。终止态等待操作系统进行善后处理，释放主存。

2.2 前趋图和进程资源图

前趋图

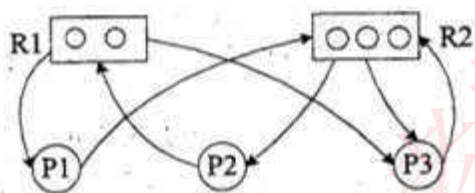
用来表示哪些任务可以并行执行，哪些任务之间有顺序关系，具体如下图：



可知，A B C 可以并行执行，但是必须 A B C 都执行完后，才能执行 D，这就确定了两点：任务间的并行、任务间的先后顺序。

进程资源图

用来表示进程和资源之间的分配和请求关系，如下图所示：



P 代表进程，R 代表资源，R 方框中有几个圆球就表示有几个这种资源，在上图中，R1 指向 P1，表示 R1 有一个资源已经分配给了 P1，P1 指向 R2，表示 P1 还需要请求一个 R2 资源才能执行。

阻塞节点：某进程所请求的资源已经全部分配完毕，无法获取所需资源，该进程被阻塞了无法继续。如上图中 P2。

非阻塞节点：某进程所请求的资源还有剩余，可以分配给该进程继续运行。如上图中 P1、P3。

当一个进程资源图中所有进程都是阻塞节点时，即陷入死锁状态。

进程资源图的化简方法：先看系统还剩下多少资源没分配，再看有哪些进程是不阻塞的，接着把不阻塞的进程的所有边都去掉，形成一个孤立的点，再把系统分配给这个进程的资源回收回来，这样，

系统剩余的空闲资源便多了起来，接着又去看看剩下的进程有哪些是不阻塞的，然后又把它们逐个变成孤立的点。最后，所有的资源和进程都变成孤立的点。

2.3 进程间的同步与互斥

互斥和同步并非反义词，互斥表示一个资源在同一时间内只能由一个任务单独使用，需要加锁，使用完后解锁才能被其他任务使用；

同步表示两个任务可以同时执行，只不过有速度上的差异，需要速度上匹配，不存在资源是否单独或共享的问题。

2.4 信号量操作

基本概念

临界资源：各个进程间需要互斥方式对其进行共享的资源，即在某一时刻只能被一个进程使用，该进程释放后又可以被其他进程使用。

临界区：每个进程中访问临界资源的那段代码。

信号量：是一种特殊的变量。

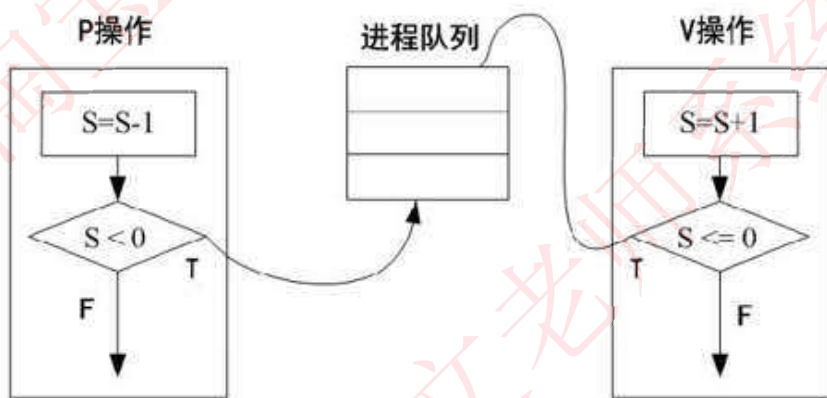
两类信号量

互斥信号量，对临界资源采用互斥访问，使用互斥信号量后其他进程无法访问，初值为 1。

同步信号量，对共享资源的访问控制，初值是共享资料的个数。

P 操作和 V 操作

都是原子操作，用来解释进程间的同步和互斥原理，PV 操作原理如下图所示：



例如在**生产者和消费者的问题**中，生产者生产一个商品 s ，而后要申请互斥的使用该仓库，即首先需要执行互斥信号量 $P(S_0)$ ，申请到仓库独立使用权后，再判断**仓库是否有空闲**（信号量 S_1 ），执行 $P(S_1)$ ，若结果大于等于 0，表示仓库有空闲，再将 s 放入仓库中，此时**仓库商品数量**（信号量 S_2 ）增加 1，即执行 $V(S_2)$ 操作，使用完毕后，释放互斥信号量 $V(S_0)$ 。

对于消费者，首先也需要执行互斥信号量 $P(S_0)$ ，申请到仓库独立使用权后，再判断仓库中是否有商品，执行 $P(S_2)$ ，若结果大于等于 0，表示有商品，可以取出，此时造成了一个结果，即仓库空闲了一个，执行 $V(S_1)$ 操作，使用完毕后，释放互斥信号量 $V(S_0)$ 。

因此，执行 P 操作是**主动的带有判断性质**的-1，执行 V 操作是**被动的因为某操作产生的**+1。

2.5 进程调度

进程调度方式是指当有更高优先级的进程到来时如何分配 CPU。分为可剥夺和不可剥夺两种，可剥夺指当有更高优先级进程到来时，强行将正在运行进程的 CPU 分配给高优先级进程；不可剥夺是指高优先级进程必须等待当前进程自动释放 CPU。

在某些操作系统中，一个作业从提交到完成需要经历高、中、低三级调度。

(1) 高级调度。高级调度又称“长调度”“作业调度”或“接纳调度”，它决定处于输入池中的哪个后备作业可以调入主系统做好运行的准备，成为一个或一组就绪进程。在系统中一个作业只需经过一次高级调度。

(2) 中级调度。中级调度又称“中程调度”或“对换调度”，它决定处于交换区中的哪个就绪进程可以调入内存，以便直接参与对 CPU 的竞争。

(3) 低级调度。低级调度又称“短程调度”或“进程调度”，它决定处于内存中的哪个就绪进程可以占用 CPU。低级调度是操作系统中最活跃、最重要的调度程序，对系统的影响很大。

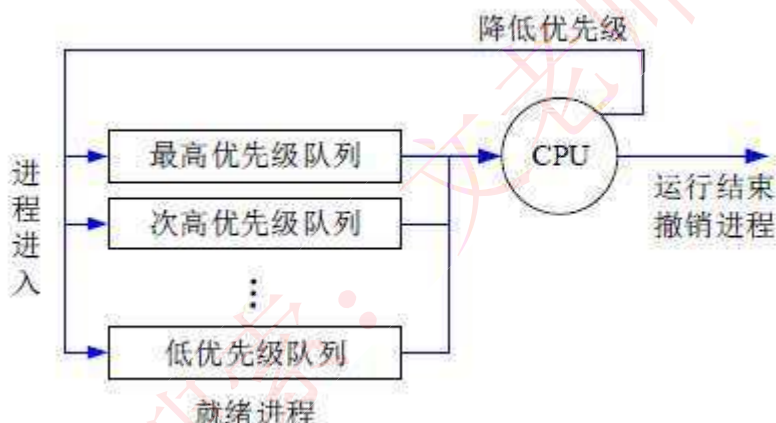
调度算法：

先来先服务 FCFS：先到达的进程优先分配 CPU。用于宏观调度。

时间片轮转：分配给每个进程 CPU 时间片，轮流使用 CPU，每个进程时间片大小相同，很公平，用于微观调度。

优先级调度：每个进程都拥有一个优先级，优先级大的先分配 CPU。

多级反馈调度：时间片轮转和优先级调度结合而成，设置多个就绪队列 1,2,3...n，每个队列分别赋予不同的优先级，分配不同的时间片长度；新进程先进入队列 1 的末尾，按 FCFS 原则，执行队列 1 的时间片；若未能执行完进程，则转入队列 2 的末尾，如此重复。



2.6 死锁问题

当一个进程在等待永远不可能发生的事件时，就会产生死锁，若系统中有多个进程处于死锁状态，就会造成系统死锁。

死锁产生的四个必要条件：资源互斥、每个进程占有资源并等待其他资源、系统不能剥夺进程资源、进程资源图是一个环路。

死锁产生后，解决措施是打破四大条件，有下列方法：

死锁预防：采用某种策略限制并发进程对于资源的请求，破坏死锁产生的四个条件之一，使系统任何时刻都不满足死锁的条件。

死锁避免：一般采用银行家算法来避免，银行家算法，就是提前计算出一条不会死锁的资源分配方法，才分配资源，否则不分配资源，相当于借贷，考虑对方还得起才借钱，提前考虑好以后，就可以避免死锁。

死锁检测：允许死锁产生，但系统定时运行一个检测死锁的程序，若检测到系统中发生死锁，则设法加以解除。

死锁解除：即死锁发生后的解除方法，如强制剥夺资源，撤销进程等。

死锁资源计算：系统内有 n 个进程，每个进程都需要 R 个资源，那么其发生死锁的最大资源数为 $n*(R-1)$ 。其不发生死锁的最小资源数为 $n*(R-1)+1$ 。

2.7 线程

传统的进程有两个属性：可拥有资源的独立单位；可独立调度和分配的基本单位。

引入线程的原因是进程在创建、撤销和切换中，系统必须为之付出较大的时空开销，故在系统中设置的进程数目不宜过多，进程切换的频率不宜太高，这就限制了并发程度的提高。引入线程后，将传统进程的两个基本属性分开，线程作为调度和分配的基本单位，进程作为独立分配资源的单位。用户可以通过创建线程来完成任务，以减少程序并发执行时付出的时空开销。

线程是进程中的一个实体，是被系统独立分配和调度的基本单位。线程基本上不拥有资源，只拥有一点运行中必不可少的资源（如程序计数器、一组寄存器和栈），它可与同属一个进程的其他线程共享进程所拥有的全部资源，例如进程的公共数据、全局变量、代码、文件等资源，但不能共享线程独有的资源，如线程的栈指针等标识数据。

3 存储管理

存储器的结构：寄存器——高速缓存 Cache——主存——外存。

地址重定位：将逻辑地址转换为实际主存物理地址的过程，分为静态重定位（在程序装入主存时就完成了转换）、动态重定位（边运行边转换）。

3.1 分区存储管理

所谓分区存储组织，就是整存，将某进程运行所需的内存整体一起分配给它，然后再执行。有三种分区方式：

固定分区：静态分区方法，将主存分为若干个固定的分区，将要运行的作业装配进去，由于分区固定，大小和作业需要的大小不同，会产生内部碎片。

可变分区：动态分区方法，主存空间的分区是在作业转入时划分，正好划分为作业需要的大小，这样就不存在内部碎片，但容易将整片主存空间切割成许多块，会产生外部碎片。可变分区的算法如下：

系统分配内存的算法有很多，如下图所示，根据分配前的内存情况，还需要分配 9K 空间，对不同算法的结果介绍如下：

首次适应法：按内存地址顺序从头查找，找到第一个 $\geq 9K$ 空间的空闲块，即切割 9K 空间分配给进程。

最佳适应法：将内存中所有空闲内存块按从小到大排序，找到第一个 $\geq 9K$ 空间的空闲块，切割分配，这个将会找到与 9K 空间大小最相近的空闲块。

最差适应法：和最佳适应法相反，将内存中空闲块空间最大的，切割 9K 空间分配给进程，这是为了预防系统中产生过多的细小空闲块。

循环首次适应法：按内存地址顺序查找，找到第一个 $\geq 9K$ 空间的空闲块，而后若还需分配，则找下一个，不用每次都从头查找，这是与首次适应法不同的地方。

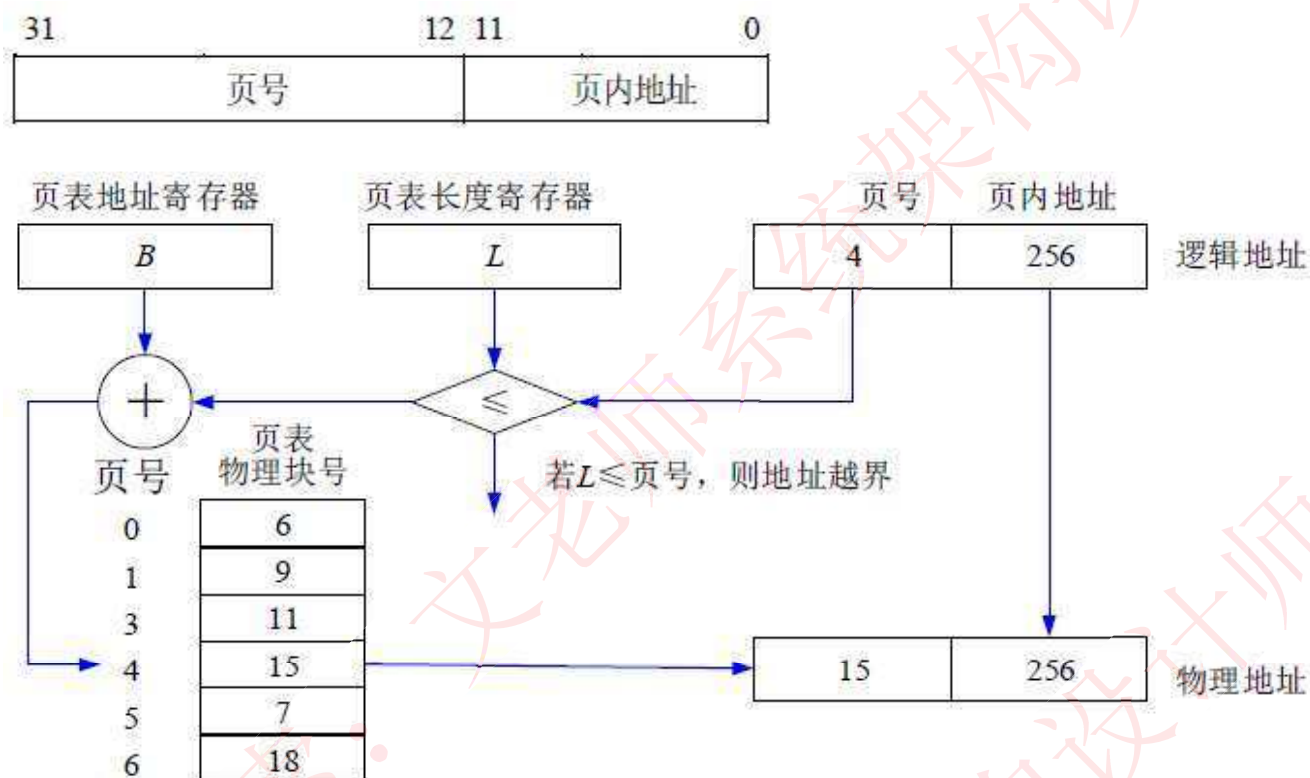


可重定位分区：可以解决碎片问题，移动所有已经分配好的区域，使其成为一个连续的区域，这样其他外部细小的分区碎片可以合并为大的分区，满足作业要求。只在外部作业请求空间得不到满足时进行。

3.2 分页存储管理

如果采用分区存储，都是整存，会出现一个问题，即当进程运行所需的内存大于系统内存时，就无法将整个进程一起调入内存，因此无法运行，若要解决此问题，就要采用段页式存储组织，页式存储是基于可变分区而提出的。

如下图所示，逻辑页分为页号和页内地址，页内地址就是物理偏移地址，而页号与物理块号并非按序对应的，需要查询页表，才能得知页号对应的物理块号，再用物理块号加上偏移地址才得出了真正运行时的物理地址。



优点：利用率高，碎片小，分配及管理简单。

缺点：增加了系统开销，可能产生抖动现象

地址表示和转换

地址组成：页地址+页内偏移地址；（页地址在高位，页内偏移地址在低位）

物理地址：物理块号+页内偏移地址；

逻辑地址：页号+页内偏移地址；

物理地址和逻辑地址的页内偏移地址是一样的，只要求出页号和物理块号之间的对应关系，首先要求出页号的位数，得出页号，再去页表里查询其对应的物理块号，使用此物理块号和页内偏移地址组合，就能得到物理地址。

详细转换流程见文老师视频课程讲解，有详细例题解析。

页面置换算法

有时候，进程空间分为 100 个页面，而系统内存只有 10 个物理块，无法全部满足分配，就需要将马上要执行的页面先分配进去，而后根据算法进行淘汰，使 100 个页面能够按执行顺序调入物理块中执行完。

缺页表示需要执行的页不在内存物理块中，需要从外部调入内存，会增加执行时间，因此，缺页数越多，系统效率越低。页面置换算法如下：

最优算法：OPT，理论上的算法，无法实现，是在进程执行完后进行的最佳效率计算，用来让其他算法比较差距。原理是选择未来最长时间不被访问的页面置换，这样可以保证未来执行的都是马上要访问的。

先进先出算法：FIFO，先调入内存的页先被置换淘汰，会产生**抖动现象**，即分配的页数越多，缺页率可能越多（即效率越低）。

最近最少使用：LRU，在最近的过去，进程执行过程中，过去最少使用的页面被置换淘汰，根据局部性原理，这种方式效率高，且不会产生抖动现象，使用大量计数器，但是没有LFU多。

淘汰原则：优先淘汰最近未访问的，而后淘汰最近未被修改的页面。

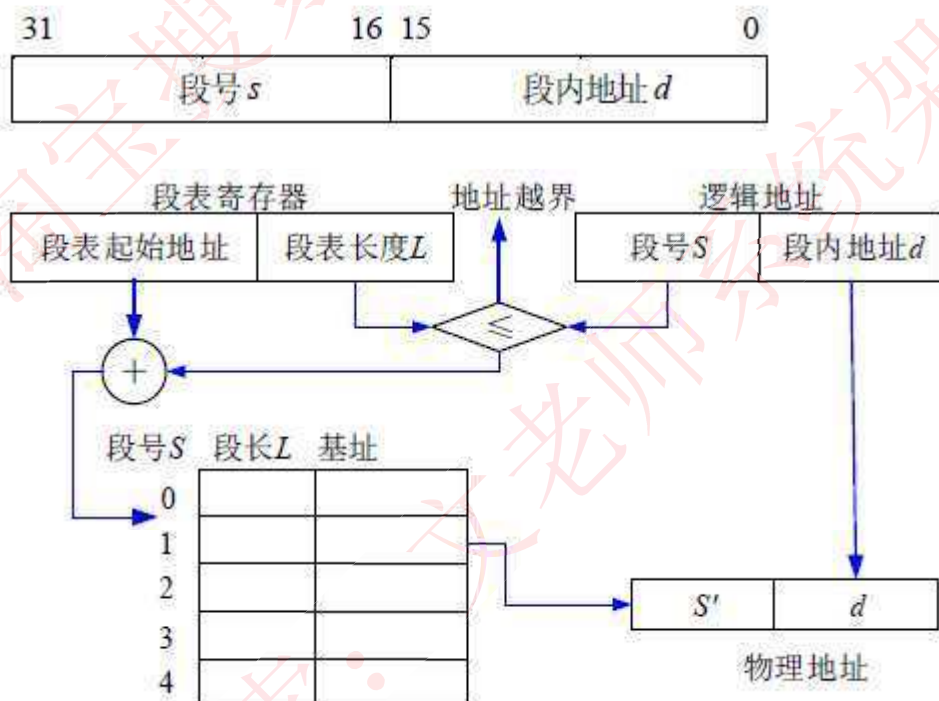
快表

是一块小容量的相联存储器，由快速存储器组成，按内容访问，速度快，并且可以从硬件上保证按内容并行查找，一般用来存放当前访问最频繁的少数活动页面的页号。

快表是将页表存于Cache中；慢表是将页表存于内存上。慢表需要访问两次内存才能取出页，而快表是访问一次Cache和一次内存，因此更快。

3.3 段式存储管理

将进程空间分为一个个段，每段也有段号和段内地址，与页式存储不同的是，**每段物理大小不同**，分段是根据逻辑整体分段的，因此，段表也与页表的内容不同，页表中直接是逻辑页号对应物理块号，而下图所示，段表有段长和基址两个属性，才能确定一个逻辑段在物理段中的位置。



优点：多道程序共享内存，各段程序修改互不影响。

缺点：内存利用率低，内存碎片浪费大。

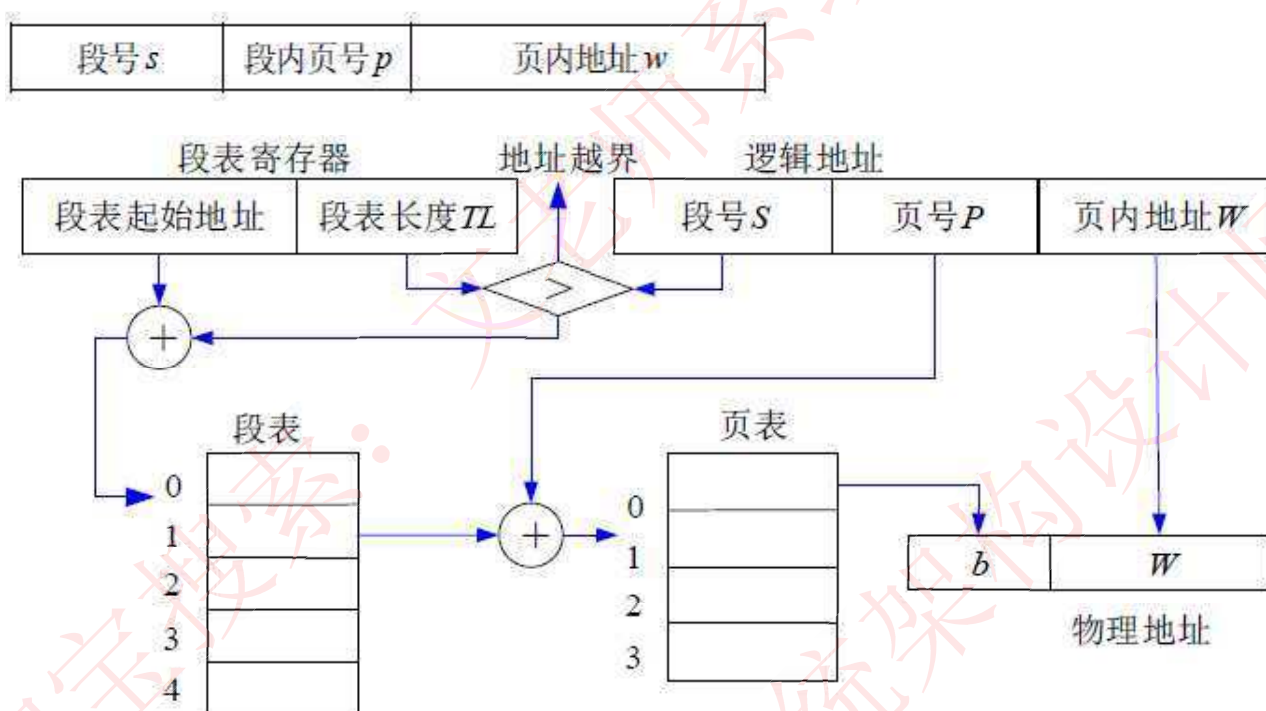
综上：分页是根据物理空间划分，每页大小相同；分段是根据逻辑空间划分，每段是一个完整的功能，便于共享，但是大小不同。

地址表示

(段号，段内偏移)：其中段内偏移不能超过该段号对应的段长，否则越界错误，而此地址对应的真正内存地址应该是：段号对应的基地址+段内偏移。

3.4 段页式存储管理

对进程空间先分段，后分页，具体原理图和优缺点如下：



优点：空间浪费小、存储共享容易、存储保护容易、能动态链接。

缺点：由于管理软件的增加，复杂性和开销也随之增加，需要的硬件以及占用的内容也有所增加，使得执行速度大大下降。

4 设备管理

4.1 概述

设备是计算机系统与外界交互的工具，具体负责计算机与外部的输入/输出工作，所以常称为外部设备（简称外设）。在计算机系统中，将负责管理设备和输入/输出的机构称为 I/O 系统。因此，I/O 系统由设备、控制器、通道（具有通道的计算机系统）、总线和 I/O 软件组成。

设备的分类：

按数据组织分类：块设备、字符设备。

按照设备功能分类：输入设备、输出设备、存储设备、网络联网设备、供电设备等等。

资源分配角度分类：独占设备、共享设备和虚拟设备。

数据传输速率分类：低速设备、中速设备、高速设备。

设备管理的任务是保证在多道程序环境下，当多个进程竞争使用设备时，按一定的策略分配和管理各种设备，控制设备的各种操作，完成 I/O 设备与主存之间的数据交换。

设备管理的主要功能是动态地掌握并记录设备的状态、设备分配和释放、缓冲区管理、实现物理 I/O 设备的操作、提供设备使用的用户接口及设备的访问和控制。

4.2 I/O 软件

I/O 设备管理软件的所有层次及每一层功能如下图：

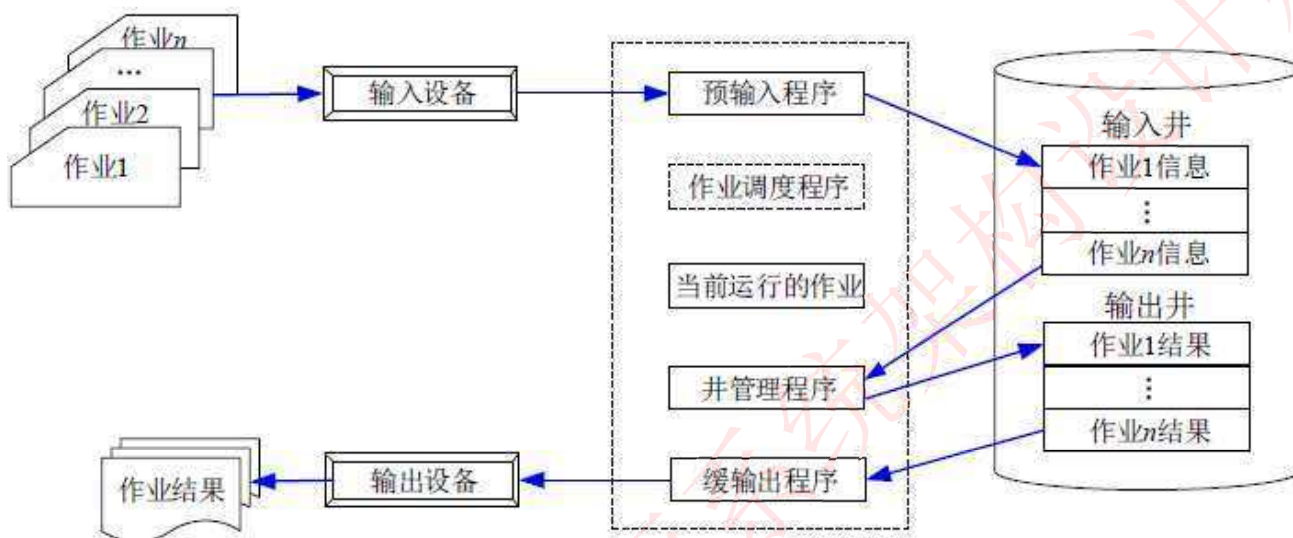
当用户程序试图读一个硬盘文件时，需要通过操作系统实现这一操作。与设备无关软件检查高速缓存中是否有要读的数据块，若没有，则调用设备驱动程序，向 I/O 硬件发出一个请求。然后，用户进程阻塞并等待磁盘操作的完成。当磁盘操作完成时，硬件产生一个中断，转入中断处理程序。中断处理程序检查中断的原因，认识到这时磁盘读取操作已经完成，于是唤醒用户进程取回从磁盘读取的信息，从而结束此次 I/O 请求。用户进程在得到了所需的硬盘文件内容之，后继续运行。



4.3 虚设备和 SPOOLING 技术

一台实际的物理设备，例如打印机，在同一时间只能由一个进程使用，其他进程只能等待，且不知道什么时候打印机空闲，此时，极大的浪费了外设的工作效率。

引入 SPOOLING（外围设备联机操作）技术，就是在外设上建立两个数据缓冲区，分别称为输入井和输出井，这样，无论多少进程，都可以共用这一台打印机，只需要将打印命令发出，数据就会排队存储在缓冲区中，打印机会自动按顺序打印，实现了物理外设的共享，使得每个进程都感觉在使用一个打印机，这就是物理设备的虚拟化。如下图所示：



单双缓冲区问题、磁盘调度问题，在第一章视频课程里已经详细解析过。

5 文件管理

5.1 概述

文件（File）是具有符号名的、在逻辑上具有完整意义的一组相关信息项的集合。

信息项是构成文件内容的基本单位，可以是一个字符，也可以是一个记录，记录可以等长，也可以不等长。一个文件包括文件体和文件说明。文件体是文件真实的内容。文件说明是操作系统为了管理文件所用到的信息，包括文件名、文件内部标识、文件的类型、文件存储地址、文件的长度、访问权限、建立时间和访问时间等。

文件管理系统，就是操作系统中实现文件统一管理的一组软件和相关数据的集合，专门负责管理和存取文件信息的软件机构，简称文件系统。文件系统的功能包括按名存取；统一的用户接口；并发访问和控制；安全性控制；优化性能；差错恢复。

文件的类型：

- （1）按文件性质和用途可将文件分为系统文件、库文件和用户文件。
- （2）按信息保存期限分类可将文件分为临时文件、档案文件和永久文件。
- （3）按文件的保护方式分类可将文件分为只读文件、读/写文件、可执行文件和不保护文件。
- （4）UNIX 系统将文件分为普通文件、目录文件和设备文件（特殊文件）。

文件的逻辑结构可分为两大类：一是有结构的记录式文件，它是由一个以上的记录构成的文件，故又称为记录式文件；二是无结构的流式文件，它是由一串顺序字符流构成的文件。

文件的物理结构是指文件的内部组织形式，即文件在物理存储设备上的存放方法，包括：

- （1）连续结构。连续结构也称顺序结构，它将逻辑上连续的文件信息（如记录）依次存放在连续编号的物理块上。只要知道文件的起始物理块号和文件的长度，就可以很方便地进行文件的存取。

(2) 链接结构。链接结构也称串联结构，它是将逻辑上连续的文件信息（如记录）存放在不连续的物理块上，每个物理块设有一个指针指向下一个物理块。因此，只要知道文件的第一个物理块号，就可以按链指针查找整个文件。

(3) 索引结构。在采用索引结构时，将逻辑上连续的文件信息（如记录）存放在不连续的物理块中，系统为每个文件建立一张索引表。索引表记录了文件信息所在的逻辑块号对应的物理块号，并将索引表的起始地址放在与文件对应的文件目录项中。

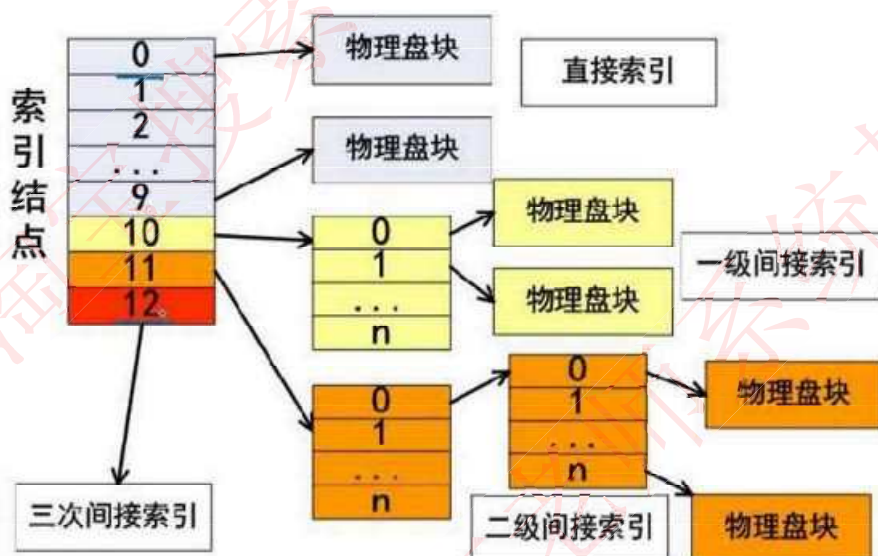
(4) 多个物理块的索引表。索引表是在文件创建时由系统自动建立的，并与文件一起存放在同一文件卷上。根据一个文件大小的不同，其索引表占用物理块的个数不等，一般占一个或几个物理块。

5.2 索引文件结构

如下图所示，系统中有 13 个索引节点，0-9 为直接索引，即每个索引节点存放的是内容，假设每个物理盘大小为 4KB，共可存 $4KB \times 10 = 10KB$ 数据；

10 号索引节点为一级间接索引节点，大小为 4KB，存放的并非直接数据，而是链接到直接物理盘块的地址，假设每个地址占 4B，则共有 1024 个地址，对应 1024 个物理盘，可存 $1024 \times 4KB = 4096KB$ 数据。

二级索引节点类似，直接盘存放一级地址，一级地址再存放物理盘快地址，而后链接到存放数据的物理盘块，容量又扩大了一个数量级，为 $1024 \times 1024 \times 4KB$ 数据。



5.3 文件目录

文件控制块中包含以下三类信息：基本信息类、存取控制信息类和使用信息类。

(1) 基本信息类。例如文件名、文件的物理地址、文件长度和文件块数等。

(2) 存取控制信息类。文件的存取权限，像 UNIX 用户分成文件主、同组用户和一般用户三类，这三类用户的读/写执行 RWX 权限。

(3) 使用信息类。文件建立日期、最后一次修改日期、最后一次访问的日期、当前使用的信息（如打开文件的进程数、在文件上的等待队列）等。

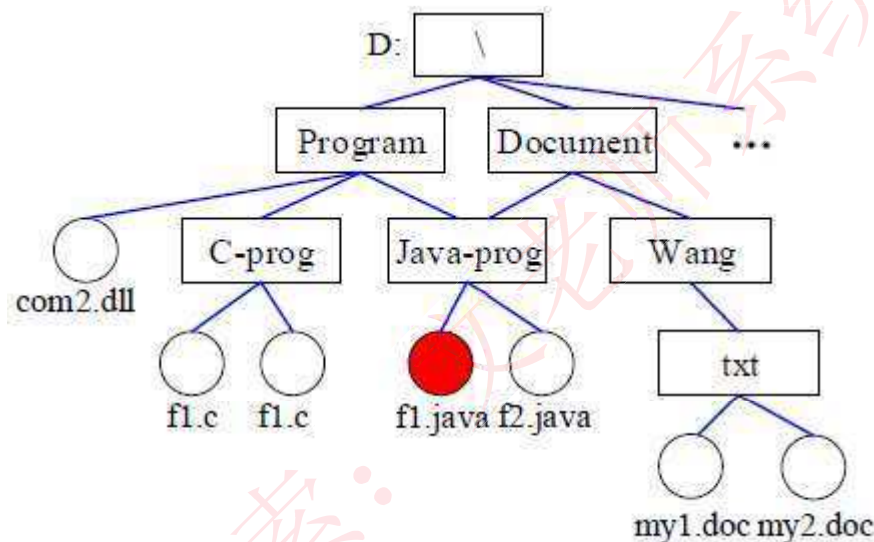
文件控制块的有序集合称为文件目录。

相对路径：是从当前路径开始的路径。

绝对路径：是从根目录开始的路径。

全文件名=绝对路径+文件名。要注意，绝对路径和相对路径是不加最后的文件名的，只是单纯的路径序列。

树形结构主要是区分**相对路径**和**绝对路径**，如下图所示：



5.4 文件存储空间管理

文件的存取方法是指读/写文件存储器上的一个物理块的方法。通常有顺序存取和随机存取两种方法。顺序存取方法是指对文件中的信息按顺序依次进行读/写；随机存取方法是指对文件中的信息可以按任意的次序随机地读/写。

文件存储空间的管理：

(1) 空闲区表。将外存空间上的一个连续的未分配区域称为“空闲区”。操作系统为磁盘外存上的所有空闲区建立一张空闲表，每个表项对应一个空闲区，适用于连续文件结构。

序号	第一个空闲块号	空闲块数	状态
1	18	5	可用
2	29	8	可用
3	105	19	可用
4	—	—	未用

(2) 位示图。这种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。

第 0 字节	1	0	1	0	0	1	...	0	1
第 1 字节	0	1	1	1	0	0	...	1	1
第 2 字节	0	1	1	1	1	0	...	1	0
第 3 字节	1	1	1	1	0	0	...	0	1
⋮	⋮								
第 n-1 字节	0	1	0	1	0	1	...	1	1

(3) 空闲块链。每个空闲物理块中有指向下一个空闲物理块的指针，所有空闲物理块构成一个链表，链表的头指针放在文件存储器的特定位置上（如管理块中），不需要磁盘分配表，节省空间。每次申请空闲物理块只需根据链表的头指针取出第一个空闲物理块，根据第一个空闲物理块的指针可找到第二个空闲物理块，依此类推。

(4) 成组链接法。例如，在实现时系统将空闲块分成若干组，每 100 个空闲块为一组，每组的第一个空闲块登记了下一组空闲块的物理盘块号和空闲块总数。假如某个组的第一个空闲块号等于 0，意味着该组是最后一组，无下一组空闲块。

三 数据库系统

数据库系统 DBS 的组成：数据库、硬件、软件、人员。

数据库管理系统 DBMS 的功能：数据定义、数据库操作、数据库运行管理、数据的存储管理、数据库的建立和维护等。

DBMS 的分类：关系数据库系统 RDBS、面向对象的数据库系统 OODBS、对象关系数据库系统 ORDBS。

数据库系统的体系结构：集中式数据库系统（所有东西集中在 DBMS 电脑上）、客户端/服务器体系结构（客户端负责请求和数据表示，服务器负责数据库服务）、并行数据库系统（多个物理上在一起的 CPU）、分布式数据库系统（物理上分布在不同地方的计算机）。

1 三级模式—两级映像

内模式：管理如何存储物理的**数据**，对数据的存储方式、优化、存放等。

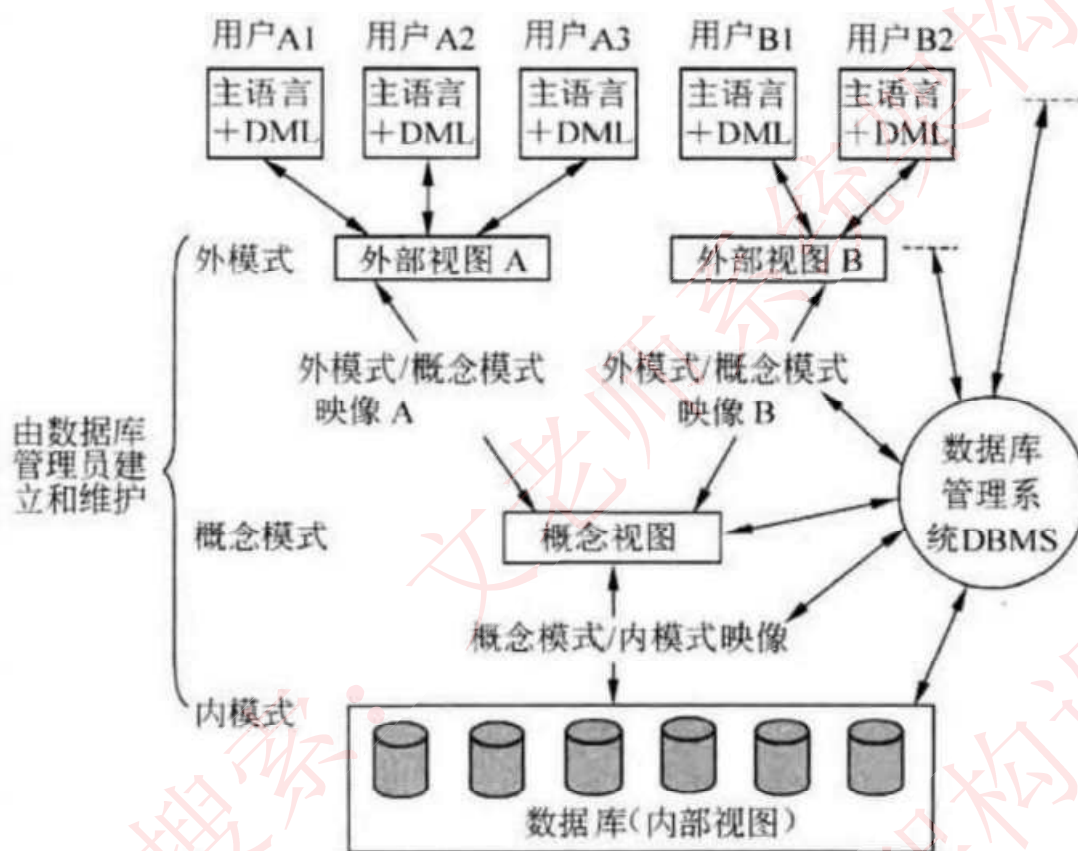
模式：又称为概念模式，就是我们通常使用的**表**这个级别，根据应用、需求将物理数据划分成一张张表。

外模式：对应数据库中的**视图**这个级别，将表进行一定的处理后再提供给用户使用，例如，将用户表中的用户名和密码组成视图提供给登录模块使用，而用户表中的其他列则不对该模块开放，增加了安全性。

外模式—模式映像：是表和视图之间的映射，存在于概念级和外部级之间，若表中数据发生了修改，只需要修改此映射，而无需修改应用程序。

模式—内模式映像：是表和数据物理存储之间的映射，存在于概念级和内部级之间，若修改了数据存储方式，只需要修改此映射，而不需要去修改应用程序。

以上的数据库系统实际上是一个分层次的设计，从底至上称为物理级数据库（实际为一个数据库文件）、概念级数据库、用户级数据库，各层情况如下：



2 数据库的设计

(1) 需求分析：即分析数据存储的要求，产出物有数据流图、数据字典、需求说明书。获得用户对系统的三个要求：信息要求、处理要求、系统要求。

(2) 概念结构设计：就是设计 E-R 图，也即实体-联系图，与物理实现无关，就是说明有哪些实体，实体有哪些属性。工作步骤包括：选择局部应用、逐一设计分 E-R 图、E-R 图合并。

分 E-R 图进行合并时，它们之间存在的冲突主要有以下 3 类。

属性冲突。同一属性可能会存在于不同的分 E-R 图中，由于设计人员不同或是出发点不同，对属性的类型、取值范围、数据单位等可能会不一致。

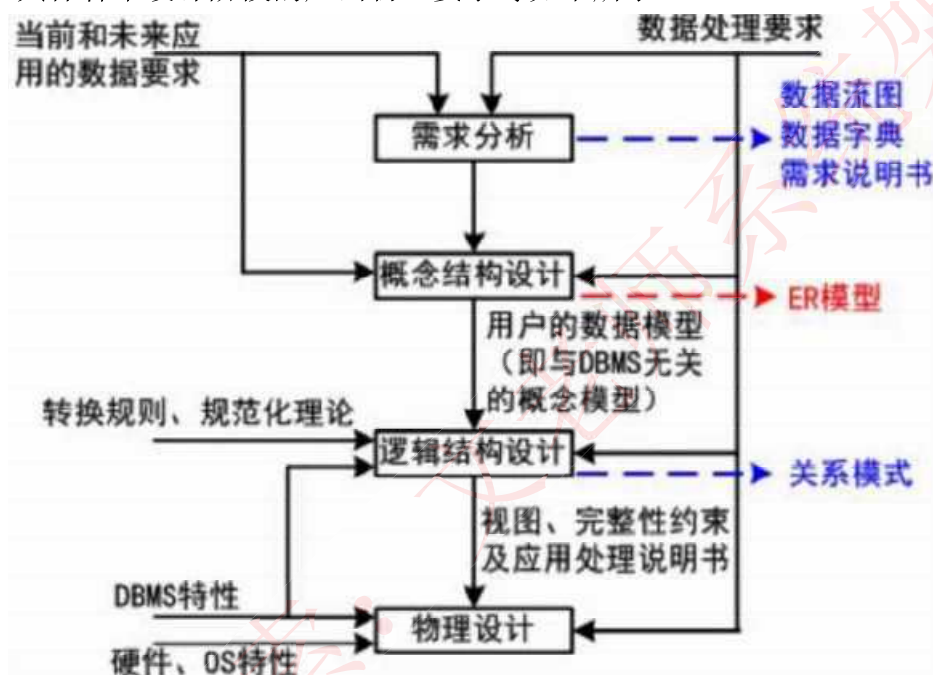
命名冲突。相同意义的属性，在不同的分 E-R 图上有着不同的命名，或是名称相同的属性在不同的分 E-R 图中代表着不同的意义。

结构冲突。同一实体在不同的分 E-R 图中有不同的属性，同一对象在某一分 E-R 图中被抽象为实体而在另一分 E-R 图中又被抽象为属性。

(3) 逻辑结构设计：将 E-R 图，转换成关系模式，也即转换成实际的表和表中的列属性，这里要考虑很多规范化的东西。工作步骤包括：确定数据模型、将 E-R 图转换成为指定的数据模型、确定完整性约束和确定用户视图。

(4) 物理设计：根据生成的表等概念，生成物理数据库。工作步骤包括确定数据分布、存储结构和访问方式。

具体各个设计阶段的产出物、要求等如下所示：



第二版教材在最后新增了两个阶段（但没有考点）：

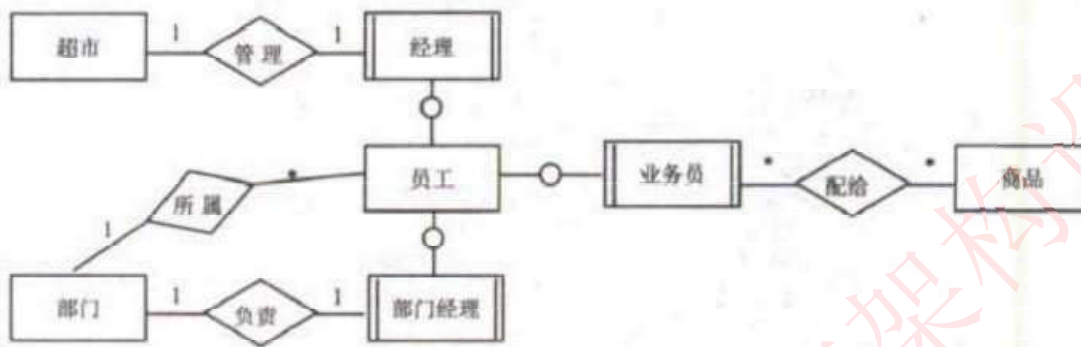
(5) 数据库实施阶段。数据库设计人员根据逻辑设计和物理设计阶段的结果建立数据库，编制与调试应用程序，组织数据入库，并进行试运行。

(6) 数据库运行和维护阶段。数据库应用系统经过试运行即可投入运行，但该阶段需要不断地对系统进行评价、调整与修改。

3 E-R 模型

数据模型的三要素：数据结构、数据操作、数据的约束条件。

在 E-R 模型中，使用椭圆表示属性（一般没有）、长方形表示实体、菱形表示联系，联系的两端要填写联系类型，示例如下图：



联系类型：一对一 1:1、一对多 1:N、多对多 M:N。

属性分类：简单属性和复合属性（属性是否可以分割）、单值属性和多值属性（属性有多个取值）、NULL 属性（无意义）、派生属性（可由其他属性生成）。

那么 **E-R 模型如何转换为关系模型**呢（实际就是转换为多少张表）？

每个实体都对应一个关系模式；

联系分为三种：**1:1** 联系中，联系可以放到任意的两端实体中，作为一个属性（**要保证 1:1 的两端关联**）；**1:N** 的联系中，联系可以单独作为一个关系模式，也可以在 **N** 端中加入 **1** 端实体的主键；**M:N** 的联系中，联系必须作为一个单独的关系模式，其主键是 **M** 和 **N** 端的联合主键。

以上，明确了有多少关系模式，就知道有多少张表，同时，表中的属性也确定了，注意联系是作为表还是属性，若是属性又是哪张表的属性即可。

4 关系代数运算

关系模式在代数运算时可以理解为数据库中的表，两个概念通用。

并：结果是两张表中所有记录数合并，相同记录只显示一次。

交：结果是两张表中相同的记录。

差： $S1-S2$ ，结果是 $S1$ 表中有而 $S2$ 表中没有的那些记录。

设有 $S1$ 和 $S2$ 关系如下图，其并交差结果如下图：

关系S1			关系S2		
Sno	Sname	Sdept	Sno	Sname	Sdept
No0001	Mary	IS	No0001	Mary	IS
No0003	Candy	IS	No0008	Katter	IS
No0004	Jam	IS	No0021	Tom	IS

S1∩S2 (交)			S1 ∪ S2 (并)		
Sno	Sname	Sdept	Sno	Sname	Sdept
No0001	Mary	IS	No0001	Mary	IS
			No0003	Candy	IS
			No0004	Jam	IS
			No0008	Katter	IS
			No0021	Tom	IS

S1 - S2 (差)		
Sno	Sname	Sdept
No0003	Candy	IS
No0004	Jam	IS

笛卡尔积： $S1 \times S2$ ，产生的结果包括 $S1$ 和 $S2$ 的所有属性列，并且 $S1$ 中每条记录依次和 $S2$ 中所有记录组合成一条记录，最终属性列为 $S1+S2$ 属性列，记录数为 $S1 \times S2$ 记录数。

投影：实际是按条件选择某关系模式中的某列，列也可以用数字表示。

选择：实际是按条件选择某关系模式中的某条记录。

设有 $S1$ 和 $S2$ 关系如下图，其笛卡尔积、投影、选择结果如下图：

S1 × S2 (笛卡尔积)					
Sno	Sname	Sdept	Sno	Sname	Sdept
No0001	Mary	IS	No0001	Mary	IS
No0001	Mary	IS	No0008	Katter	IS
No0001	Mary	IS	No0021	Tom	IS
No0003	Candy	IS	No0001	Mary	IS
No0003	Candy	IS	No0008	Katter	IS
No0003	Candy	IS	No0021	Tom	IS
No0004	Jam	IS	No0001	Mary	IS
No0004	Jam	IS	No0008	Katter	IS
No0004	Jam	IS	No0021	Tom	IS

关系S1		
Sno	Sname	Sdept
No0001	Mary	IS
No0003	Candy	IS
No0004	Jam	IS

关系S2		
Sno	Sname	Sdept
No0001	Mary	IS
No0008	Katter	IS
No0021	Tom	IS

(投影)	
Sno	Sname
No0001	Mary
No0003	Candy
No0004	Jam

(选择)		
Sno	Sname	Sdept
No0003	Candy	IS

自然连接：结果显示全部的属性列，但是相同属性列只显示一次，显示两个关系模式中属性相同且值相同的记录。自然联接结果如下：

关系S1		
Sno	Sname	Sdept
No0001	Mary	IS
No0003	Candy	IS
No0004	Jam	IS

关系S2	
Sno	Age
No0001	23
No0008	21
No0021	22

S1 ⋈ S2			
Sno	Sname	Sdept	Age
No0001	Mary	IS	23

效率问题：关系代数运算的效率，归根结底是看参与运算的两张表格的属性列数和记录数，属性列和记录数越少，参与运算的次数自然越少，效率就越高。因此，效率高的运算一般都是在两张表格参与运算之前就将条件判断完。如：

$\pi_{1,2,3,8} (\sigma_{2='大数据' \wedge 1=5 \wedge 3=6 \wedge 8='开发平台'} (R \times S))$ 和

$\pi_{1,2,3,8} (\sigma_{1=5 \wedge 3=6} (\sigma_{2='大数据'} (R) \times \sigma_{4='开发平台'} (S)))$ 。

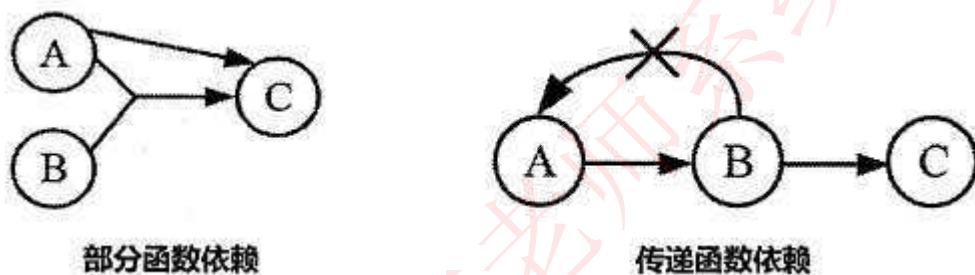
后者效率比前者效率高很多。

5 关系数据库的规范化

5.1 函数依赖

给定一个 X ，能唯一确定一个 Y ，就称 X 确定 Y ，或者说 Y 依赖于 X ，例如 $Y=X*X$ 函数。

函数依赖又可扩展以下两种规则：



部分函数依赖：A 可确定 C，(A,B)也可确定 C，(A,B)中的一部分（即 A）可以确定 C，称为部分函数依赖。

传递函数依赖：当 A 和 B 不等价时，A 可确定 B，B 可确定 C，则 A 可确定 C，是传递函数依赖；若 A 和 B 等价，则不存在传递，直接就可确定 C。

Armstrong 公理系统

设 U 是关系模式 R 的属性集， F 是 R 上成立的只涉及 U 中属性的函数依赖集。函数依赖的推理规则有：

自反律：若属性集 Y 包含于属性集 X ，属性集 X 包含于 U ，则 $X \rightarrow Y$ 在 R 上成立。(此处 $X \rightarrow Y$ 是平凡函数依赖)

增广律：若 $X \rightarrow Y$ 在 R 上成立，且属性集 Z 包含于属性集 U ，则 $XZ \rightarrow YZ$ 在 R 上成立。

传递律：若 $X \rightarrow Y$ 和 $Y \rightarrow Z$ 在 R 上成立，则 $X \rightarrow Z$ 在 R 上成立。

合并规则：若 $X \rightarrow Y$ ， $X \rightarrow Z$ 同时在 R 上成立，则 $X \rightarrow YZ$ 在 R 上也成立。

分解规则：若 $X \rightarrow W$ 在 R 上成立，且属性集 Z 包含于 W ，则 $X \rightarrow Z$ 在 R 上也成立。

伪传递规则：若 $X \rightarrow Y$ 在 R 上成立，且 $WY \rightarrow Z$ ，则 $XW \rightarrow Z$ 。

5.2 键和约束

超键：能唯一标识此表的属性的组合。

候选键：超键中去掉冗余的属性，剩余的属性就是候选键。

主键：任选一个候选键，即可作为主键。

外键：其他表中的主键。

候选键的求法：根据依赖集画出有向图，从入度为 0 的节点开始，找出图中一个节点或者一个节点组合，能够遍历完整个图，就是候选键。

主属性：候选键内的属性为主属性，其他属性为非主属性。

实体完整性约束：即主键约束，主键值不能为空，也不能重复。

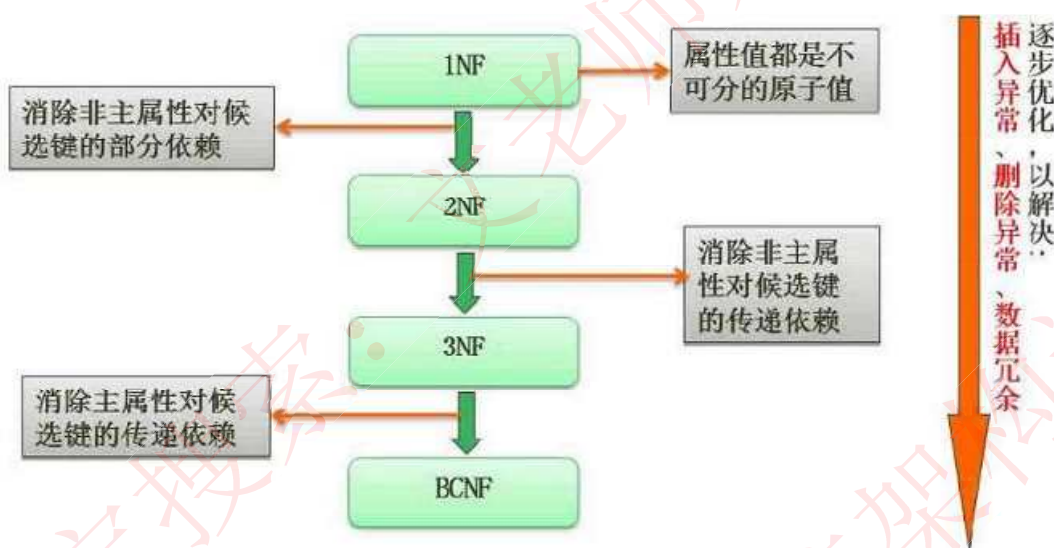
参照完整性约束：即外键约束，外键必须是其他表中已经存在的主键的值，或者为空。

用户自定义完整性约束：自定义表达式约束，如设定年龄属性的值必须在 0 到 150 之间。

触发器：通过写脚本来规定复杂的约束。本质属于用户自定义完整性约束。

5.3 范式

数据库中的范式总体概论如下图：



第一范式 1NF

关系中的每一个分量必须是一个不可分的数据项。通俗地说，第一范式就是表中不允许有小表的存在。比如，对于如下的员工表，就不属于第一范式：

员工编号	员工姓名	出生日期	薪资/月		所属部门
			基本工资/月	补贴/月	
1	王红	19900908	9000	1000	101
...	...	...	...	...	...

上表中，出现了属性薪资又被分为基本工资和补贴两个子属性，就好像表中有分割了一个小表，这就不属于第一范式。如果将基本工资和补贴合并，那么该表符合 1NF。

1NF 可能存在的问题：1NF 是最低一级的范式，范式程度不高，存在很多的问题。比如用一个单一的关系模式学生来描述学校的教务系统：学生(学号,学生姓名,系名,系主任姓名,课程号,成绩)

学号	学生姓名	所在系	系主任姓名	课程号	成绩
201102	张明	计算机系	章三	04	70
201103	王红	计算机系	章三	05	60
201103	王红	计算机系	章三	04	80
201103	王红	计算机系	章三	06	87
201104	李青	机械系	王五	09	79
...	...	...	...	...	...

这个表满足第一范式，但是存在如下问题：

数据冗余：一个系有很多的学生，同一个系的学生的系主任是相同的，所以系主任名会重复出现。

更新复杂：当一个系换了一个系主任后，对应的这个表必须修改与该系学生有关的每个元组。

插入异常：如果一个系刚成立，没有任何学生，那么这个无法把这个系的信息插入表中。

删除异常：如果一个系的学生都毕业了，那么在删除该系学生信息时，这个系的信息也丢了。

第二范式 2NF

如果关系 R 属于 1NF，且每一个非主属性完全函数依赖于任何一个候选码，则 R 属于 2NF。通俗地说，2NF 就是在 1NF 的基础上，表中的每一个非主属性不会依赖复合主键中的某一个列。

按照定义，上面的学生表就不满足 2NF，因为学号不能完全确定课程号和成绩(每个学生可以选择多门课)。将学生表分解为：

学生(学号,学生姓名,系名,系主任)

选课(学号,课程号,成绩)。

每张表均属于 2NF。

第三范式 3NF

在满足 2NF 的基础上，表中不存在非主属性对码的传递依赖。

继续上面的实例，学生关系模式就不属于 3NF，因为学生无法直接决定系主任和系名，是由学号->系名，再由系名->系主任，因此存在非主属性对主属性的传递依赖，将学生表进一步分解为：

学生(学号，学生姓名，系名)

系(系名，系主任)

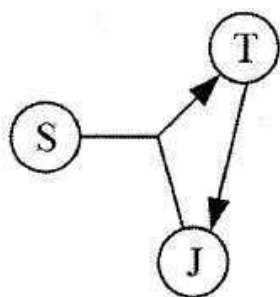
选课(学号,课程号,成绩)

每张表都属于 3NF。

BC 范式 BCNF

所谓 BCNF，是指在第三范式的基础上进一步消除主属性对于码的部分函数依赖和传递依赖。

通俗的来说，就是在每一种情况下，每一个依赖的左边决定因素都必然包含候选键，如下：



上图中，候选键有两种情况：组合键(S,T)或者(S,J)，依赖集为{SJ→T, T→J}，可知，STJ 三个属性都是主属性，因此其达到了 3NF（无非主属性），然而，第二种情况，即(S,J)为候选键的时候，对于依赖 T→J，T 在这种情况下不是候选键，即 T→J 的决定因素不包含任意候选码，因此上图不是 BCNF。

要使上图关系模式转换为 BCNF 也很简单，只需要将依赖 T→J 变为 TS→J 即可，这样其左边决定因素就包含了候选键之一 S。

第四范式 4NF

若关系模式 $R(U)$ 中， $X、Y、Z$ 是 U 的子集，并且 $Z=U-X-Y$ 。当且仅当对 $R(U)$ 的任何一个关系 T ，给定一对 (x,z) 值，有一组 Y 的值，这组值仅仅决定于 x 值而与 z 值无关，则称“ Y 多值依赖于 X ”或“ X 多值决定 Y ”成立。记为： $X \twoheadrightarrow Y$ 。（若 Z 为空集是平凡的多值依赖， Z 不为空是非平凡）

多值依赖具有如下 6 条性质：

- 多值依赖具有对称性。即若 $X \twoheadrightarrow Y$ ，则 $X \twoheadrightarrow Z$ ，其中 $Z=U-X-Y$ 。
- 多值依赖的传递性。即若 $X \twoheadrightarrow Y, Y \twoheadrightarrow Z$ ，则 $X \twoheadrightarrow Z-Y$ 。
- 函数依赖可以看成是多值依赖的特殊情况。
- 若 $X \twoheadrightarrow Y, X \twoheadrightarrow Z$ ，则 $X \twoheadrightarrow YZ$ 。
- 若 $X \twoheadrightarrow Y, X \twoheadrightarrow Z$ ，则 $X \twoheadrightarrow Y \cap Z$ 。
- 若 $X \twoheadrightarrow Y, X \twoheadrightarrow Z$ ，则 $X \twoheadrightarrow Z-Y$ 。

【定义 7.14】 关系模式 $R \in 1NF$ ，若对于 R 的每个非平凡多值依赖 $X \twoheadrightarrow Y$ ，且 $Y \not\subseteq X$ 时， X 必含有码，则关系模式 $R(U,F) \in 4NF$ 。

4NF 是限制关系模式的属性间不允许有非平凡且非函数依赖的多值依赖。

注意：如果只考虑函数依赖，关系模式最高的规范化程度是 BCNF；如果考虑多值依赖，关系模式最高的规范化程度是 4NF。

根据 4NF 的定义可知，4NF 所允许的非平凡的多值依赖实际上就是函数依赖，4NF 就是消除表中的非平凡多值依赖关系。

实例：有一个用户联系方式表 TELEPHONE(CUSTOMERID,PHONE,CELL)。

CUSTOMERID 为用户 ID,PHONE 为用户的固定电话,CELL 为用户的移动电话。

这是一个非常简单的 BCNF。主键为 CUSTOMERID。如果只有一个固定电话和移动电话，这个表就没问题。但是，如果用户有两个固定电话，两个移动电话。这时，表的具体表示如下：

CUSTOMERID	PHONE	CELL
1000	8828-1234	149088888888
1000	8838-1234	149099999999

由于 PHONE 和 CELL 是互相独立的，而有些用户又有两个和多个值。这时此表就违反第四范式，因为其非主属性互相独立且含有多值。

在这种情况下，此表的设计就会带来很多维护上的麻烦。例如，如果用户放弃第一行的固定电话和第二行的移动电话，那么这两行会合并吗？

解决问题的方法为，设计一个新表 NEW_PHONE(CUSTOMERID,NUMBER,TYPE).这样就可以对每个用户处理不同类型的多个电话号码，而不会违反第四范式。

显然，第四范式的应用范围比较小，因为只有在某些特殊情况下，要考虑将表规范到第四范式。所以在实际应用中，一般不要求表满足第四范式。

5.4 模式分解

范式之间的转换一般都是通过拆分属性，即模式分解，将具有部分函数依赖和传递依赖的属性分离出来，来达到一步步优化，一般分为以下两种：

保持函数依赖分解

对于关系模式 R，有依赖集 F，若对 R 进行分解，分解出来的多个关系模式，保持原来的依赖集不变，则为保持函数依赖的分解。另外，注意要消除掉冗余依赖（如传递依赖）。

实例：设原关系模式 R(A,B,C)，依赖集 F(A->B, B->C, A->C)，将其分解为两个关系模式 R1(A,B)和 R2(B,C)，此时 R1 中保持依赖 A->B，R2 保持依赖 B->C，说明分解后的 R1 和 R2 是保持函数依赖的分解，因为 A->C 这个函数依赖实际是一个冗余依赖，可以由前两个依赖传递得到，因此不需要管。

保持函数依赖的判断（补充，第 2 点不强求）：

1、如果 F 上的每一个函数依赖都在其分解后的某一个关系上成立，则这个分解是保持依赖的（这是一个充分条件）。也即我们课堂上说的简单方法，看函数每个依赖的左右两边属性是否都在同一个分解的模式中。

2、如果上述判断失败，并不能断言分解不是保持依赖的，还要使用下面的通用方法来做进一步判断。

该方法的表述如下：

算法二：

对 F 上的每一个 $\alpha \rightarrow \beta$ 使用下面的过程：

result:= α ;

while(result 发生变化)do

for each 分解后的 Ri

t=(result $\cap$ Ri)+ $\cap$ Ri

result=result $\cup$ t

以下面的例题纠错作为讲解：正确答案是无损分解，不保持函数依赖。

假设关系模式 $R(U, F)$, 属性集 $U=\{A, B, C\}$, 函数依赖集 $F=\{A \rightarrow B, B \rightarrow C\}$ 。若将其分解为 $p=\{R_1(U_1, F_1), R_2(U_2, F_2)\}$, 其中 $U_1=\{A, B\}, U_2=\{A, C\}$ 。那么, 关系模式 R, R_1, R_2 分别达到了 (5); 分解 D (6)。

(5) A. 1NF、2NF、3NF B. 1NF、3NF、3NF C. 2NF、2NF、3NF D. 2NF、3NF、3NF

(6) A. 有损连接但保持函数依赖 B. 既无损连接又保持函数依赖
C. 有损连接且不保持函数依赖 D. 无损连接但不保持函数依赖

【答案】D B

首先, 该分解, U_1 保持了依赖 $A \rightarrow B$, 然而 $B \rightarrow C$ 没有保持, 因此针对 $B \rightarrow C$ 需要用第 2 点算法来判断:

$result=B$, $result \cap U_1=B$, $B^+ = BC$, $BC \cap U_1=B$, $result=B \cup B=B$, $result$ 没变, 然后, $result$ 再和 U_2 交是空, 结束了, 不保持函数依赖。

注意, 这里 B^+ , + 的意思是代表由 B 能够推导出的其他所有属性的集合, 这里, $B \rightarrow C$, 因此 $B^+ = BC$ 。

无损分解

分解后的关系模式能够还原出原关系模式, 就是无损分解, 不能还原就是有损。

当分解为两个关系模式, 可以通过以下定理判断是否无损分解:

定理: 如果 R 的分解为 $p=\{R_1, R_2\}$, F 为 R 所满足的函数依赖集合, 分解 p 具有无损连接性的充分必要条件是 $R_1 \cap R_2 \rightarrow (R_1 - R_2)$ 或者 $R_1 \cap R_2 \rightarrow (R_2 - R_1)$ 。

当分解为多个关系模式时, 通过表格法求解:

思考题：

有关系模式：成绩（学号，姓名，课程号，课程名，分数）

函数依赖：学号→姓名，课程号→课程名，（学号，课程号）→分数

若将其分解为：

成绩（学号，课程号，分数）

学生（学号，姓名）

课程（课程号，课程名）

请思考该分解是否为无损分解？

由于有：学号→姓名，所以：

成绩（学号，课程号，分数，姓名）

由于有：课程号→课程名，所以：

成绩（学号，课程号，分数，姓名，课程名）

由思考题可知，无损分解，要注意将候选键和其能决定的属性放在一个关系模式中，这样才能还原。也可以用表格法求解如下（需要依赖左右边的属性同时在一个关系模式中，才能补充）：

初始表如下：

	学号	姓名	课程号	课程名	分数
成绩	√	×	√	×	√
学生	√	√	×	×	×
课程	×	×	√	√	×

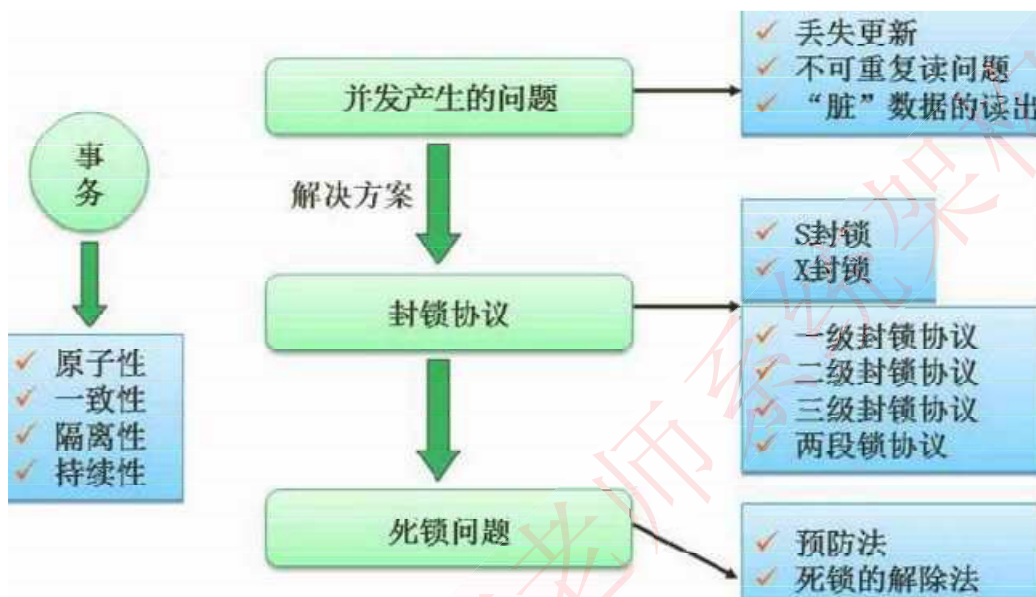
根据学号→姓名，对上表进行处理，将×改成符号√；然后考虑课程号→课程名，将×改为√，得下表：

	学号	姓名	课程号	课程名	分数
成绩	√	√	√	√	√
学生	√	√	×	×	×
课程	×	×	√	√	×

从上图中可以看出，第1行已全部为√，因此本次R分解是无损联接分解。

注意：拆分成单属性集必然是有损分解，因为单属性不可能包含依赖左右两边属性，这个单属性已经无法再恢复。

5.5 并发控制基本概念图



5.6 事务管理

事务提交 commit，事务回滚 rollback。

事务：由一系列操作组成，这些操作，要么全做，要么全不做，拥有四种特性，详解如下：

（操作）**原子性**：要么全做，要么全不做。

（数据）**一致性**：事务发生后数据是一致的，例如银行转账，不会存在 A 账户转出，但是 B 账户没收到的情况。

（执行）**隔离性**：任一事务的更新操作直到其成功提交的整个过程对其他事务都是不可见的，不同事务之间是隔离的，互不干涉。

（改变）**持续性**：事务操作的结果是持续性的。

事务是并发控制的前提条件，并发控制就是控制不同的事务并发执行，提高系统效率，但是并发控制中存在下面三个问题：

丢失更新：事务 1 对数据 A 进行了修改并写回，事务 2 也对 A 进行了修改并写回，此时事务 2 写回的数据会覆盖事务 1 写回的数据，就丢失了事务 1 对 A 的更新。即对数据 A 的更新会被覆盖。

不可重复读：事务 2 读 A，而后事务 1 对数据 A 进行了修改并写回，此时若事务 2 再读 A，发现数据不对。即一个事务重复读 A 两次，会发现数据 A 有误。

读脏数据：事务 1 对数据 A 进行了修改后，事务 2 读数据 A，而后事务 1 回滚，数据 A 恢复了原来的值，那么事务 2 对数据 A 做的事是无效的，读到了脏数据。

◆ 丢失更新

T1	T2
①读A=10 ② ③A=A-5写回 ④	读A=10 A=A-8 写回

◆ 不可重复读

T1	T2
①读A=20 读B=30 求和=50 ② ③读A=70 读B=30 求和=100 (验算不对)	读A=20 A←A+50 写A=70

◆ 读“脏”数据

T1	T2
①读A=20 A←A+50 写回70 ② ③ROLLBACK A恢复为20	读A=70

5.7 封锁协议

X 锁是排它锁（写锁）。若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A，其他事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁。

S 锁是共享锁（读锁）。若事务 T 对数据对象 A 加上 S 锁，则只允许 T 读取 A，但不能修改 A，其他事务只能再对 A 加 S 锁（也即能读不能修改），直到 T 释放 A 上的 S 锁。

共分为三级封锁协议，定义如下：

一级封锁协议：事务在修改数据 R 之前必须先对其加 X 锁，直到事务结束才释放。可解决丢失更新问题。

二级封锁协议：一级封锁协议的基础上加上事务 T 在读数据 R 之前必须先对其加 S 锁，读完后即可释放 S 锁。可解决丢失更新、读脏数据问题。

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。可解决丢失更新、读脏数据、数据重复读问题。

三级封锁协议的应用

丢失更新加锁（一级封锁协议）：

T1	T2
① 对A加写锁 ② ③ 读A=10 ④ A=A-5写回 ⑤ 释放对A的写锁 ⑥ ⑦ ⑧	对A加写锁 等待 等待 等待 读A= 5 A=A-8 写回 释放对A的写锁

读脏数据加锁（二级封锁协议）：

T1	T2
① 对A加写锁 ② 读A=20 ③ $A \leftarrow A+50$ ④ 写回70 ⑤ ⑥ ROLLBACK ⑦ A恢复为20 ⑧	对A加读锁 等待 等待 读A=20 释放对A的读锁

不可重复读加锁（三级封锁协议）：

T1	T2
① 对A与B加S锁（读锁） 读A=20 读B=30 求和=50 ② ③ 读A=20 读B=30 求和=50 释放对A和B的读锁	对A加X锁（写锁） 注：由于A已加了读锁，所以等待 等待 等待 等待 等待 读A=20 $A \leftarrow A+50$ 写A=70 释放对A的写锁

两段锁协议

每个事务的执行可以分为两个阶段：生长阶段（加锁阶段）和衰退阶段（解锁阶段）。

加锁阶段：在该阶段可以进行加锁操作。在对任何数据进行读操作之前要申请并获得 S 锁，在进行写操作之前要申请并获得 X 锁。加锁不成功，则事务进入等待状态，直到加锁成功才继续执行。

解锁阶段：当事务释放了一个封锁以后，事务进入解锁阶段，在该阶段只能进行解锁操作不能再进行加锁操作。

两段封锁法可以这样来实现：事务开始后就处于加锁阶段，一直到执行 ROLLBACK 和 COMMIT 之前都是加锁阶段。ROLLBACK 和 COMMIT 使事务进入解锁阶段，即在 ROLLBACK 和 COMMIT 模块中 DBMS 释放所有封锁。

6 数据故障与备份

6.1 安全措施

措施	说明
用户标识和鉴定	最外层的安全保护措施，可以使用用户帐户、口令及随机数检验等方式
存取控制	对用户进行授权，包括操作类型（如查找、插入、删除、修改等动作）和数据对象（主要是数据范围）的权限。
密码存储和传输	对远程终端信息用密码传输
视图的保护	对视图进行授权
审计	使用一个专用文件或数据库，自动将用户对数据库的所有操作记录下来

6.2 数据故障

故障关系	故障原因	解决方法
事务本身的可预期故障	本身逻辑	在程序中预先设置Rollback语句
事务本身的不可预期故障	算术溢出、违反存储保护	由DBMS的恢复子系统通过日志，撤消事务对数据库的修改，回退到事务初始状态
系统故障	系统停止运转	通常使用检查点法
介质故障	外存被破坏	一般使用日志重做业务

6.3 数据备份

静态转储：即冷备份，指在转储期间不允许对数据库进行任何存取、修改操作；优点是快速的备份方法、容易归档（直接物理复制操作）；缺点是只能提供到某一时间点上的恢复，不能做其他工作，不能按表或按用户恢复。

动态转储：即热备份，在转储期间允许对数据库进行存取、修改操作，因此，转储和用户事务可并发执行；优点是可在表空间或数据库文件级备份，数据库仍可使用，可达到秒级恢复；缺点是不能出错，否则后果严重，若热备份不成功，所得结果几乎全部无效。

完全备份：备份所有数据。

增量备份：仅备份上一次完全备份之后变化的数据。

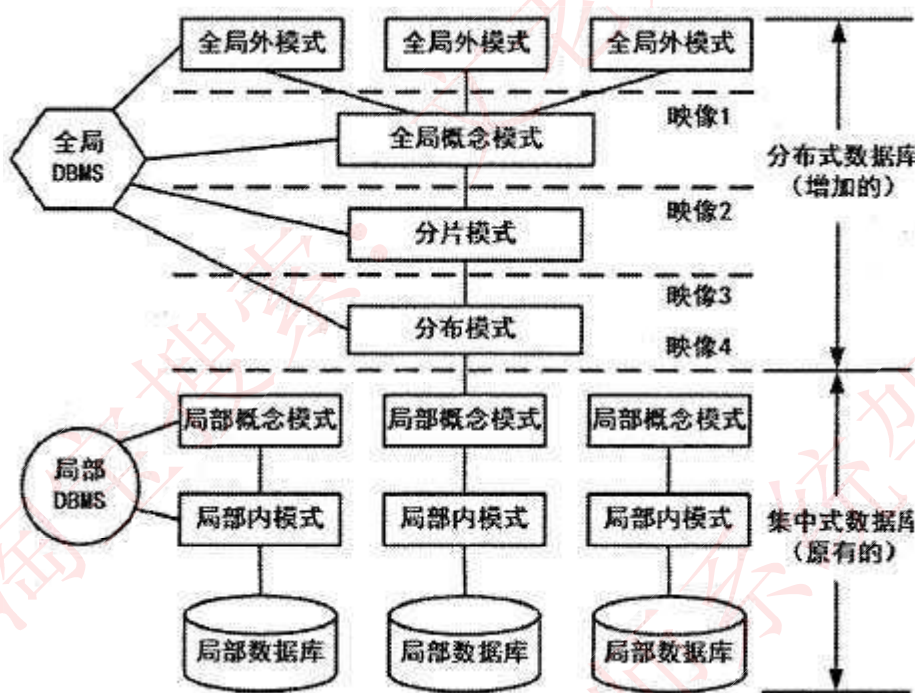
增量备份：备份上一次备份之后变化的数据。

日志文件：在事务处理过程中，DBMS 把事务开始、事务结束以及对数据库的插入、删除和修改的每一次操作写入日志文件。一旦发生故障，DBMS 的恢复子系统利用日志文件撤销事务对数据库的改变，回退到事务的初始状态。

备份毕竟是有时间节点的，不是实时的，例如：上一次备份到这次备份之间数据库出现了故障，则这期间的数据无法恢复，因此，引入日志文件，可以实时记录针对数据库的任何操作，保证数据库可以实时恢复。

7 分布式数据库

局部数据库位于不同的物理位置，使用一个全局 DBMS 将所有局部数据库联网管理，这就是分布式数据库。其体系结构如下图所示：



分片模式

水平分片：将表中水平的记录分别存放在不同的地方。

垂直分片：将表中的垂直的列值分别存放在不同的地方。

分布透明性

分片透明性：用户或应用程序不需要知道逻辑上访问的表具体是如何分块存储的。

位置透明性：应用程序不关心数据存储物理位置的改变。

逻辑透明性：用户或应用程序无需知道局部使用的是哪种数据模型。

复制透明性：用户或应用程序不关心复制的数据从何而来。

8 数据仓库与数据挖掘

数据仓库是一种特殊的数据库，也是按数据库形式存储数据的，但是目的不同：数据库经过长时间的运行，里面的数据会保存的越来越多，就会影响系统运行效率，对于某些程序而言，很久之前的数据并非必要的，因此，可以删除掉以减少数据，增加效率，考虑到删除这些数据比较可惜，因此，一般都把这些数据从数据库中提取出来保存到另外一个数据库中，称为数据仓库。

数据仓库四大特点

面向主题：按照一定的主题域进行组织的。

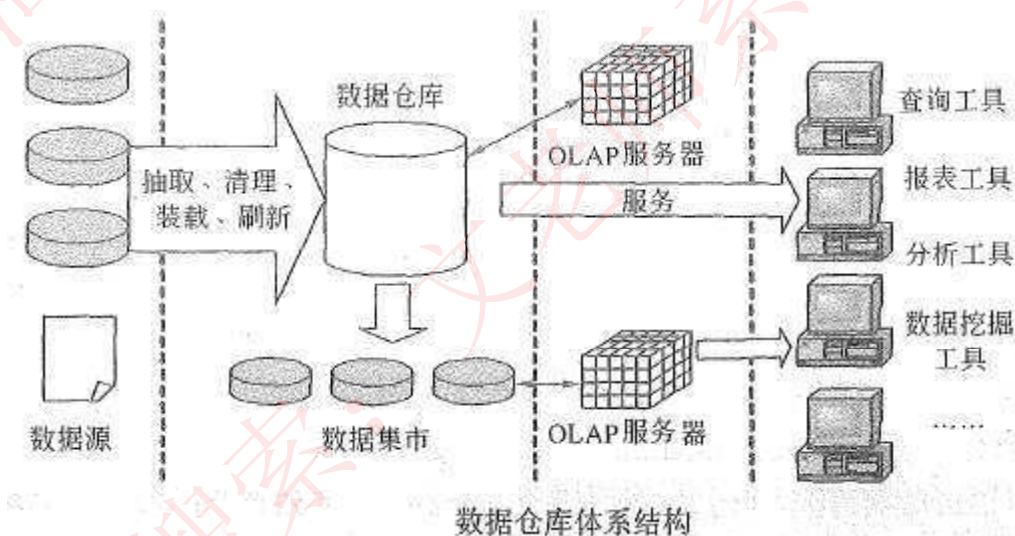
集成的：数据仓库中的数据是在对原有分散的数据库数据抽取、清理的基础上经过系统加工、汇总和整理得到的，必须消除源数据中的不一致性，以保证数据仓库内的信息是关于整个企业的一致的全局信息。

相对稳定的：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，通常只需要定期的加载、刷新。

反映历史变化：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。

数据仓库的结构通常包含四个层次，如下图所示：

- 1.数据源：是数据仓库系统的基础，是整个系统的数据源泉。
- 2.数据的存储与管理：是整个数据仓库系统的核心。
- 3.OLAP（联机分析处理）服务器：对分析需要的数据进行有效集成，按多维模型组织，以便进行多角度、多层次的分析，并发现趋势。
- 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、数据挖掘工具以及各种基于数据仓库或数据集市的应用开发工具。



数据挖掘的分析方法

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。

序列分析：序列分析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。

分类分析：分类分析通过分析具有类别的样本特点，得到决定样本属于各种类别的规则或方法。分类分析时首先为每个记录赋予一个标记（一组具有不同特征的类别），即按标记分类记录，然后检查这些标定的记录，描述出这些记录的特征。

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，并且对每个这样的组进行描述的过程。

商业智能 BI

BI 系统主要包括数据预处理、建立数据仓库、数据分析和数据展现四个主要阶段。

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（Extraction）、转换（Transformation）和加载（Load）三个过程（ETL 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（OLAP）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。

9 反规范化技术

由前面介绍可知，规范化操作可以防止插入异常、更新、删除异常和数据冗余，一般是通过模式分解，将表拆分，来达到这个目的。

但是表拆分后，解决了上述异常，却不利于查询，每次查询时，可能都要关联很多表，严重降低了查询效率，因此，有时候需要使用反规范化技术来提高查询效率。

技术手段包括：增加派生性冗余列，增加冗余列，重新组表，分割表。

主要就是增加冗余，提高查询效率，为规范化操作的逆操作。

10 SQL 语言

SQL 语言中的**语法关键字**，不区分大小写：

创建表 **create table**；

指定主键 **primary key()**；

指定外键 **foreign key()**；

修改表 **alter table**；

删除表 **drop table**;

索引 **index**, 视图 **view**;

数据库查询 **select...from...where**;

分组查询 **group by**, 分组时要注意 **select** 后的列名要适应分组, **having** 为分组查询附加条件;

更名运算 **as**;

字符串匹配 **like**, %匹配多个字符串, _匹配任意一个字符串;

数据库插入 **insert into...values()**;

数据库删除 **delete from...where**;

数据库修改 **update...set...where**;

排序 **order by**, 默认为升序, 降序要加关键字 **DESC**。

授权 **grant...on...to**, 允许其将权限再赋给另一用户 **with grant option**;

收回权限 **revoke...on...from**;

with check option 表示要检查 **where** 后的谓词条件。

DISTINCT: 过滤重复的选项, 只保留一条记录。

UNION: 出现在两个 **SQL** 语句之间, 将两个 **SQL** 语句的查询结果取或运算, 即值存在于第一句或第二句都会被选出。

INTERSECT: 对两个 **SQL** 语句的查询结果做与运算, 即值同时存在于两个语句才被选出。

SQL 语法原理

SELECT 之后的为要查询显示的属性列名; **FROM** 后面是要查询的表名; **WHERE** 后面是查询条件; 涉及到平均数、最大值、求和等运算, 必须要分组, **group by** 后面是分组的属性列名, 分组的条件使用 **Having** 关键字, 后面跟条件。

在 **SQL** 语句中, 条件判断时数字无需打引号, 字符串要打单引号。

11 NoSQL 数据库

NoSQL 最常见的解释是 **Non-Relational, Not Only SQL** 也被很多人接受。NoSQL 仅仅是一个概念, 泛指非关系型的数据库, 区别于关系数据库, 它们不保证关系数据的 **ACID** 特性。

按照所使用的数据结构类型, 一般可以将 **NoSQL** 数据库分为以下 4 种类型。

1 列式存储数据库: 行式数据库即传统的关系型数据库, 数据按记录存储, 每一条记录的所有属性存储在一行。列式数据库是按数据库记录的列来组织和存储数据的, 数据库中每个表由一组页链的集合组成, 每条页链对应表中的一个存储列。

2. 键值对存储数据库: 键值存储的典型数据结构一般为数组链表: 先通过 **Hash** 算法得出 **Hashcode**, 找到数组的某一个位置, 然后插入链表。

3. 文档型数据库: 文档型数据库同键值对存储数据库类似。该类型的数据模型是版本化的文档, 半结构化的文档以特定的格式存储, 比如 **JSON**。

4. 图数据库：图形结构的数据库同其他采用行列以及刚性结构的 SQL 数据库不同，它使用灵活的图形模型，并且能够扩展到多个服务器上。NoSQL 数据库没有标准的查询语言（SQL），因此进行数据库查询需要指定数据模型。

目前业界对于 NoSQL 并没有一个明确的范围和定义，但是它们普遍存在下面一些共同特征：

- 易扩展：去掉了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。
- 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。这得益于它的无关系性，数据库的结构简单。
- 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。
- 高可用：NoSQL 在不太影响性能的情况下，就可以方便地实现高可用的架构，有些产品通过复制模型也能实现高可用。

NoSQL 整体框架分为 4 层，由下至上分为数据持久层、数据分布层、数据逻辑模型层和接口层。

（1）数据持久层定义了数据的存储形式，主要包括基于内存、硬盘、内存和硬盘接口、订制可插拔 4 种形式。

（2）数据分布层定义了数据是如何分布的，相对于关系型数据库，NoSQL 可选的机制比较多，主要有 3 种形式：一是 CAP 支持，可用于水平扩展；二是多数据中心支持，可以保证在横跨多数数据中心时也能够平稳运行；三是动态部署支持，可以在运行着的集群中动态地添加或删除结点。

（3）数据逻辑层表述了数据的逻辑表现形式。

（4）接口层为上层应用提供了方便的数据调用接口，提供的选择远多于关系型数据库。

NoSQL 分层架构并不代表每个产品在每一层只有一种选择。相反，这种分层设计提供了很大的灵活性和兼容性，每种数据库在不同层面可以支持多种特性。

NoSQL 数据库在以下这几种情况比较适用：

- 数据模型比较简单：
- 需要灵活性更强的 IT 系统：
- 对数据库性能要求较高：
- 不需要高度的数据一致性：

对于给定 key，比较容易映射复杂值的环境。

四 嵌入式技术

1 第二版教材 2.4 嵌入式内容补充

嵌入式系统是以应用为中心、以计算机技术为基础，并将可配置与可裁减的软、硬件集成于一体的专用计算机系统，需要满足应用对功能、可靠性、成本、体积和功耗等方面的严格要求。

一般嵌入式系统由嵌入式处理器、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。

(1) 嵌入式处理器。由于嵌入式系统一般是在恶劣的环境条件下工作，与一般处理器相比，嵌入式处理器应可抵抗恶劣环境的影响，比如高温、寒冷、电磁、加速度等环境因素。为适应恶劣环境，嵌入式处理器芯片除满足低功耗、体积小等需求外，根据不同环境需求，其工艺可分为民用、工业和军用等三个档次。

(2) 相关支撑硬件。相关支撑硬件是指除嵌入式处理器以外的构成系统的其他硬件，包括存储器、定时器、总线、IO 接口以及相关专用硬件。

(3) 嵌入式操作系统。嵌入式操作系统是指运行在嵌入式系统中的基础软件，主要用于管理计算机资源和应用软件。与通用操作系统不同，嵌入式操作系统应具备实时性、可剪裁性和安全性等特征。

(4) 支撑软件。支撑软件是指为应用软件开发与运行提供公共服务、软件开发、调试能力的软件，支撑软件的公共服务通常运行在操作系统之上，以库的方式被应用软件所引用。

(5) 应用软件。应用软件是指为完成嵌入式系统的某一特定目标所开发的软件。

嵌入式系统应具备以下特性：

(1) 专用性强。嵌入式系统面向特定应用需求，能够把通用 CPU 中许多由板卡完成的任务集成在芯片内部，从而有利于嵌入式系统的小型化。

(2) 技术融合。嵌入式系统将先进的计算机技术、通信技术、半导体技术和电子技术与各个行业的具体应用相结合，是一个技术密集、资金密集、高度分散、不断创新的知识集成系统。

(3) 软硬一体软件为主。软件是嵌入式系统的主体，有 IP 核。嵌入式系统的硬件和软件都可以高效地设计，量体裁衣，去除冗余，可以在同样的硅片面积上实现更高的性能。

(4) 比通用计算机资源少。由于嵌入式系统通常只完成少数几个任务。设计时考虑到其经济性，不能使用通用 CPU，这就意味着管理的资源少，成本低，结构更简单。

(5) 程序代码固化在非易失存储器中。为了提高执行速度和系统可靠性，嵌入式系统中的软件一般都固化在存储器芯片或单片机本身中，而不是存在磁盘中。

(6) 需专门开发工具和环境。嵌入式系统本身不具备开发能力，即使设计完成以后，用户通常也不能对其中的程序功能进行修改，必须有一套开发工具和环境才能进行开发。

(7) 体积小、价格低、工艺先进、性能价格比高、系统配置要求低、实时性强。

(8) 对安全性和可靠性的要求高。

嵌入式系统分类

根据不同用途可将嵌入式系统划分为嵌入式实时系统和嵌入式非实时系统两种，而实时系统又可分为强实时系统和弱实时系统。如果从安全性要求看，嵌入式系统还可分为安全攸关系统和非安全攸关系统。

嵌入式系统分为硬件层、抽象层、操作系统层、中间件层和应用层等 5 层。

(1) 硬件层。硬件层主要是为嵌入式系统提供运行支撑的硬件环境，其核心是微处理器、存储器（ROM、SDRAM、Flash 等）、I/O 接口（A/D、D/A、I/O 等）和通用设备以及总线、电源、时钟等。

(2) 抽象层。在硬件层和软件层之间为抽象层，主要实现对硬件层的硬件进行抽象，为上层应用（操作系统）提供虚拟的硬件资源：板级支持包(BSP)是一种硬件驱动软件，它是面向硬件层的硬件芯片或电路进行驱动，为上层操作系统提供对硬件进行管理的支持。

(3) 操作系统层。操作系统层主要由嵌入式操作系统、文件系统、图形用户接口、网络系统和通用组件等可配置模块组成。

(4) 中间件层。中间件层一般位于操作系统之上，管理计算机资源和网络通信，中间件层是连接两个独立应用的桥梁。

(5) 应用层。应用层是指嵌入式系统的具体应用，主要包括不同的应用软件。

嵌入式软件的主要特点如下。

(1) 可剪裁性。嵌入式软件能够根据系统功能需求，通过工具进行适应性功能的加或减，删除系统不需要的软件模块，使得系统更加紧凑。

(2) 可配置性。嵌入式软件需要具备根据系统运行功能或性能需要而被配置的能力，使得嵌入式软件能够根据系统的不同状态、不同容量和不同流程，对软件工作状况进行能力的扩展、变更和增量服务。

(3) 强实时性。嵌入式系统中的大多数都属于强实时性系统，要求任务必须在规定的时限内处理完成，因此，嵌入式软件采用的算法优劣是影响实时性的主要原因。

(4) 安全性。安全性是指系统在规定的条件下和规定的时间内不发生事故的能力。

(5) 可靠性。可靠性是指系统在规定的条件下和规定的时间周期内程序执行所要求的功能的能力。

(6) 高确定性。嵌入式系统运行的时间、状态和行为是预先设计规划好的，其行为不能随时间、状态的变迁而变化

M2M 全称 Machine to Machine，是指数据从一台终端传送到另一台终端，也就是机器与机器的对话。**M2M 应用系统构成有智能化机器、M2M 硬件、通信网络、中间件。**

1、智能化机器

“智能化”，所谓使机器“开口说话”，让机器具有信息感知、信息加工及无线加工的能力。

2、M2M 硬件

使机器可具备联网能力和远程通信的部件，进行信息提取，从不同设备内汲取需要的信息，传输到分析部分。

3、通信网络

包括广域网(无线移动通讯网络、卫星通讯网络、互联网和公众电话网)，局域网(以太网、无线局域网、wifi)，个域网(Zigbee、传感器网络、蓝牙)，通过上述网络将 M2M 硬件传输的信息送达指定位置，是出于 M2M 技术框架的核心地位。

4、中间件

M2M 网关完成在不同协议之间的转换，在通信网络和 IT 系统之间建立桥梁。

DO-178B 的目的在于为制造机载系统和设备的机载软件提供指导，使其能够提供在满足符合适航要求的安全性水平下完成预期的功能的信心。为了满足该目标，DO-178B 给予了以下三方面的指导：

- a) 软件生命周期过程的目标；
- b) 为满足上述目标要进行的活动；
- c) 证明上述目标已经达到的证据，也即软件生命周期数据。

在 DO-178B 中，目标、过程、数据是软件适航的基本要求。

NPU：国产嵌入式神经网络处理器，是一种专门应用于网络应用数据包的处理器，采用“数据驱动并行计算”的架构，特别擅长处理视频、图像类的海量多媒体数据。

NPU 也是集成电路的一种，但区别于特殊用途集成电路 (ASIC) 的单一功能，网络处理更加复杂、更加灵活，一般可以利用软件或硬件依照网络运算的特性特别编程从而实现网络的特殊用途，在一块芯片上实现许多不同功能，以应用于多种不同的网络设备及产品。

NPU 的亮点在于能够运行多个并行线程 —— NPU 通过一些特殊的硬件级优化，比如为一些真正不同的处理核提供一些容易访问的缓存系统，将其提升到另一个层次。这些高容量内核比通常的“常规”处理器更简单，因为它们不需要执行多种类型的任务。这一整套的“优化”使得 NPU 更加高效，这就是为什么这么多的研发会投入到 ASIC 中的原因。

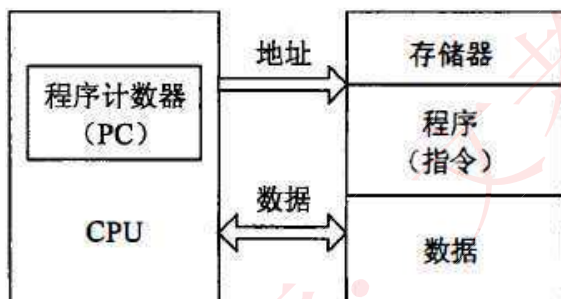
NPU 的优势之一在于，大部分时间集中在低精度的算法，新的数据流架构或内存计算能力。与 GPU 不同，它们更关注吞吐量而不是延迟。

2 嵌入式微处理器

2.1 嵌入式微处理器体系结构

冯诺依曼结构

传统计算机采用冯·诺依曼 (Von Neumann) 结构，也称普林斯顿结构，是一种将程序指令存储器和数据存储器合并在一起的存储器结构。



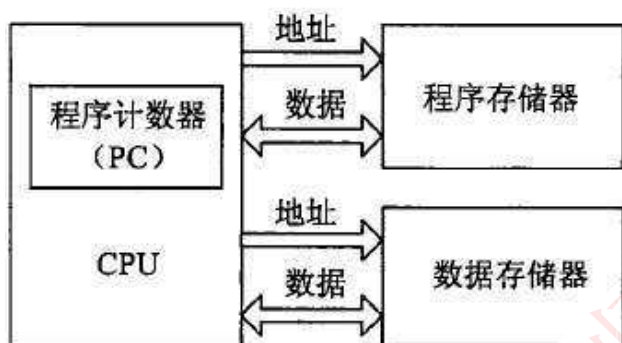
冯·诺依曼结构的计算机程序和数据共用一个存储空间，程序指令存储地址和数据存储地址指向同一个存储器的不同物理位置。

采用单一的地址及数据总线，程序指令和数据的宽度相同。

处理器执行指令时，先从储存器中取出指令解码，再取操作数执行运算，即使单条指令也要耗费几个甚至几十个周期，在高速运算时，在传输通道上会出现瓶颈效应。

哈佛结构

哈佛结构是一种并行体系结构，它的主要特点是将程序和数据存储在不同的存储空间中，即程序存储器和数据存储器是两个相互独立的存储器，每个存储器独立编址、独立访问。



与两个存储器相对应的是系统中的两套独立的地址总线 and 数据总线。

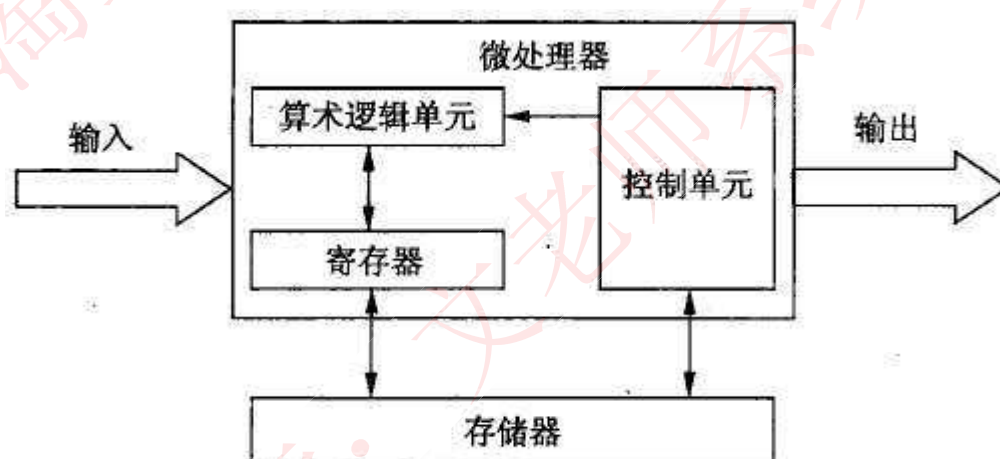
这种分离的程序总线 and 数据总线可允许在一个机器周期内同时获取指令字（来自程序存储器）和操作数（来自数据存储器），从而提高了执行速度，使数据的吞吐率提高了 1 倍。

但这并不意味着可以在一个机器周期内多次访问存储器。

2.2 嵌入式微处理器分类

嵌入式硬件系统基本结构如下图所示，一般由嵌入式微处理器、存储器、输入/输出部分组成，其中，嵌入式微处理是嵌入式硬件系统的核心，通常由 3 大部分组成：控制单元（控制器）、算术逻辑单元（运算器）、寄存器。

具体功能在嵌入式系统基础章节有详细讲解。



嵌入式微处理器的分类

根据嵌入式微处理器的字长宽度，可分为 4 位、8 位、16 位、32 位和 64 位。一般把 16 位及以下的称为嵌入式微控制器（Embedded Micro Controller），32 位及以上的称为嵌入式微处理器。

如果按系统集成度划分，可分为两类：一种是微处理器内部仅包含单纯的中央处理器单元，称为一般用途型微处理器；另一种则是将 CPU、ROM、RAM 及 I/O 等部件集成到同一个芯片上，称为单芯片微控制器（Single Chip Microcontroller）。

如果根据用途分类，一般分为嵌入式微控制器 MCU、嵌入式微处理器 MPU、嵌入式数字信号处理器 DSP、嵌入式片上系统 SOC 等。

嵌入式微控制器 MCU 的典型代表是单片机，其片上外设资源比较丰富，适合于控制。MCU 芯片内部集成 ROM/EPROM、RAM、总线、总线逻辑、定时/计数器、看门狗、I/O、串行口、脉宽调制输出、A/D、D/A、Flash RAM、EEPROM 等各种必要功能和外设。和嵌入式微处理器相比，微控制器的最大特点是单片化，体积大大减小，从而使功耗和成本下降、可靠性提高，其片上外设资源一般较丰富，适合于控制，是嵌入式系统工业的主流。

嵌入式微处理器 MPU 由通用计算机中的 CPU 演变而来。它的特征是具有 32 位以上的处理器，具有较高的性能，当然其价格也相应较高。但与计算机处理器不同的是，在实际嵌入式应用中，只保留和嵌入式应用紧密相关的功能硬件，去除其他的冗余功能部分，这样就以最低的功耗和资源实现嵌入式应用的特殊要求。与工业控制计算机相比，嵌入式微处理器具有体积小、重量轻、成本低、可靠性高的优点。目前常见的有 ARM、MIPS、POWER PC 等。

嵌入式数字信号处理器 DSP 是专门用于信号处理方面的处理器，其在系统结构和指令算法方面进行了特殊设计，具有很高的编译效率和指令的执行速度。采用哈佛结构，流水线处理，其处理速度比最快的 CPU 还快 10-50 倍。在数字滤波、FFT、谱分析等各种仪器上 DSP 获得了大规模的应用。

DSP 特点：

多总线结构，允许 CPU 同时进行指令和数据的访问，因而可以实现流水线操作。

哈佛体系结构，程序和数据空间分开，可以同时访问指令和数据。

数字信号处理的运算特点：乘/加，及反复相乘求和（乘积累加）。

DSP 设置了硬件乘法/累加器，能在单个指令周期内完成乘法/加法运算。

DSP 主要应用：信号处理、图像处理，仪器、语言处理、控制、军事、通讯、医疗、家用电器等领域。

嵌入式片上系统 soc，是追求产品系统最大包容的集成器件。SoC 最大的特点是成功实现了软硬件无缝结合，直接在处理器片内嵌入操作系统的代码模块。是一个有专用目标的集成电路，其中包含完整系统并有嵌入软件的全部内容。

2.3 多核处理器结构

多核指多个微处理器内核，是将两个或更多的微处理器封装在一起，集成在一个电路中。多核处理器是单枚芯片，能够直接插入单一的处理器插槽中。多核与多 CPU 相比，很好的降低了计算机系统的功耗和体积。在多核技术中，由操作系统软件进行调度，多进程，多线程并发都可以。

双核处理器：基于单个半导体的一个处理器上拥有 2 个处理器核心。

由于将 2 个或多个运算核封装在一个芯片上,节省大量晶体管、封装成本;显著提高处理器性能;兼容性好;系统升级方便。

2 个或多个内核工作协调实现方式:

对称多处理技术:将 2 颗完全一样的处理器封装在一个芯片内,达到双倍或接近双倍的处理性能,节省运算资源。

非对称处理技术:2 个处理内核彼此不同,各自处理和执行特定的功能,在软件的协调下分担不同的计算任务。

从目前已经发布或透露的多核处理器原型来看,对称式的处理方式将成为未来多核处理器的主要体系结构,同时,多核间将共享大容量的缓存作为处理器之间及处理器与系统内存之间交换数据的“桥梁”。为了提高交换速度,这些缓存往往集成在片内,其数据传输速度是惊人的。

多核 CPU 的调度

多核 CPU 环境下进程的调度算法一般有全局队列调度和局部队列调度两种。

全局队列调度是指操作系统维护一个全局的任务等待队列,当系统中有一个 CPU 空闲时,操作系统就从全局任务等待队列中选取就绪任务开始执行,CPU 核心利用率高。

局部队列调度是操作系统为每个 CPU 内核维护一个局部的任务等待队列,当系统中有一个 CPU 内核空闲时,就从该核心的任务等待队列中选取适当的任务执行,优点是无需在多个 CPU 之间切换。

3 嵌入式软件与操作系统

3.1 嵌入式软件基础

嵌入式软件是指应用在嵌入式计算机系统当中的各种软件,除了具有通用软件的一般特性,还有一些与嵌入式系统相关的特点,包括:规模较小、开发难度大、实时性和可靠性要求高、要求固化存储。

嵌入式软件分类

系统软件:控制和管理嵌入式系统资源,为嵌入式应用提供支持的各种软件,如设备驱动程序、嵌入式操作系统、嵌入式中间件等。

应用软件:嵌入式系统中的上层软件,定义了嵌入式设备的主要功能和用途,并负责与用户交互,一般面向特定的应用领域,如飞行控制软件、手机软件、地图等。

支撑软件:辅助软件开发的工具软件,如系统分析设计工具、在线仿真工具、交叉编译器等。

设备驱动层

又称为板级支持包 BSP,包含了嵌入式系统中所有与硬件相关的代码,直接与硬件打交道,对硬件进行管理和控制,并为上层软件提供所需的驱动支持。

板级支持包(BSP)是介于主板硬件和操作系统中驱动层程序之间的一层,一般认为它属于操作系统一部分,主要是实现对操作系统的支持,为上层的驱动程序提供访问硬件设备寄存器的函数包,使之能够更好的运行于硬件主板。在嵌入式系统软件的组成中,就有 BSP。BSP 是相对于操作系统而

言的,不同的操作系统对应于不同定义形式的 BSP,例如 VxWorks 的 BSP 和 Linux 的 BSP 相对于某一 CPU 来说尽管实现的功能一样,可是写法和接口定义是完全不同的,所以写 BSP 一定要按照该系统 BSP 的定义形式来写(BSP 的编程过程大多数是在某一个成型的 BSP 模板上进行修改)。这样才能与上层 OS 保持正确的接口,良好的支持上层 OS。

BSP 主要功能为屏蔽硬件,提供操作系统及硬件驱动,具体功能包括:

- 1.单板硬件初始化,主要是 CPU 的初始化,为整个软件系统提供底层硬件支持
- 2.为操作系统提供设备驱动程序和系统中断服务程序
- 3.定制操作系统的功能,为软件系统提供一个实时多任务的运行环境
- 4.初始化操作系统,为操作系统的正常运行做好准备。

板级支持包(BSP,也称为硬件抽象层 HAL)一般包含相关底层硬件的初始化、数据的输入/输出操作和硬件设备的配置等功能,它主要具有以下两个特点。

①硬件相关性,因为嵌入式实时系统的硬件环境具有应用相关性,而作为上层软件与硬件平台之间的接口,BSP 需为操作系统提供操作和控制具体硬件的方法。

②操作系统相关性,不同的操作系统具有各自的软件层次结构,因此不同操作系统具有特定的硬件接口形式。

一般来说,BSP 主要包括两个方面的内容:引导加载程序 BootLoader 和设备驱动程序。

引导加载程序

BootLoader 是嵌入式系统加电后运行的第一段软件代码,是在操作系统内核运行之前运行的一小段程序,通过这段程序,可以初始化硬件设备、建立内存空间的映射图,从而将系统的软硬件环境设置到一个合适的状态,以便为最终调用操作系统内核做好准备。一般包括以下功能:

片级初始化:主要完成微处理器的初始化,包括设置微处理器的核心寄存器和控制寄存器、微处理器的核心工作模式及其局部总线模式等。片级初始化把微处理器从上电时的默认状态逐步设置成系统所要求的工作状态。这是一个纯硬件的初始化过程。

板级初始化:通过正确地设置各种寄存器的内容来完成微处理器以外的其他硬件设备的初始化。例如,初始化 LED 显示设备、初始化定时器、设置中断控制寄存器、初始化串口通信、初始化内存控制器、建立内存空间的地址映射等。在此过程中,除了要设置各种硬件寄存器以外,还要设置某些软件的数据结构和参数。因此,这是一个同时包含有软件和硬件在内的初始化过程。

加载内核(系统级初始化):将操作系统和应用程序的映像从 Flash 存储器复制到系统的内存当中,然后跳转到系统内核的第一条指令处继续执行。

设备驱动程序

在一个嵌入式系统当中,操作系统是可能有也可能无的。但无论如何,设备驱动程序是必不可少的。所谓的设备驱动程序,就是一组库函数,用来对硬件进行初始化和管管理,并向上层软件提供良好的访问接口。

对于不同的硬件设备来说,它们的功能是不一样的,所以它们的设备驱动程序也是不一样的。但是一般来说,大多数的设备驱动程序都会具备以下的一些基本功能。

硬件启动:在开机上电或系统重启的时候,对硬件进行初始化。

硬件关闭：将硬件设置为关机状态。

硬件停用：暂停使用这个硬件。

硬件启用：重新启用这个硬件。

读操作：从硬件中读取数据。

写操作：往硬件中写入数据。

3.2 嵌入式操作系统

嵌入式操作系统(EOS, Embedded Operating System)是指用于嵌入式系统的操作系统。

嵌入式操作系统是一种用途广泛的系统软件，通常包括与硬件相关的底层驱动软件、系统内核、设备驱动接口、通信协议、图形界面、标准化浏览器等。

嵌入式操作系统负责嵌入式系统的全部软、硬件资源的分配、任务调度，控制、协调并发活动。它必须体现其所在系统的特征，能够通过装卸某些模块来达到系统所要求的功能。

目前在嵌入式领域广泛使用的操作系统有：嵌入式实时操作系统 μ C/OS-II、嵌入式 Linux、Windows Embedded、VxWorks 等，以及应用在智能手机和平板电脑的 Android、iOS 等。

嵌入式操作系统的特点

(1)系统内核小。由于嵌入式系统一般是应用于小型电子装置，系统资源相对有限，所以内核较之传统的操作系统要小得多。

(2)专用性强。嵌入式系统的个性化很强，其中的软件系统和硬件的结合非常紧密，一般要针对硬件进行系统的移植，即使在同一品牌、同一系列的产品中也需要根据系统硬件的变化和增减不断进行修改。同时针对不同的任务，往往需要对系统进行较大更改，程序的编译下载要和系统相结合，这种修改和通用软件的“升级”是完全两个概念。

(3)系统精简。嵌入式系统一般没有系统软件和应用软件的明显区分，不要求其功能设计及实现上过于复杂，这样一方面利于控制系统成本，同时也利于实现系统安全。

(4)高实时性。高实时性的系统软件是嵌入式软件的基本要求，而且软件要求固态存储，以提高速度；软件代码要求高质量和高可靠性。

(5)多任务的操作系统。嵌入式软件开发需要使用多任务的操作系统。嵌入式系统的应用程序可以没有操作系统直接在芯片上运行。但是为了合理地调度多任务、利用系统资源、系统函数以及和专用库函数接口，用户必须自行选配操作系统开发平台，这样才能保证程序执行的实时性、可靠性，并减少开发时间，保障软件质量。

3.3 嵌入式实时操作系统

嵌入式实时系统是一种完全嵌入受控器件内部，为特定应用而设计的专用计算机系统。在嵌入式实时系统中，要求系统在投入运行前即具有确定性和可预测性。

可预测性是指系统在运行之前，其功能、响应特性和执行结果是可预测的；

确定性是指系统在给定的初始状态和输入条件下，在确定的时间内给出确定的结果。

对嵌入式实时系统失效的判断，不仅依赖其运行结果的数值是否正确，也依赖提供结果是否及时。

实时操作系统(RTOS)的特点

当外界事件或数据产生时，能够接受并以足够快的速度予以处理，其处理的结果又能在规定的时

因而，提供**及时响应和高可靠性**是其主要特点。

实时操作系统有硬实时和软实时之分，硬实时要求在规定的时间内必须完成操作，这是在操作系统设计时保证的；软实时则只要按照任务的优先级，尽可能快地完成操作即可。

实时操作系统的特征

①高精度计时系统

计时精度是影响实时性的一个重要因素。在实时应用系统中，经常需要精确确定实时地操作某个设备或执行某个任务，或精确的计算一个时间函数。这些不仅依赖于一些硬件提供的时钟精度，也依赖于实时操作系统实现的高精度计时功能。

②多级中断机制

一个实时应用系统通常需要处理多种外部信息或事件，但处理的紧迫程度有轻重缓急之分。有的必须立即作出反应，有的则可以延后处理。因此，需要建立多级中断嵌套处理机制，以确保对紧迫程度较高的实时事件进行及时响应和处理。

③实时调度机制

实时操作系统不仅要及时响应实时事件中断，同时也要及时调度运行实时任务。但是，处理机调度并不能随心所欲的进行，因为涉及到两个进程之间的切换，只能在确保“安全切换”的时间点上进行，实时调度机制包括两个方面，一是在调度策略和算法上保证优先调度实时任务；二是建立更多“安全切换”时间点，保证及时调度实时任务。

因此，实际上来看，实时操作系统如同操作系统一样，就是一个后台的支撑程序，可以按照实时性的要求进行配置、裁剪等。其关注的重点在于任务完成的时间是否能够满足要求。

4 嵌入式软件设计

4.1 开发流程

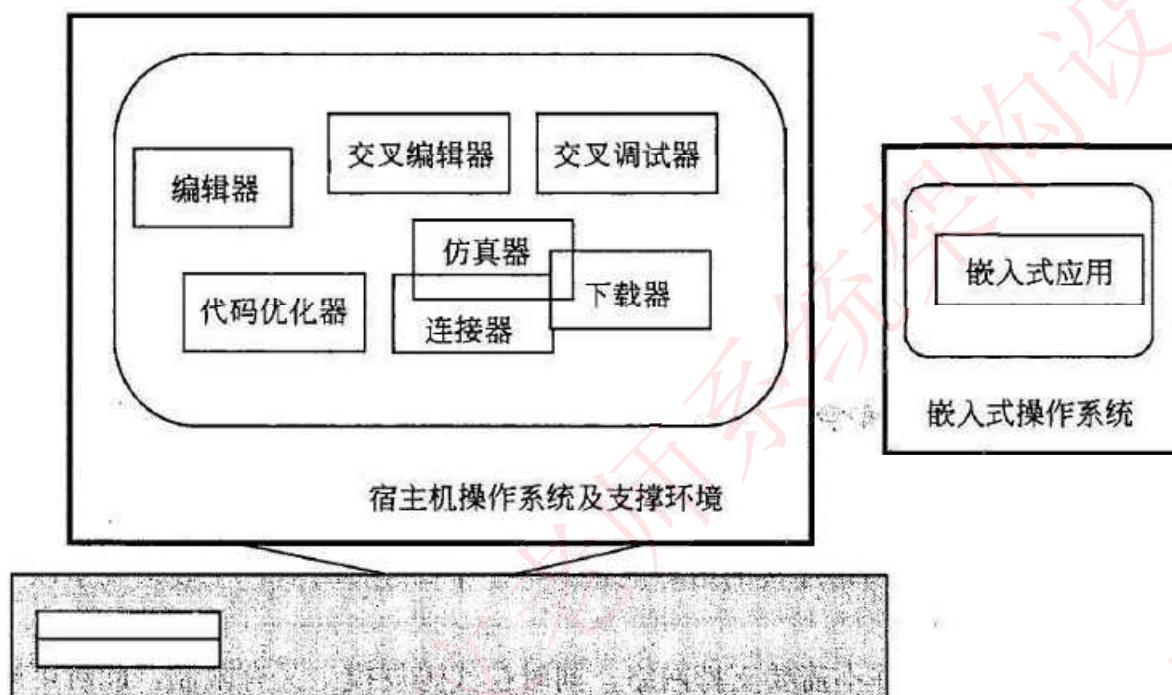
宿主机和目标机

嵌入式软件开发不同于传统软件开发，其所使用的开发环境、工具都有特殊性，在嵌入式软件开发中，一般使用宿主机和目标机的模式进行系统开发，并且借助于开发工具进行目标开发。

宿主机是指普通 PC 机中构建的开发环境，一般需要配置交叉编译器，借助于宿主机的环境，使用交叉编译器进行目标编译，代码生成，同时借助仿真器或者是网络进行目标机的程序调式。

目标机可以是嵌入式系统的实际运行环境，也可以是能够替代实际运行环境的仿真系统。

嵌入式软件开发方式一般是：在宿主机上建立开发环境，完成编码和交叉编译工作，然后在宿主机和目标机之间建立连接，将目标程序下载到目标机中进行交叉调试和运行。如下图：



交叉编译

嵌入式软件开发所采用的编译为交叉编译。所谓交叉编译就是在一个平台上生成可以在另一个平台上执行的代码。编译的最主要的工作就在将程序转化成运行该程序的 CPU 所能识别的机器代码，由于不同的体系结构有不同的指令系统。因此，不同的 CPU 需要有相应的编译器，而交叉编译就如同翻译一样，把相同的程序代码翻译成不同 CPU 的对应可执行二进制文件。

由于一般通用计算机拥有非常丰富的系统资源、使用方便的集成开发环境和调试工具等，而嵌入式系统的系统资源非常紧缺，无法在其上运行相关的编译工具，因此，嵌入式系统的开发需要借助宿主机（通用计算机）来编译出目标机的可执行代码。

交叉调试

嵌入式软件经过编译和链接后即进入调试阶段，调试是软件开发过程中必不可少的一个环节，嵌入式软件开发过程中的交叉调试与通用软件开发过程中的调试方式有很大的差别。

在常见软件开发中，调试器与被调试的程序往往运行在同一台计算机上，调试器是一个单独运行着的进程它通过操作系统提供的调试接口来控制被调试的进程。

而在嵌入式软件开发中，调试时采用的是在宿主机和目标机之间进行的交叉调试，调试器仍然运行在宿主机的通用操作系统之上，但被调试的进程却是运行在基于特定硬件平台的嵌入式操作系统中，

调试器和被调试进程通过串口或者网络进行通信，调试器可以控制、访问被调试进程，读取被调试进程的当前状态，并能够改变被调试进程的运行状态。

4.2 开发工具

嵌入式软件的开发可以分为几个阶段：编码、交叉编译、交叉调试。各个阶段工具如下：

编辑器

用于编写嵌入式源代码程序，从理论上来说，任何一个文本编辑器都可以用来编写源代码。

各种集成开发环境会提供功能强大的编辑器，如 VS 系列、eclipse、keil、CSS 等。

常见的独立编辑器：UE、Source Insight、vim 等。

编译器 gcc

编译阶段的工作是用交叉编译工具处理源代码，生成可执行的目标文件，在嵌入式系统中，由于宿主机和目标机系统不一样，需要使用交叉编译，GNU C/C++ (gcc) 是目前常用的一种交叉编译器，支持非常多的宿主机/目标机组合。

gcc 是一个功能强大的工具集合，包含了预处理器、编译器、汇编器、连接器等组件，会在需要时去调用这些组件来完成编译任务。

调试器 gdb

在开发嵌入式软件时，交叉调试是必不可少的一步。嵌入式软件调试特点：

调试器运行在宿主机上，被调试程序运行在目标机上。

调试器通过某种通信方式与目标机建立联系，如串口、并口、网络、JTAG 等。

在目标机上一般有调试器的某种代理，能配合调试器一起完成对目标机上运行程序的调试，可以是软件或支持调试的硬件。

gdb 是 GNU 开源组织发布的一个强大的程序调试工具。

五 计算机网络

1 网络概述和模型

计算机网络是计算机技术与通信技术相结合的产物，它实现了远程通信、远程信息处理和资源共享。

计算机网络的功能：数据通信、资源共享、管理集中化、实现分布式处理、负载均衡。

网络性能指标：速率、带宽（频带宽度或传送线路速率）、吞吐量、时延、往返时间、利用率。

网络非性能指标：费用、质量、标准化、可靠性、可扩展性、可升级性、易管理性和可维护性。

通信技术

计算机网络是利用通信技术将数据从一个结点传送到另一结点的过程。通信技术是计算机网络的基础。

信道可分为物理信道和逻辑信道。物理信道由传输介质和设备组成，根据传输介质的不同，分为无线信道和有线信道。逻辑信道是指在数据发送端和接收端之间存在的一条虚拟线路，可以是有连接的或无连接的。逻辑信道以物理信道为载体。

发信机进行的信号处理包括信源编码、信道编码、交织、脉冲成形和调制。相反地，收信机进行的信号处理包括解调、采样判决、去交织、信道译码和信源译码。

如果同时传递多路数据就需要复用技术和多址技术。复用技术是指在一条信道上同时传输多路数据的技术，如 TDM 时分复用、FDM 频分复用和 CDM 码分复用等。多址技术是指在一条线上同时传输多个用户数据的技术，在接收端把多个用户的数据分离(TDMA 时分多址、FDMA 频分多址和 CDMA 码分多址)。

作为新一代的移动通信技术，5G 的网络结构、网络能力和应用场景等都与过去有很大不同，其特征体现在以下方面。

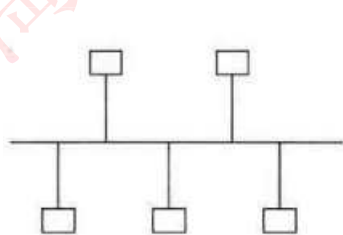
- 1) 基于 OFDM 优化的波形和多址接入
- 2) 实现可扩展的 OFDM 间隔参数配置
- 3) OFDM 加窗提高多路传输效率
- 4) 灵活框架设计
- 5) 大规模 MIMO：最多 256 根天线
- 6) 毫米波：频率大于 24GHz 以上的频段
- 7) 频谱共享
- 8) 先进的信道编码设计

5G 网络的主要特征：服务化架构、网络切片。

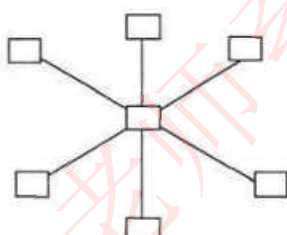
计算机网络按分布范围和拓扑结构划分如下图所示：

网络分类	缩写	分布距离	计算机分布范围	传输速率范围
局域网	LAN	10m 左右	房间	4Mbps~1Gbps
		100m 左右	楼宇	
		1000m 左右	校园	
城域网	MAN	10km	城市	50Kbps~100Mbps
广域网	WAN	100km 以上	国家或全球	9.6Kbps~45Mbps

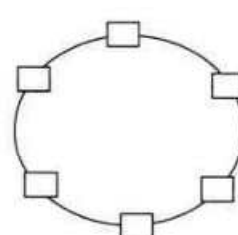
总线型（利用率低、干扰大、价格低）、星型（交换机形成的局域网、中央单元负荷大）、环型（流动方向固定、效率低扩充难）、树型（总线型的扩充、分级结构）、分布式（任意节点连接、管理难成本高）



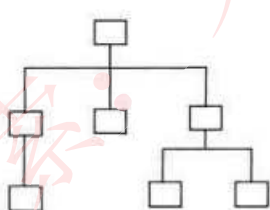
(a) 总线型



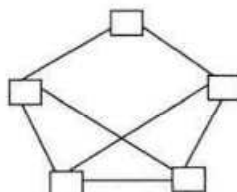
(b) 星型



(c) 环型



(d) 树型



(e) 分布式

一般来说，办公室局域网是星型拓扑结构，中间节点就是交换机，一旦交换机损坏，整个网络都瘫痪了，这就是星型结构。同理，由路由器连接起来的小型网络也是星型结构。

OSI/RM 七层模型

层	功能	单位	协议	设备
1. 物理层	是网络通信的数据传输介质，由连接不同结点的电缆与设备共同构成。主要功能是：利用传输介质为数据链路层提供物理连接，负责处理数据传输并监控数据出错率，以便数据流的透明传输。	比特	EIA/TIA RS-232、RS-449、V. 35、RJ-45、FDDI	中继器、集线器
2. 数据链路层	在物理层提供的服务基础上，在通信的实体间建立数据链路连接，传输以“帧”为单位的数据包，并采用差错控制与流量控制方法，使有差错的物理线路变成无差错的数据链路。	帧	SDLC（同步数据链路控制）、HDLC（高级数据链路控制）、PPP（点对点协议）、STP（生成树协议）、帧中继等、IEEE802、ATM（异步传输）	交换机、网桥
3. 网络层	为数据在结点之间传输创建逻辑链路，通过路由选择算法为分组通过通信子网选择最适当的路径，以及实现拥塞控制、网络互联等功能。	IP分组	IP（网络互连）、IPX（互联网数据包交换协议）、ICMP（网际控制报文协议）、IGMP（网络组管理协议）、ARP（地址转换协议）、RARP	路由器
4. 传输层	向用户提供可靠的端到端（End-to-End）服务，处理数据包错误、数据包次序，以及其他一些关键传输问题。传输层向高层屏蔽了下层数据通信的细节	报文段	TCP（传输控制协议）、UDP（用户数据报协议）、SPX（序列分组交换协议）	网关
5. 会话层	负责维护两个结点之间的传输连接，以便确保点到点传输不中断，以及管理数据交换等功能。		RPC（远程过程调用）、SQL（结构化查询语言）、NFS（网络文件系统）	网关
6. 表示层	用于处理在两个通信系统中交换信息的表示方式，主要包括数据格式变换、数据加密与解密、数据压缩与恢复等功能。		JPEG、ASCII、GIF、MPEG、DES	网关
7. 应用层	为应用软件提供了很多服务，例如文件服务器、数据库服务、电子邮件与其他网络软件服务。	数据	Telnet、FTP、HTTP、SMTP、POP3、DNS、DHCP等	网关

以太网是一种计算机局域网组网技术。

以太网规范 IEEE 802.3 是重要的局域网协议，包括：

IEEE 802.3	标准以太网	10Mb/s	传输介质为细同轴电缆
IEEE 802.3u	快速以太网	100Mb/s	双绞线
IEEE 802.3z	千兆以太网	1000Mb/s	光纤或双绞线
IEEE 802.3ae	万兆以太网	10Gb/s	光纤

DMAC	SMAC	Length/Type	DATA/PAD	FCS
------	------	-------------	----------	-----

图 2-20 以太帧结构

上图依次为：目的 MAC 地址、源 MAC 地址、长度/类型、数据填充、校验。

最小帧长：64 字节。

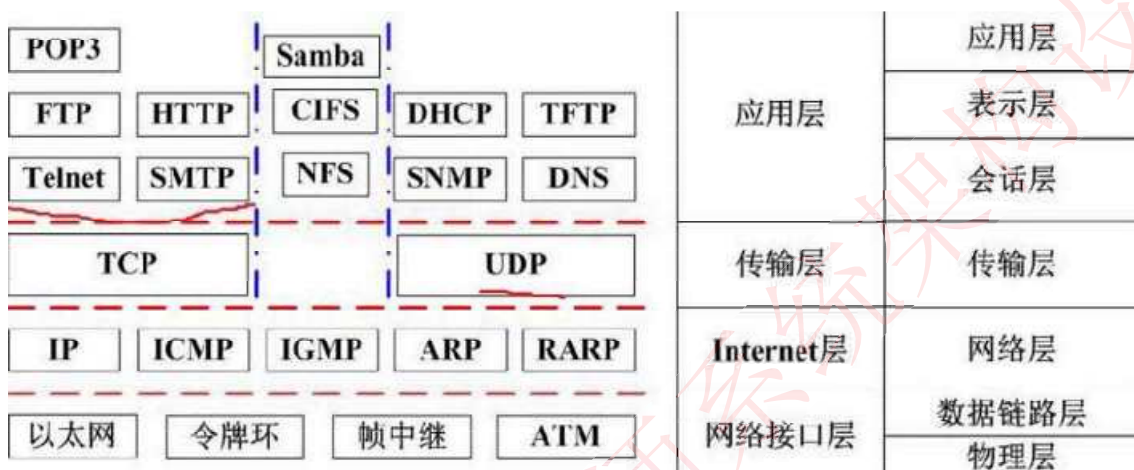
无线局域网 WLAN 技术标准：IEEE 802.11

在 WLAN 中，通常使用的拓扑结构主要有 3 种形式：点对点型、HUB 型和全分布型。

广域网相关技术：同步光网络（SONET，利用光纤进行数字化信息通信）、数字数据网（DDN，利用数字信道提供半永久性连接电路以传输数据）、帧中继（FR，数据包交换技术）、异步传输技术（ATM，以信元为基础的面向连接的一种分组交换和复用技术）。

TCP/IP 协议

网络协议三要素：语法、语义、时序。



网络层协议：

IP：网络层最重要的核心协议，在源地址和目的地址之间传送数据报，无连接、不可靠。

ICMP：因特网控制报文协议，用于在 IP 主机、路由器之间传递控制消息。控制消息是指网络不通、主机是否可达、路由是否可用等网络本身的消息。

ARP 和 RARP：地址解析协议，ARP 是将 IP 地址转换为物理地址，RARP 是将物理地址转换为 IP 地址。

IGMP：网络组管理协议，允许因特网中的计算机参加多播，是计算机用做向相邻多目路由器报告多目组成员的协议，支持组播。

传输层协议：

TCP：整个 TCP/IP 协议族中最重要的协议之一，在 IP 协议提供的不可靠数据数据基础上，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。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、无连接的协议，有助于提高传输速率，一般用于传输数据量大，对可靠性要求不高，但要求速度快的场合。

应用层协议：

基于 TCP 的 FTP、HTTP 等都是可靠传输。基于 UDP 的 DHCP、DNS 等都是不可靠传输。

FTP：可靠的文件传输协议，用于因特网上的控制文件的双向传输。

HTTP：超文本传输协议，用于从 WWW 服务器传输超文本到本地浏览器的传输协议。使用 SSL 加密后的安全网页协议为 HTTPS。

SMTP 和 POP3：简单邮件传输协议，是一组用于由源地址到目的地址传送邮件的规则，邮件报文采用 ASCII 格式表示。

Telnet：远程连接协议，是因特网远程登录服务的标准协议和主要方式。

TFTP：不可靠的、开销不大的小文件传输协议。

SNMP：简单网络管理协议，由一组网络管理的标准协议，包含一个应用层协议、数据库模型和一组资源对象。该协议能够支持网络管理系统，泳衣监测连接到网络上的设备是否有任何引起管理师行关注的情况。

DHCP：动态主机配置协议，基于 UDP，基于 C/S 模型，为主机动态分配 IP 地址，有三种方式：固定分配、动态分配、自动分配。

DNS：域名解析协议，通过域名解析出 IP 地址。

协议端口号对照表

端 口	服 务	端 口	服 务
20	文件传输协议（数据）	80	超文本传输协议（HTTP）
21	文件传输协议（控制）	110	POP3 服务器（邮箱接收服务器）
23	Telnet 终端仿真协议	69	简单文件传输协议（TFTP）
67	DHCP（服务端）	68	DHCP（客户端）
25	SMTP简单邮件发送协议	161	SNMP（轮询）
53	域名服务器（DNS）	162	SNMP（陷阱）

交换技术

数据在网络中转发通常离不开交换机。人们日常使用的计算机通常就是通过交换机接入网络的。交换机功能包括：

集线功能。提供大量可供线缆连接的端口达到部署星状拓扑网络的目的。

中继功能。在转发帧时重新产生不失真的电信号。

桥接功能。在内置的端口上使用相同的转发和过滤逻辑。

隔离冲突域功能。将部署好的局域网分为多个冲突域，而每个冲突域都有自己独立的带宽，以提高交换机整体宽带利用效率。

交换机需要实现的功能如下所述。

（1）转发路径学习。根据收到数据帧中的源 MAC 地址建立该地址同交换机端口的映射，写入 MAC 地址表中。

（2）数据转发。如果交换机根据数据帧中的目的 MAC 地址在建立好的 MAC 地址表中查询到了，就向对应端口进行转发。

（3）数据泛洪。如果数据帧中的目的 MAC 地址不在 MAC 地址表中，则向所有端口转发，也就是泛洪。广播帧和组播帧向所有端口（不包括源端口）进行转发。

（4）链路地址更新。MAC 地址表会每隔一定时间（如 300s）更新一次。

路由技术

路由功能由路由器来提供，具体包括：

（1）异种网络互连，比如具有异种子网协议的网络互连：

（2）子网协议转换，不同子网间包括局域网和广域网之间的协议转换：

- (3) 数据路由，即将数据从一个网络依据路由规则转发到另一个网络；
- (4) 速率适配，利用缓存和流控协议进行适配；
- (5) 隔离网络，防止广播风暴，实现防火墙；
- (6) 报文分片和重组，超过接口的 MTU 报文被分片，到达目的地之后的报文被重组；
- (7) 备份、流量控制，如主备线路的切换和复杂流量控制等。

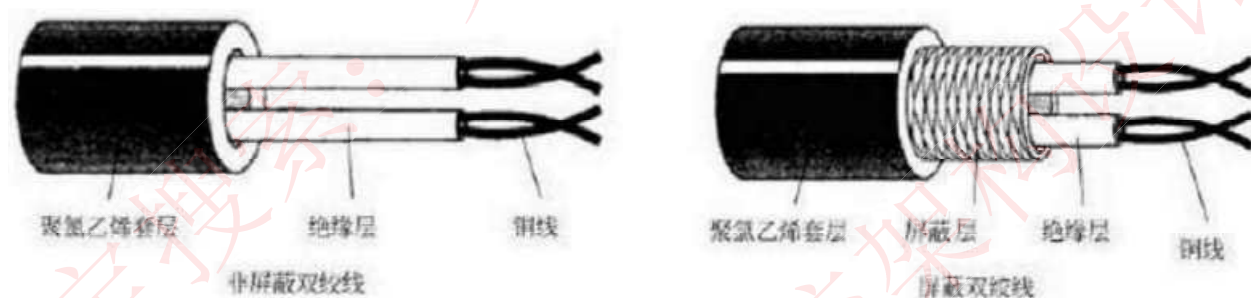
路由器工作在 OSI 七层协议中的第 3 层，即网络层。其主要任务是接收来源于一个网络接口的数据包，通常根据此数据包的目地址决定待转发的下一个地址（即下一跳地址）。路由器中维持着数据转发所需的路由表，所有数据包的发送或转发都通过查找路由表来实现。这个路由表可以静态配置，也可以通过动态路由协议自动生成。

一般来说，路由协议可分为内部网关协议（IGP）和外部网关协议（EGP）两类。

网络建设工程可分为网络规划、网络设计和网络实施三个环节。

2 传输介质

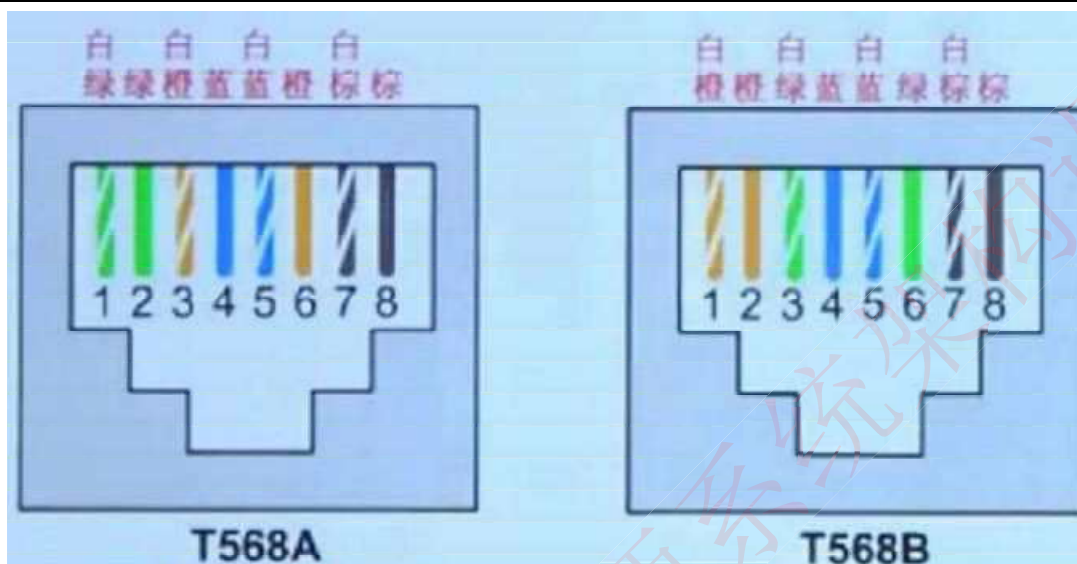
双绞线：将多根铜线按规则缠绕在一起，能够减少干扰；分为无屏蔽双绞线 UTP 和屏蔽双绞线 STP，都是由一对铜线簇组成。也即我们常说的网线；双绞线的传输距离在 100m 以内。



无屏蔽双绞线 UTP：价格低，安装简单，但可靠性相对较低，分为 CAT3（3 类 UTP，速率为 10Mbps）、CAT4（4 类 UTP，与 3 类差不多，无应用）、CAT5（5 类 UTP，速率为 100Mbps，用于快速以太网）、CAT5E（超 5 类 UTP，速率为 1000Mbps）、CAT6（6 类 UTP，用来替代 CAT5E，速率也是 1000Mbps）。

屏蔽双绞线 STP：比之 UTP 增加了一层屏蔽层，可以有效的提高可靠性，但对应的价格高，安装麻烦，一般用于对传输可靠性要求很高的场合。

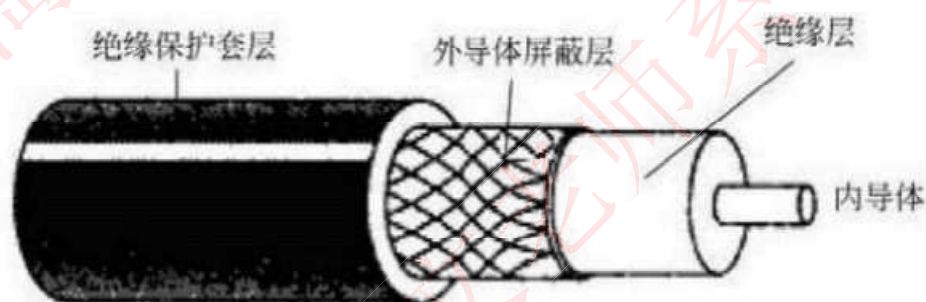
网线有如下两种安装标准：都是八根不同颜色的网线，按照不同的顺序排序，插入水晶头中，区分在第 1236 四根网线的位置不同。



网线又分为直通线和交叉线，直通线就是网线两头采用同一种标准，交叉线就是网线两头采用不同的标准，交叉线和直通线的使用范围如下：

线缆连接设备情况	所采用的线缆种类
计算机-计算机	交叉线
计算机-交换机	直通线
计算机-路由器	交叉线
交换机-交换机	交叉线
交换机-路由器	直通线
路由器-路由器	交叉线

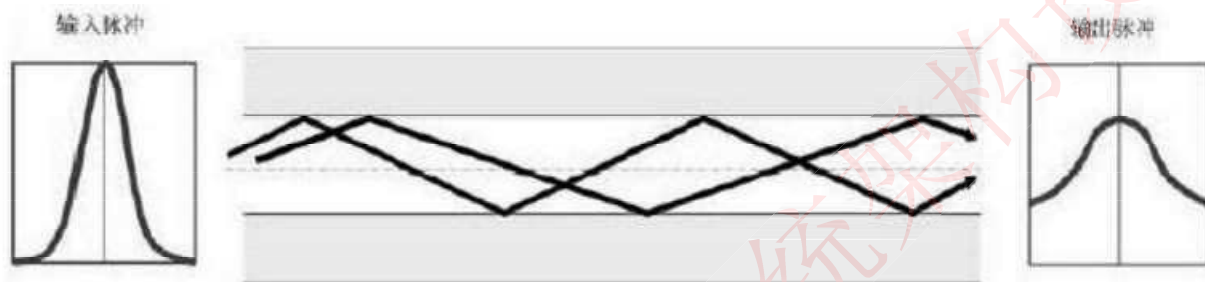
同轴电缆



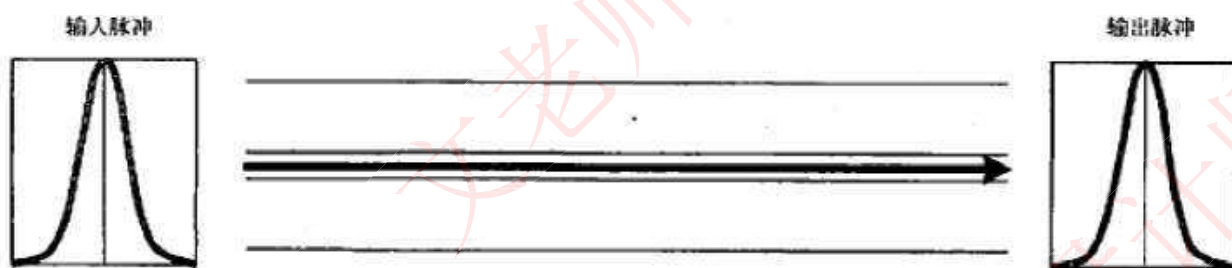
分为粗、细同轴电缆，传输距离长，信号稳定，常用于电视、监视系统、视频信号等。传输距离有几百米，随着数字网络的发展，已经逐步淘汰。

光纤：由纤芯和包层组成，传输的光信号在纤芯中传输，然而从 PC 端出来的信号都是电信号，要经过光纤传输的话，就必须将电信号转换为光信号。

多模光纤 MMF：纤芯半径较大，因此可以同时传输多种不同的信号，光信号在光纤中以全反射的形式传输，采用发光二极管 LED 为光源，成本低，但是传输的效率和可靠性都较低，适合于短距离传输，其传输距离与传输速率相关，速率为 100Mbps 时为 2KM，速率为 1000Mbps 时为 550m。



单模光纤 SMF：纤芯半径很小，一般只能传输一种信号，采用激光二极管 LD 作为光源，并且只支持激光信号的传播，同样是以全反射形式传播，只不过反射角很大，看起来像一条直线，成本高，但是传输距离远，可靠性高。传输距离可达 5KM。



无线信道

分为无线电波和红外光波

无线电波	长波		红外光波	近红外线
	中波			
	短波			中红外线
	超短波			
	微波	地面微波		远红外线
		卫星微波		

3 通信方式和交换方式

通信方向

数据通信是指发送方发送数据到接收方，这个传输过程可以分类如下：

单工：只能由设备 A 发给设备 B，即数据流只能单向流动。

半双工：设备 A 和设备 B 可以互相通信，但是同一时刻数据流只能单向流动。

全双工：设备 A 和设备 B 在任意时刻都能互相通信。



同步方式

异步传输：发送方每发送一个字符，需要约定一个起始位和停止位插入到字符的起始和结尾处，这样当接收方接收到该字符时能够识别，但是这样会造成资源浪费，传输效率降低。

同步传输：以数据块为单位进行传输，当发送方要发送数据时，先发送一个同步帧，接收方收到后做好接收准备，开始接收数据块，结束后又会有结束帧确认，这样一次传输一个数据块，效率高。

串行传输：只有一根数据线，数据只能 1bit 挨个排队传送，适合低速设备、远距离的传送，一般用于广域网中。

并行传输：有多根数据线，可以同时传输多个 bit 数据，适合高速设备的传送，常用于计算机内部各硬件模块之间。

交换方式

电路交换（线路交换）：通信一方进行呼叫，另一方接收后，在二者之间会建立一个专用电路，特点为面向连接、实时性高、链路利用率低、但没有差错检测，不可靠，一般用于语音视频通信。

报文交换：以报文为单位，存储转发模式，接收到数据后先存储，进行差错校验，没有错误则转发，有错误则丢弃，因此会有延时，但可靠性高，是面向无连接的。

分组交换：以分组为单位，也是存储转发模式，因为分组的长度比报文小，所以时延小于报文交换，又可分为三种方式：

——数据报：是现在主流的交换方式，各个分组携带地址信息，自由的选择不同的路由路径传送到接收方，接收方接收到分组后再根据地址信息重新组装成原数据，是面向无连接的，但是不可靠的。

——虚电路：发送方发送一个分组，接收方收到后二者之间就建立了一个虚拟的通信线路，二者之间的分组数据交互都通过这条线路传送，在空闲的时候这条线路也可以传输其他数据，是面向连接的，可靠的。

——信元交换：异步传输模式 ATM 采用的交换方式，本质是按照虚电路方式进行转发，只不过信元是固定长度的分组，共 53B，其中 5B 为头部，48B 为数据域，也是面向连接的，可靠的。

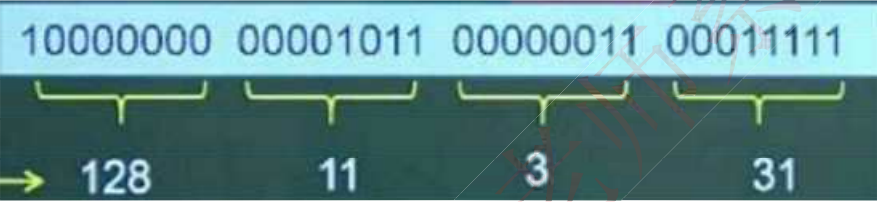
4 IP 地址

4.1 分类地址格式

主机位：全 0 是本地网络地址，全 1 是广播地址，其他是主机地址。

IP 地址的表示

机器中存放的 IP 地址是 32 位的二进制代码，每隔 8 位插入一个空格，可提高可读性，为了便于理解和设置，一般会采用点分十进制方法表示：将 32 位二进制代码每 8 位二进制转换成十进制，就变成了 4 个十进制数，而后在每个十进制数间隔中插入.，如下所示，最终为 128.11.3.31：



因为每个十进制数都是由 8 个二进制数转换而来，因此每个十进制数的取值范围为 0-255（掌握二进制转十进制的快速计算方法，牢记 2 的幂指数值，实现快速转换）。

分类 IP 地址

IP 地址分四段，每段八位，共 32 位二进制数组成。

在逻辑上，这 32 位 IP 地址分为网络号和主机号，依据网络号位数的不同，可以将 IP 地址分为以下几类：

类别	点分十进制	二进制	
A类	0.0.0.0	最低	00000000 00000000 00000000 00000000
	127.255.255.255	最高	01111111 11111111 11111111 11111111
B类	128.0.0.0	最低	10000000 00000000 00000000 00000000
	191.255.255.255	最高	10111111 11111111 11111111 11111111
C类	192.0.0.0	最低	11000000 00000000 00000000 00000000
	223.255.255.255	最高	11011111 11111111 11111111 11111111
D类 组播	224.0.0.0	最低	11100000 00000000 00000000 00000000
	239.255.255.255	最高	11101111 11111111 11111111 11111111
E类 保留	240.0.0.0	最低	11110000 00000000 00000000 00000000
	255.255.255.255	最高	11111111 11111111 11111111 11111111

特殊 IP 地址

公有地址：由 Inter NIC（因特网信息中心）负责，这些 IP 地址分配给注册并向 Inter NIC 提出申请的组织机构，通过它直接访问因特网。是全网唯一的 IP 地址。

私有地址：属于非注册地址，专门为组织机构内部使用，在不同企业内部可以重复使用，不能直接访问因特网，下表所示为私有地址范围：

类别	IP地址范围	网络号	网络数
A	10.0.0.0~10.255.255.255	10	1
B	172.16.0.0~172.31.255.255	172.16~172.31	16
C	192.168.0.0~192.168.255.255	192.168.0~192.168.255	256

其他特殊地址如下表所示：

网络号	主机号	源地址使用	目的地址使用	代表的意义
0	0	可以	不可	在本网络上的本主机
全1	全1	不可	可以	在本网络上进行广播
Net-ID	全1	不可	可以	对net-ID上的所有主机进行广播
127	非全0或全1的数	可以	可以	用作本地软件环回测试之用
169.254	非全0或全1的数	可以	可以	Windows主机DHCP服务器故障分配

169.254 是自动专用 IP 地址，当 DHCP 故障时为主机分配的临时 IP 地址。

4.2 子网划分

可变长子网掩码 VLSM

按上述划分的 A B C 三类，并不实用，因为主机数之间相差的太大了，不利于分配，导致 IP 地址空间的利用率很低，并且两级 IP 地址不够灵活。

因此，我们一般采用子网划分的方法来划分网络，将主机号拿出几位作为子网号，就可以划分出多个子网，此时 IP 地址组成为：网络号+子网号+主机号。（本质还是保留了分类编址结构）

网络号和子网号都为 1，主机号都为 0，这样的地址为子网掩码。

要判断两个 IP 地址是否在同一网段，只需要确定其网络号+子网号是否相同，可以将子网掩码和源 IP 地址进行与运算，得出网络地址，判断网络地址是否相同即可。网络号不同的主机属于不同的网段，不能直接通信，必须通过网络路由才能互相通信。

要注意的是：子网号可以为全 0 和全 1，主机号不能为全 0 或全 1，因此，在计算时，主机数需要-2，而子网数不用。

划分子网之后，就变成了三级 IP 地址，如果此时还需要进一步划分子网，还可以继续向主机号借位，将主机号拿出几位再作为子网号，这就是可变长度子网划分，根据需要灵活选择，但要注意的是都是向主机号借位。

4.3 无分类编址

上述的子网划分方式，本质是在分类编址的基础上，向主机位借位，还是保留了分类编址的结构，而无分类编址 CIDR 方式，则是完全忽略分类编址，可以按照需要自定义网络号和主机号位数（只需要保证整体 IP 地址长度为 32bit 即可）。

无分类编址格式为：**IP 地址/网络号**，示例：**128.168.0.11/20** 表示的 IP 地址为 **128.168.0.11**，其网络号占 20 位，因此主机号占 $32-20=12$ 位，无分类编址也可以划分子网，原理相同。要掌握的是，/后的数字是网络号，32 减去此数字就是主机号。

CIDR 路由汇聚

汇聚规则：选择网络地址相同的位进行汇聚，不同的位划分至主机位，从而实现将多个网段汇聚成一个新的超网网段。

可以将汇聚超网的过程理解成划分子网的逆过程，划分子网是不断向主机位借位，导致子网越来越多，每个子网的主机号越来越少；而汇聚超网是将不同网段的相同的网络号作为汇聚后的网络号，不同的网络号给主机号，相当于缩小了子网的数量，增加了网络内主机的数量。

最佳路由匹配原则

使用路由汇聚时，路由表中的每个项目由网络前缀和下一跳地址组成。因此查找路由表时可能会得到不止一个匹配结果。此时应当从匹配结果中选择具有最长网络前缀的路由。

网络前缀越长，其地址块就越小，因而路由就越具体，最长前缀匹配又称为最长匹配或最佳匹配。

5 IPv6

主要是为了解决 IPv4 地址数不够用的情况而提出的设计方案，IPv6 具有以下特性：

IPv6 地址长度为 128 位，地址空间增大了 2^{96} 倍；

灵活的 IP 报文头部格式，使用一系列固定格式的扩展头部取代了 IPv4 中可变长度的选项字段。IPv6 中选项部分的出现方式也有所变化，使路由器可以简单撸过选项而不做任何处理，加快了报文处理速度；

IPv6 简化了报文头部格式，加快报文转发，提高了吞吐量；

提高安全性，身份认证和隐私权是 IPv6 的关键特性；

支持更多的服务类型；

允许协议继续演变，增加新的功能，使之适应未来技术的发展。

IPv4 和 IPv6 的过渡期间，主要采用三种基本技术：

（1）双协议栈：主机同时运行 IPv4 和 IPv6 两套协议栈，同时支持两套协议，一般来说 IPv4 和 IPv6 地址之间存在某种转换关系，如 IPv6 的低 32 位可以直接转换为 IPv4 地址，实现互相通信。

（2）隧道技术：这种机制用来在 IPv4 网络之上建立一条能够传输 IPv6 数据报的隧道，例如可以将 IPv6 数据报当做 IPv4 数据报的数据部分加以封装，只需要加一个 IPv4 的首部，就能在 IPv4 网络中传输 IPv6 报文。

(3) 翻译技术：利用一台专门的翻译设备（如转换网关），在纯 IPv4 和纯 IPv6 网络之间转换 IP 报头的地址，同时根据协议不同对分组做相应的语义翻译，从而使纯 IPv4 和纯 IPv6 站点之间能够透明通信。

6 网络规划和设计

6.1 层次化局域网模型

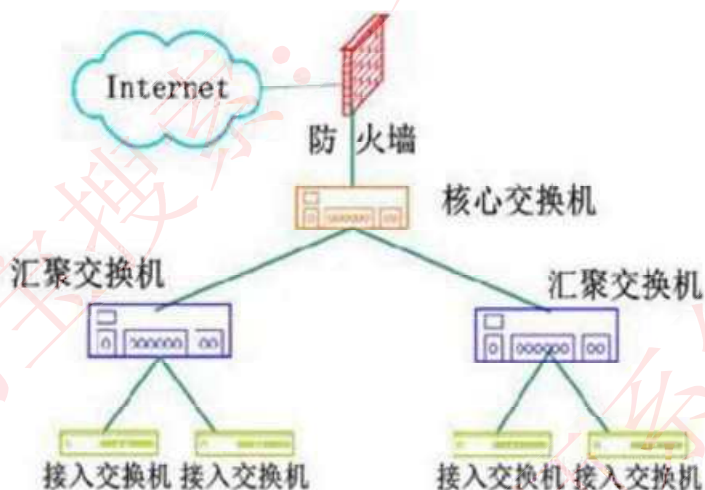
三层模型将网络划分为核心层、汇聚层和接入层，每一层都有着特定的作用。

核心层提供不同区域之间的最佳路由和高速数据传送；

汇聚层将网络业务连接到接入层，并且实施与安全、流量、负载和路由相关的策略；

接入层为用户提供了在本地网段访问应用系统的能力，还要解决相邻用户之间的互访需要，接入层要负责一些用户信息（例如用户 IP 地址、MAC 地址和访问日志等）的收集工作和用户管理功能（包括认证和计费等）。

下面是一个三层示例，一般设计思路都是从下往上设计，即先设计接入层、汇聚层，再考虑核心层数据交换的速度以满足下面两层的要求。



6.2 建筑物综合布线系统 PDS

工作区子系统：实现工作区终端设备到水平子系统的信息插座之间的互联。

水平布线子系统：实现信息插座和管理子系统之间的连接。

设备间子系统：实现中央主配线架与各种不同设备之间的连接。

垂直干线子系统：实现各楼层设备间子系统之间的互连。

管理子系统：为连接其他子系统提供连接手段。

建筑群子系统：各个建筑物通信系统之间的互联。

7 网络管理命令

ipconfig 命令：用来查看网络配置信息，如 IP 地址、子网掩码、网关 IP、DNS、DHCP 等。Linux 下是 **ifconfig** 命令。

ping 命令：利用 ICMP 报文来测试网络的连通性、是否丢包、名称解析等，如 **ping 127.0.0.1** 检查 TCP/IP 协议，**ping** 本机 IP 检查网卡工作，**ping** 网关等。

arp 命令：用于显示和修改 ARP 缓存中的表项。

netstat 命令：用来显示网络活动状态，如 TCP/UDP 的 IP、端口号、统计信息等。

tracert 命令：利用 ICMP 报文，来探测到达目标的路径，参数通常为 **-d**。

pathing 命令：把 **ping** 和 **tracert** 结合起来，探测路径、延时、丢包率等。

nbtstat 命令：用来显示 NetBIOS 的名称缓存。

route 命令：显示和修改本地 IP 路由器表，参数 **add** 添加、**delete** 删除、**change** 修改、**print** 显示路由表（同 **netstat -r**）。

netsh 命令：命令行脚本程序，可修改计算机的网络配置。

net 命令：管理网络服务。

nslookup 命令：命令用于显示 DNS 查询信息，诊断和排除 DNS 故障。

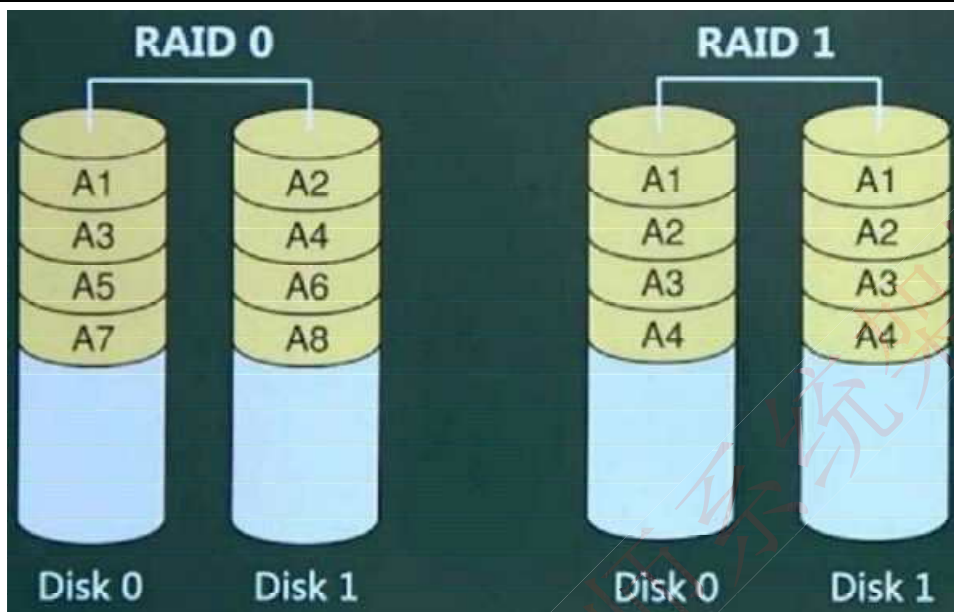
8 网络存储技术

8.1 廉价磁盘冗余阵列

RAID 即磁盘冗余阵列技术，将数据分散存储在不同磁盘中，可并行读取，可冗余存储，提高磁盘访问速度，保障数据安全性。

RAID0 将数据分散的存储在不同磁盘中，磁盘利用率 **100%**，访问速度最快，但是没有提供冗余和错误修复技术；

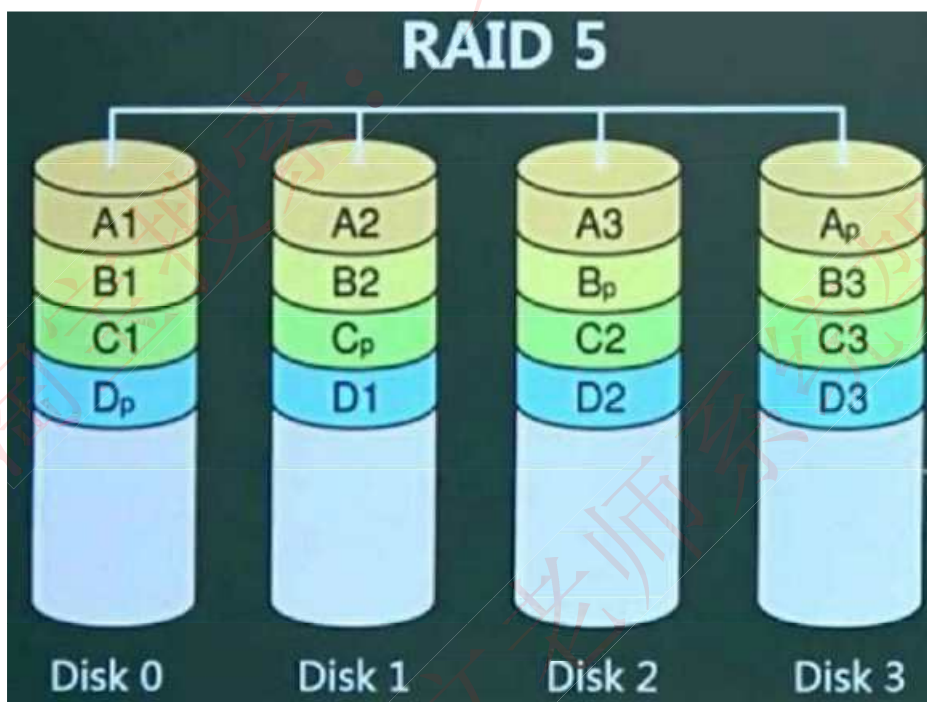
RAID1 在成对的独立磁盘上产生互为备份的数据，增加存储可靠性，可以纠错，但磁盘利用率只有 **50%**；



RAID2 将数据条块化的分布于不同硬盘上，并使用海明码校验；

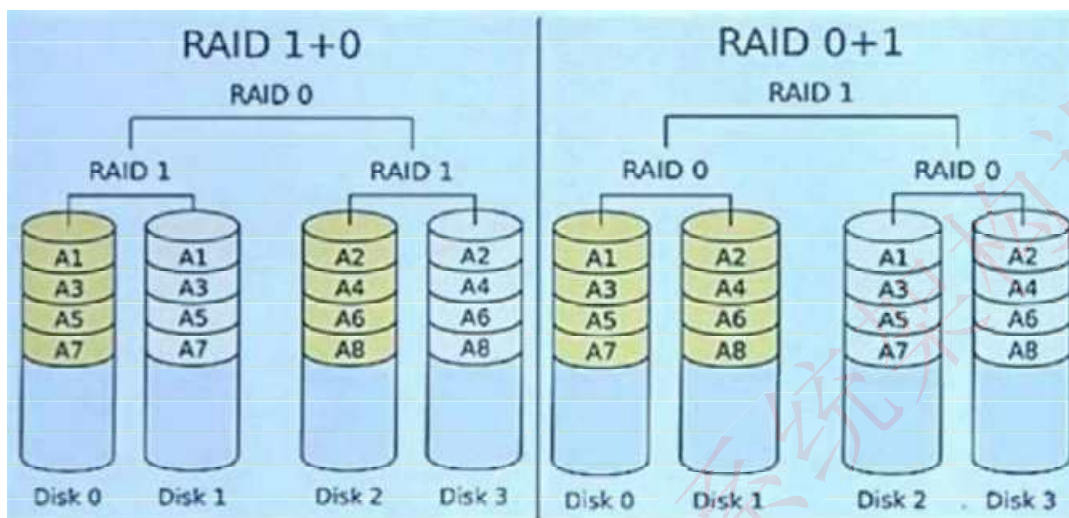
RAID3 使用奇偶校验，并用单块磁盘存储奇偶校验信息（可靠性低于 RAID5）；

RAID5 在所有磁盘上交叉的存储数据及奇偶校验信息（所有校验信息存储总量为一个磁盘容量，但分布式存储在不同的磁盘上），读/写指针可同时操作；



RAID0+1（是两个 RAID0，若一个磁盘损坏，则当前 RAID0 无法工作，即有一半的磁盘无法工作）；

RAID1+0（是两个 RAID1，不允许同一组中的两个磁盘同时损坏）与 RAID1 原理类似，磁盘利用率都只有 50%，但安全性更高。



8.2 网络存储

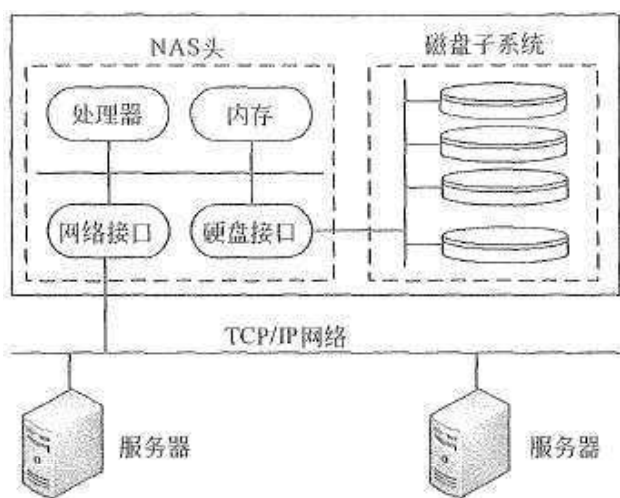
具体有下列四种存储技术：

1.直接附加存储(DAS)：是指将存储设备通过 SCSI 接口直接连接到一台服务器上使用，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。

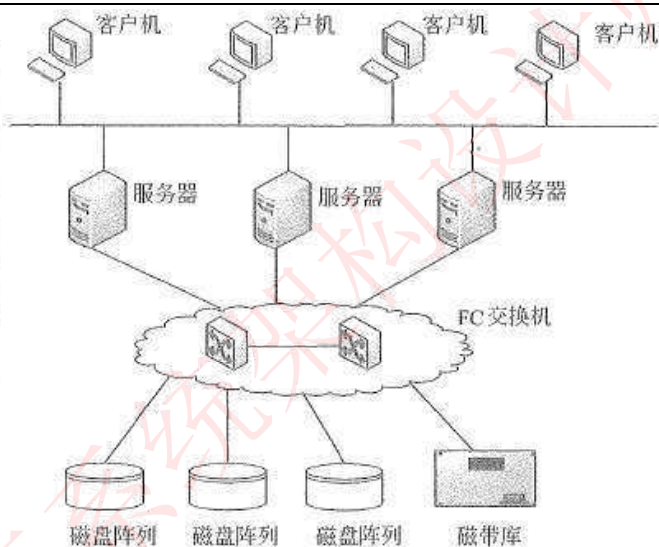
存在问题：在传递距离、连接数量、传输速率等方面都受到限制。容量难以扩展升级；数据处理和传输能力降低；服务器异常会波及存储器。

2.网络附加存储(NAS)：通过网络接口与网络直接相连，由用户通过网络访问，有独立的存储系统。如下图所示。NAS 存储设备类似于一个专用的文件服务器，去掉了通用服务器大多数计算功能，而仅提供文件系统功能。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 的性能特点是进行小文件级的共享存取；支持即插即用；可以很经济的解决存储容量不足的问题，但难以获得满意的性能。



NAS 存储系统的结构



SAN 存储系统的结构

3. 存储区域网(SAN): 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了出来, 成为独立的存储区域网络 SAN 的系统结构。根据数据传输过程采用的协议, 其技术划分为 FC SAN (光纤通道)、IP SAN (IP 网络) 和 IB SAN (无线带宽) 技术。

9 其他考点汇总

网络地址翻译 NAT

公司内有很多电脑, 在公司局域网内可以互联通信, 但是要访问外部因特网时, 只提供固定的少量 IP 地址能够访问因特网, 将公司所有电脑这个大的地址集合映射到能够访问因特网的少量 IP 地址集合的过程就称为 NAT。

很明显, 使用了 NAT 后, 一个公司只有少量固定 IP 地址可以上网, 大大减少了 IP 地址的使用量。

默认网关

一台主机可以有多个网关。默认网关的意思是一台主机如果找不到可用的网关, 就把数据包发给默认指定的网关, 由这个网关来处理数据包。现在主机使用的网关, 一般指的是默认网关。

默认网关的 IP 地址必须与本机 IP 地址在同一个网段内, 即同网络号。

虚拟局域网 VLAN

是一组逻辑上的设备和用户, 这些设备和用户并不受物理位置的限制, 可以根据功能、部门及应用等因素将它们组织起来, 相互之间的通信就好像它们在同一个网段中一样。

VLAN 工作在 OSI 参考模型的第 2 层和第 3 层, 一个 VLAN 就是一个广播域, VLAN 之间的通信是通过第 3 层的路由器来完成的。

与传统的局域网技术相比较, VLAN 技术更加灵活, 它具有以下优点: 网络设备的移动、添加和修改的管理开销减少; 可以控制广播活动; 可提高网络的安全性。

虚拟专用网 VPN

是在公用网络上建立专用网络的技术。其之所以称为虚拟网，主要是因为整个 VPN 网络的任意两个节点之间的连接并没有传统专网所需的端到端的物理链路，而是架构在公用网络服务商所提供的网络平台，如 Internet、ATM(异步传输模式)、Frame Relay（帧中继）等之上的逻辑网络，用户数据在逻辑链路中传输。

PPP

安全认证介绍：PPP 的 NCP 可以承载多种协议的三层数据包。PPP 使用 LCP 控制多种链路的参数（建立、认证、压缩、回拨）。

PPP 的认证类型：pap 认证是通过二次握手建立认证（明文不加密），chap 挑战握手认证协议，通过三次握手建立认证（密文采用 MD5 加密）。PPP 的双向验证，采用的是 chap 的主验证风格。PPP 的加固验证，采用的是两种（pap,chap）验证同时使用

冲突域和广播域

路由器可以阻断广播域和冲突域，交换机只能阻断冲突域，因此一个路由器下可以划分多个广播域和多个冲突域；一个交换机下整体是一个广播域，但可以划分多个冲突域；而物理层设备集线器下整体作为一个冲突域和一个广播域。

六 其他计算机系统基础知识

1 计算机语言

计算机语言是指用于人与计算机之间交流的一种语言，是人与计算机之间传递信息的媒介。计算机语言主要由一套指令组成，而这种指令一般包括表达式、流程控制和集合三大部分内容。

表达式又包含变量、常量、字面量和运算符。

流程控制有分支、循环、函数和异常。

集合包括字符串、数组、散列表等数据结构。

计算机语言的分类

1.机器语言：最早使用，第一代计算机语言，计算机自身语言，二进制代码串。

机器语言指令必须包括：操作码、操作数的地址、操作结果的存储地址、下条指令的地址。

常见的指令格式：三地址指令（2 个操作数+1 个结果地址）、二地址指令（1 操作数+1 操作数和结果地址）、单地址指令（1 操作数+固定寄存器存放）、零地址指令（堆栈顶指示操作数和结果地址）、可变地址数指令（0-6 个）。

2.汇编语言：用一些简洁的英文字母、符号串来替代一个特定指令的二进制串。第二代，仍然是面向机器的语言。通过汇编程序将汇编语言翻译成机器语言。

汇编语言 3 中语句：指令（汇编后直接产生机器代码）、伪指令（指示汇编源程序时完成的操作，汇编后不产生机器代码）、宏指令（多次重复使用的程序段，宏的引用）。

指令语句和伪指令语句格式：名字（标号，第一个字节单元地址）、操作符、操作数、注释。

3.高级语言：一类语言统称，符合人类习惯，如 C、C++、Java、VB、C#、Python 等。

4.建模语言：主导地位是面向对象的建模技术，主要使用 UML 作为建模语言。

这里教材提到了 UML 的 3 个要素、事物、关系、视图。但是没有提到 UML 的图、面向对象基础、设计模式，不是体系化的内容。因此将面向对象独立出来，这里暂不阐述。

5.形式化语言：形式化方法是把概念、判断、推理转化成特定的形式符号后，对形式符号表达系统进行研究的方法，是用具有精确语义的形式语言书写的程序功能描述，它是设计和编制程序的出发点，也是验证程序是否正确的依据。

形式化方法的开发过程：可行性分析、需求分析、体系结构设计、详细设计、编码、测试发布。

2 多媒体

媒体是承载信息的载体，即信息的表现形式（或者传播形式），如文字、声音、图像、动画和视频等。按照 ITU-T 建议的定义，媒体可分为感觉媒体、表示媒体、显示媒体、存储媒体和传输媒体。

（1）感觉媒体，指的是用户接触信息的感受形式，如视觉、听觉和触觉等。

（2）表示媒体，指的是信息的表示形式，如图像、声音、视频等。

（3）表现媒体，也称为显示媒体，指表现和获取信息的物理设备，如键盘、鼠标、扫描仪、话筒和摄像机等为输入媒体；显示器、打印机和音箱等为输出媒体。

（4）存储媒体，指用于存储表示媒体的物理介质，如硬盘、软盘、磁盘、光盘、ROM 及 RAM 等。

（5）传输媒体，指传输表示媒体的物理介质，如电缆、光缆和电磁波等。

多媒体有 4 个重要的特征。

（1）多媒化。多媒化是指媒体的多样化。它提供了多媒化信息空间下的交互能力和获得多媒化信息空间的方法，如输入、输出、传输、存储和处理的手段与方法等。

（2）集成性。集成性不仅指多媒体设备集成，而且指多媒体信息集成或表现集成。

（3）交互性。交互性是人们获取和使用信息时变被动为主动的最重要的标志。交互性可向用户提供更有效地控制和使用信息的手段，可增加人们对信息的注意和理解。

（4）实时性。实时性是指多媒体技术中涉及的一些媒体。例如，音频和视频信息具有很强的时间特性，会随着时间的变化而变化。

多媒体系统的基本组成



多媒体系统的关键技术

1.视音频技术：视频技术包括视频数字化和视频编码技术两个方面。音频技术包括音频数字化、语音处理、语音合成及语音识别 4 个方面。

2.通信技术是多媒体系统中的一项关键技术，是指将信息从一个地点传送到另一个地点所采取的方法和措施。通信技术通常包括了数据传输信道技术和数据传输技术。

3.数据压缩技术。数据压缩算法分为下面三类：

(1) 即时压缩和非即时压缩。即时 / 非即时压缩的区别在于信息在传输过程中被压缩还是信息压缩后再传输。即时压缩一般应用在影像、声音数据的传送中。即时压缩常用到专门的硬件设备，如压缩卡等。

(2) 数据压缩和文件压缩。数据压缩是专指一些具有时间性的数据，这些数据常常是即时采集、即时处理或传输的。而文件压缩是指对将要保存在磁盘等物理介质的数据进行压缩。

(3) 无损压缩与有损压缩。无损压缩是利用数据的统计冗余进行压缩，通常无损压缩的压缩比较低。而有损压缩是利用了人类对视觉、听觉对图像、声音中的某些频率成分不敏感的特性，允许压缩的过程中损失一定的信息。

压缩编码格式：联合图像专家小组标准 (JPEG)、动态图像视频编码标准 (MPEG)、视频编解码器标准 (H.26L)。

4.虚拟现实 (VR) /增强现实 (AR) 技术

VR 是一种可以创建和体验虚拟世界的计算机仿真系统，它利用计算机生成一种模拟环境，使用户沉浸到该环境中，让人有种身临其境的感觉。

VR 采用计算机技术生成一个逼真的视觉、听觉、触觉、味觉及嗅觉的感知系统，用户可以用人的自然技能与这个生成的虚拟实体进行交互操作，其概念包含 3 层含义。

- 虚拟实体是用计算机生成的一个逼真的实体。
- 用户可以通过人的自然技能（头部转动、眼动、手势或其他身体动作）与该环境交互。
- 要借助一些三维传感设备来完成交互动作，常用的有头盔立体显示器、数据手套、数据服装和三维鼠标等。

增强现实技术是指把原本在现实世界的一定时间和空间范围内很难体验到的实体信息(视觉信息、声音、味道和触觉等),通过模拟仿真后,再叠加到现实世界中被人类感官所感知,从而达到超越现实的感官体验。增强现实的出现与计算机图形图像技术、空间定位技术和人文智能等技术的发展密切相关。

(1) 计算机图形图像技术。增强现实的用户可以戴上透明的护目镜,透过它看到整个世界,连同计算机生成而投射到这一世界表面的图像,从而使物理世界的景象超出用户的日常经验之外。这种增强的信息可以是在真实环境中与之共存的虚拟物体,也可以是实际存在的物体的非几何信息。

(2) 空间定位技术。为了改善效果,增强现实所投射的图像必须在空间定位上与用户相关。当用户转动或移动头部时,视野变动,计算机产生的增强信息随之做相应的变化。

(3) 人文智能。该技术以将处理设备和人的身心能力结合起来为特点,并非仿真人的智能,而是试图发挥传感器、可穿戴计算等技术的优势,使人们能够捕获自己的日常经历、记忆及所见所闻,并与其他人进行更有效的交流。

VR/AR 技术主要分为桌面式、分布式、沉浸式和增强式 4 种。

名称	定义	特点
桌面式 VR	利用计算机形成三维交互场景,通过鼠标、力矩球等输入设备交互,由屏幕呈现出虚拟环境	易实现、应用广泛、成本较低,但因会受到环境干扰而缺乏体验感
分布式 VR	将 VR 与网络技术相融合,在同一 VR 环境下,多用户之间可以相互共享任何信息	忽略地域限制因素,共享度高,同时研发成本极高,适合专业领域
沉浸式 VR	借助各类型输入设备与输出设备,给予用户一个可完全沉浸,全身心参与的环境	良好的实时交互性和体验感,但对硬件配置、混合技术要求较高,开发成本高
增强式 VR (AR)	将虚拟现实模拟仿真的世界与现实世界叠加到一起,用户无须脱离真实世界即可提高感知	体验更完美,但对混合技术要求更高,开发成本高,起步晚

VR/AR 仍需要研究的关键技术:数据采集与优化传输技术、交互与情形实时再现技术。

3 系统工程

系统工程是利用计算机作为工具,对系统的结构、元素、信息和反馈等进行分析,以达到最优规划、最优设计、最优管理和最优控制的目的。

系统之系统 (SoS) 适用于其系统元素本身也是系统的情况。

系统工程方法是一种现代的科学决策方法,也是一门基本的决策技术。它针对主要问题、主要情况和全过程,运用有效工具进行全面的分析和处理。包括:

1.霍尔的三维结构。集中体现了系统工程方法的系统化、综合化、最优化、程序化和标准化等特点,是系统工程方法论的重要基础内容。

霍尔三维结构是将系统工程整个活动过程分为前后紧密衔接的 7 个阶段和 7 个步骤,同时还考虑了为完成这些阶段和步骤所需要的各种专业知识和技能。这样,就形成了由时间维、逻辑维和知识维组成的三维空间结构。

其中,时间维表示系统工程活动从开始到结束按时间顺序排列的全过程,分为规划、拟订方案、研制、生产、安装、运行、更新 7 个时间阶段。

逻辑维是指时间维的每个阶段内所要进行的工作内容和应该遵循的思维程序,包括明确问题、确定目标、系统综合、系统分析、优化、决策、实施 7 个逻辑步骤。

知识维需要运用包括工程、医学、建筑、商业、法律、管理、社会科学、艺术等各种知识和技能。

2.切克兰德方法。社会经济系统中的问题往往很难和工程技术系统中的问题一样,事先将需求确定清楚,难以接价值系统的评价准则设计出符合这种需求的最优系统方案。切克兰德方法论的核心不是“最优化”而是“比较”与“探寻”。从模型和现状的比较中来学习改善现状的途径。“比较”这一步骤,含有组织讨论、听取各方面有关人员意见的意思,不拘泥于非要进行定量分析的要求,能更好地反映人的因素和社会经济系统的特点。

切克兰德方法将工作过程分为 7 个步骤。

(1)认识问题。收集与问题有关的信息,表达问题现状,寻找构成和影响因素及其关系,以便明确系统问题结构、现存过程及其相互之间的不适应之处,确定有关的行为主体和利益主体。

(2)根底定义。初步弄清、改善与现状有关的各种因素及其相互关系。根底定义的目的是弄清系统问题的关键要素以及关联因素,为系统的发展及其研究确立各种基本的看法,并尽可能选择出最合适的基本观点。

(3)建立概念模型。在不能建立精确数学模型的情况下,用结构模型或语言模型来描述系统的现状。概念模型来自于根底定义,是通过系统化语言对问题抽象描述的结果,其结构及要素必须符合根底定义的思想,并能实现其要求。

(4)比较及探寻。将现实问题和概念模型进行对比,找出符合决策者意图且可行的方案或途径。有时通过比较,需要对根底定义的结果进行适当修正。

(5)选择。针对比较的结果,考虑有关人员的态度及其他社会、行为等因素,选出现实可行的改善方案。

(6)设计与实施。通过详尽和有针对性的设计,形成具有可操作性的方案,并使得有关人员乐于接受和愿意为方案的实现竭尽全力。

(7)评估与反馈。根据在实施过程中获得的新认识,修正问题描述、根底定义及概念模型等。

3.并行工程方法。是对产品及其相关过程(包括制造过程和支持过程)进行并行、集成化处理的系统方法和综合技术。它要求产品开发人员从设计开始就考虑产品生命周期的全过程,不仅考虑产品的各项性能,如质量、成本和用户要求,还应考虑与产品有关的各工艺过程的质量及服务的质量。

并行工程的目标是提高质量、降低成本、缩短产品开发周期和产品上市时间。

并行工程强调以下 3 点。

(1)在产品的设计开发期间,将概念设计、结构设计、工艺设计、最终需求等结合起来,保证以最快的速度按要求的质量完成。

(2)各项工作由与此相关的项目小组完成。进程中小组成员各自安排自身的工作,但可以随时或定期反馈信息,并对出现的问题协调解决。

(3) 依据适当的信息系统工具，反馈与协调整个项目的进行。利用现代 CIM 技术，在产品的研制与开发期间，辅助项目进程的并行化。

4.综合集成法。钱学森等首次把处理开放的复杂巨系统的方法命名为从定性到定量的综合集成法。综合集成是从整体上考虑并解决问题的方法论。是现代科学条件下认识方法论上的一次飞跃。

从系统的本质出发，根据组成子系统及子系统种类的多少和它们之间关联关系的复杂程度，可以把系统分为简单系统和巨系统两大类。

(1) 如果组成系统的子系统数量比较少，它们之间的关系比较单纯的系统称为简单系统

(2) 如果子系统数量非常巨大（如成千上万），则称作巨系统。

(2) 如巨系统中子系统种类不太多（几种、几十种），且它们之间的关联关系又比较简单，就称作简单巨系统，如激光系统。

(4) 如果子系统种类很多并有层次结构，它们之间的关联关系又很复杂，这就是复杂巨系统，如果这个系统又是开放的，就称作开放的复杂巨系统。

开放的复杂巨系统的一般基本原则与一般系统论的原则相一致：一是整体论原则；二是相互联系的原则；三是有序性原则；四是动态原则。

开放的复杂巨系统主要性质：开放性、复杂性、进化与涌现性、层次性、巨量性。

综合集成方法的主要特点有：

(1) 定性研究与定量研究有机结合，贯穿全过程。

(2) 科学理论与经验知识结合，把人们对客观事物的知识综合集成解决问题。

(3) 应用系统思想把多种学科结合起来进行综合研究。

(4) 根据复杂巨系统的层次结构，把宏观研究与微观研究统一起来。

(5) 必须有大型计算机系统支持，不仅有管理信息系统、决策支持系统等功能，而且还要有综合集成的功能。

5.WSR（物理-事理-人理）系统方法。WSR 是物理、事理和人理三者如何巧妙配置、有效利用以解决问题的一种系统方法论。“懂物理、明事理、通人理”就是 WSR 方法论的实践准则，

WSR 方法论一般工作过程可理解为这样的 7 步：理解意图、制定目标、调查分析、构造策略、选择方案、协调关系和实现构想。

系统工程生命周期 7 阶段

(1) 探索性研究阶段：目的是识别利益攸关者的需求，探索创意和技术。

(2) 概念阶段：目的是细化利益攸关者的需求，探索可行概念，提出有望实现的解决方案。

(3) 开发阶段：目的是细化系统需求，创建解决方案的描述，构建系统，验证并确认系统。

(4) 生产阶段：目的是生产系统并进行检验和验证。

(5) 使用阶段：目的是运行系统以满足用户需求。

(6) 保障阶段：目的是提供持续的系统能力。

(7) 退役阶段：目的是存储、归档或退出系统。

生命周期方法

(1) 计划驱动方法：特征在于整个过程始终遵守规定流程的系统化方法。特别关注文档的完整性、需求的可追溯性以及每种表示的事后验证。

(2) 渐进迭代式开发：允许为项目提供一个初始能力，随之提供连续交付以达到期望的系统。目标在于快速产生价值并提供快速响应能力。适合需求不清晰，较小的，不太复杂的系统。

(3) 精益开发：精益思想是一种整体性的范式，聚焦于向客户交付最大价值并使浪费活动最小化。是一个动态的、知识驱动的、以客户为中心的过程，通过这一过程使特定企业的所有人员以创造价值为目标不断地消除浪费。

(4) 敏捷开发：敏捷的关键目标在于灵活性，当风险可接受时允许从序列中排除选定的事件。适用于系统工程的敏捷原则如下：

- 最高的优先级是通过尽早地和持续地交付有价值的软件来满足客户。
- 欢迎需求变更，即使是在项目开发后期。敏捷流程利用需求变更帮助客户获得竞争优势。
- 不断交付可用的软件，周期从几周到几个月不等，且越短越好。
- 在项目中业务人员与开发人员每天在一起工作，业务人员始终参与到开发工作中。
- 在开发团队内部和团队之间，传递信息最有效的方法是面对面交谈。
- 工作软件是进展的主要度量。
- 对技术的精益求精以及对设计的不断完善将提升敏捷性。
- 简单性（尽最大可能减少不必要的工作的艺术）是精髓。
- 最佳的架构、需求和设计出自于自组织的团队。
- 团队要定期反省如何能够做到更加高效，并相应地调整团队的行为。

基于模型的系统工程（MBSE）

是建模方法的形式化应用，以使建模方法支持系统需求、分析、设计、验证和确认等活动，这些活动从概念性设计阶段开始，持续贯穿到设计开发以及后来的所有生命周期阶段。

系统工程过程的三个阶段分别产生三种图形：在需求分析阶段，产生需求图、用例图及包图；在功能分析与分配阶段，产生顺序图、活动图及状态机图；在设计综合阶段，产生模块定义图、内部块图及参数图等。

MBSE 的三大支柱分别是建模语言（SysML）、建模工具（支持系统建模语言画图的计算机和网络环境）和建模思路（如何利用建模语言的图形来建模）。

七 系统配置与性能评价

1 性能指标

性能指标，是软、硬件的性能指标的集成。在硬件中，包括计算机、各种通信交换设备、各类网络设备等；在软件中，包括：操作系统、协议以及应用程序等。

1. 计算机

对计算机评价的主要性能指标有：时钟频率（主频）；运算速度；运算精度；内存的存储容量；存储器的存取周期；数据处理速率 PDR（processing data rate）；吞吐率；各种响应时间；各种利用率；RASIS 特性（即：可靠性 Reliability、可用性 Availability、可维护性 Serviceability、完整性和安全性 Integrity and Security）；平均故障响应时间；兼容性；可扩充性；性能价格比。

2、路由器

对路由器评价的主要性能指标有：设备吞吐量、端口吞吐量、全双工线速转发能力、背靠背帧数、路由表能力、背板能力、丢包率、时延、时延抖动、VPN 支持能力、内部时钟精度、队列管理机制、端口硬件队列数、分类业务带宽保证、RSVP、IP Diff Serv、CAR 支持、冗余、热插拔组件、路由器冗余协议、网管、基于 Web 的管理、网管类型、带外网管支持、网管粒度、计费能力 / 协议、分组语音支持方式、协议支持、语音压缩能力、端口密度、信令支持。

3、交换机

对交换机评价的主要性能指标有：交换机类型、配置、支持的网络类型、最大 ATM 端口数、最大 SONET 端口数、最大 FDDI 端口数、背板吞吐量、缓冲区大小、最大 MAC 地址表大小、最大电源数、支持协议和标准、路由信息协议 RIP、RIP2、开放式最短路径优先第 2 版、边界网关协议 BGP、无类域间路由 CIDR、互联网成组管理协议 IGMP、距离矢量多播路由协议 DVMRP、开放式最短路径优先多播路由协议 MOSPF、协议无关的多播协议 PIM、资源预留协议 RSVP、802.1p 优先级标记，多队列、路由、支持第 3 层交换、支持多层（4 到 7 层交换、支持多协议路由、支持路由缓存、可支持最大路由表数、VLAN、最大 VLAN 数量、网管、支持网管类型、支持端口镜像、QoS、支持基于策略的第 2 层交换、每端口最大优先级队列数、支持基于策略的第 3 层交换、支持基于策略的应用级 QoS、支持最小 / 最大带宽分配、冗余、热交换组件（管理卡，交换结构，接口模块，电源，冷却系统、支持端口链路聚集协议、负载均衡。

4、网络

评价网络的性能指标有：设备级性能指标；网络级性能指标；应用级性能指标；用户级性能指标；吞吐量。

5、操作系统

评价操作系统的性能指标有：系统的可靠性、系统的吞吐率（量）、系统响应时间、系统资源利用率、可移植性。

6、数据库管理系统

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，有：数据库的大小、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、最大连接数等等。

7、WEB 服务器

评价 Web 服务器的主要性能指标有：最大并发连接数、响应延迟、吞吐量。

2 性能评价方法

性能评测的常用方法

- (1) 时钟频率。一般来讲，主频越高，速度越快。
- (2) 指令执行速度。计量单位 KIPS、MIPS。
- (3) 等效指令速度法。统计各类指令在程序中所占比例，并进行折算，是一种固定比例法。
- (4) 数据处理速率（Processing Data Rate，PDR）法。采用计算 PDR 值的方法来衡量机器性能，PDR 值越大，机器性能越好。PDR 与每条指令和每个操作数的平均位数以及每条指令的平均运算速度有关。

基准程序法（Benchmark）

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）。是目前被用户一致承认的测试性能的较好方法，有多种多样的基准程序，包括：

- (1) 整数测试程序。同一厂家的机器，采用相同的体系结构，用相同的基准程序测试，得到的 MIPS 值越大，一般说明机器速度越快。
- (2) 浮点测试程序。指标 MFLOPS（理论峰值浮点速度）。
- (3) SPEC 基准程序（SPEC Benchmark）。重点面向处理器性能的基准程序集，将被测计算机的执行时间标准化，即将被测计算机的执行时间除以一个参考处理器的执行时间，
- (4) TPC 基准程序。用于评测计算机在事务处理、数据库处理、企业管理与决策支持系统等 方面的性能。其中，TPC-C 是在线事务处理（On-line Transaction Processing，OLTP）的基准程序，TPC-D 是决策支持的基准程序。TPC-E 作为大型企业信息服务的基准程序。

大多数情况下，为测试新系统的性能，用户必须依靠评价程序来评价机器的性能。下面列出了 4 种评价程序，它们评测的准确程度依次递减：真实的程序、核心程序、小型基准程序、合成基准程序。

3 阿姆达尔解决方案

阿姆达尔（Amdahl）定律主要用于系统性能改进的计算中。阿姆达尔定律是指计算机系统中对某一部件采用某种更快的执行方式所获得的系统性能改变程度，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。假定我们使用某种增强部件，计算机的性能就会得到提高，那么加速比就是下式所定义的比率：

$$\text{加速比} = \frac{\text{不使用增强部件时完成整个任务的时间}}{\text{使用增强部件时完成整个任务的时间}}$$

$$\text{总加速比} = \frac{\text{原来的执行时间}}{\text{新的执行时间}} = \frac{1}{\left((1 - \text{增强比例}) + \frac{\text{增强比例}}{\text{增强加速比}} \right)}$$

八 信息系统基础知识

1 信息系统概述

信息系统是由计算机硬件、网络和通信设备、计算机软件、信息资源、信息用户和规章制度组成的以处理信息流为目的的人机一体化系统。

信息系统的 5 个基本功能：输入、存储、处理、输出和控制。

信息系统的性质影响着系统开发者和系统用户的知识需求。“以计算机为基础”要求系统设计者必须具备计算机及其在信息处理中的应用知识。“人机交互”要求系统设计者还需要了解人作为系统组成部分的能力以及人作为信息使用者的各种行为。

诺兰模型：信息系统进化的阶段模型。将计算机信息系统的发展道路划分为 6 个阶段：

1. 初始阶段：计算机刚进入企业时只作为办公设备使用，应用非常少。一般仅用于财务部门。
2. 传播阶段：企业对计算机有了一定了解，想利用计算机解决工作中的问题，比如进行更多的数据处理，给管理工作和业务带来便利。会大幅度增加软件投入，盲目投入产生问题，效率低。
3. 控制阶段：从整体上控制计算机信息系统的发展，在客观上要求组织协调、解决数据共享问题。一些职能部门内部实现了网络化，但各软件系统之间还存在“部门壁垒”与“信息孤岛”。信息系统呈现单点、分散的特点，系统和资源利用率不高。这一阶段是计算机管理变为数据管理的关键。
4. 集成阶段：在控制的基础上，企业开始重新进行规划设计，建立基础数据库，并建成统一的信息管理系统。使人、财、物等资源信息能够在企业集成共享，更有效地利用现有的 IT 系统和资源。
5. 数据管理阶段：企业高层意识到信息战略的重要，信息成为企业的重要资源，企业的信息化建设也真正进入到数据处理阶段。使用统一平台，各部门、各系统基本实现资源整合和信息共享。
6. 成熟阶段：信息系统已经可以满足企业各个层次的需求，从简单的事务处理到支持高效管理的决策。企业真正把 IT 与管理过程结合起来，将组织内部、外部的资源充分整合和利用，从而提升了企业的竞争力和发展潜力。

信息系统的分类（低级到高级）

1. 业务（数据）处理系统（TPS/DPS）：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业务（数据）的管理，如财会管理、销售管理、物资管理和生产管理等，即计算机应用发展到对企业的局部事务的管理。

2. 管理信息系统 (MIS): 由人和计算机等组成的, 能进行管理信息的收集、传输、存储、加工、维护和使用的系统。形成了对企业全局性的、整体性的计算机应用。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内部。

3. 决策支持系统 (DSS): 帮助决策者利用数据和模型去解决半结构化决策问题和非结构化决策问题的交互式系统。服务于高层决策的管理信息系统, 按功能可分为专用 DSS、DSS 工具和 DSS 生成器。

4. 专家系统 (ES): 一个智能计算机程序系统, 其内部含有某个领域具有专家水平的大量知识与经验, 能够利用人类专家的知识和解决问题的方法来处理该领域的问题。

5. 办公自动化系统 (OAS): 人机结合的综合性的办公事务管理系统, 或称办公事务处理系统。该系统将当代各种先进技术和设备应用于办公室的办公活动中, 使办公活动实现科学化、自动化, 以达到改善工作环境、最大限度地提高办公事务工作质量和工作效率。

目前企业主要使用的信息化系统主要有 ERP 系统 (企业资源管理)、WMS 系统 (仓储管理系统)、MES 系统 (也称之为 SFC, 即制造过程管理系统) 和产品数据管理系统 (PDM)。

- ERP 系统: 主要管理公司的各种资源, 负责处理进销存、供应链、生产计划 MPS、MRP 计算、生产订单、管理会计, 是财务数据的强力支撑。

- WMS 系统: 主要包括库房货位管理, 主要有收发料, 通过扫码进出库, 对库存进行库位、先进先出与盘点; 栈板出货管控、库龄管理等内容, 主要是立体仓库或大批量仓库数据需求。

- MES 系统: 负责生产过程和生产过程中防呆、自动化设备集成, 是各个客户审核的重点, 是生产全流程管控, 也有企业称之为 SFC, 其实大同小异, 但是它是生产过程、生产工艺、生产设备、自动化生产直接的核心。

- PDM 系统: 管理研发阶段的物料、BOM、工程变更数据, 负责产品数据为主。PDM 系统是产品研发全过程管理, 主要涉及协同研发等能力。

信息系统的生命周期

1. 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地调研和分析, 并形成需求分析报告。

2. 信息系统的开发阶段: 最重要、关键的阶段。包括总体规划、系统分析、系统设计、系统实施和系统验收这 5 个阶段。

(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。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。总体规划产出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。目标是为系统设计阶段提供系统的逻辑模型。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。内容主要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。根据系统分析的结果，设计出信息系统的实施方案。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。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。系统实施阶段以后，用户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，系统性能的优劣以及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。系统维护包括即排错性维护、适应性维护、完善性维护和预防性维护。

4. 信息系统的消亡阶段：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。

信息系统建设的原则：高层管理人员介入原则、用户参与开发原则、自顶向下规划原则、工程化原则、其他原则（创新性，整体性，发展性，经济性等）。

信息系统开发方法

1. 结构化方法。结构是指系统内各个组成要素之间的相互联系、相互作用的框架。结构化方法也称为生命周期法，是一种传统的信息系统开发方法，由结构化分析（Structured Analysis, SA）、结构化设计（Structured Design, SD）和结构化程序设计（Structured Programming, SP）三部分有机组合而成，其精髓是自顶向下、逐步求精和模块化设计。

结构化方法的主要特点

(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。

(2) 开发工作阶段化。每个阶段工作完成后，要根据阶段工作目标和要求进行审查，这使各阶段工作有条不紊地进行，便于项目管理与控制。

(3) 开发文档规范化。结构化方法每个阶段工作完成后，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遍历。

(4) 设计方法结构化。在系统分析与设计时，从整体和全局考虑，自顶向下地分解；在系统实现时，根据设计的要求，先编写各个具体的功能模块，然后自底向上逐步实现整个系统。

结构化方法的不足和局限

(1) 开发周期长：按顺序经历各个阶段，直到实施阶段结束后，用户才能使用系统。

(2) 难以适应需求变化：不适用于需求不明确或经常变更的项目。

(3) 很少考虑数据结构：结构化方法是一种面向数据流的开发方法，很少考虑数据结构。

结构化方法常用工具

结构化方法一般利用图形表达用户需求，常用工具有数据流图、数据字典、结构化语言、判定表以及判定树等。

2. 原型化方法

原型化方法也称为快速原型法，或者简称为原型法。它是一种根据用户初步需求，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。

按是否实现功能分类：分为水平原型（行为原型，功能的导航）、垂直原型（结构化原型，实现了部分功能）。

按最终结果分类：分为抛弃式原型、演化式原型。

原型法的特点

原型法可以使系统开发的周期缩短、成本和风险降低、速度加快，获得较高的综合开发效益。

原型法是以用户为中心来开发系统的，用户参与的程度大大提高，开发的系统符合用户的需求，因而增加了用户的满意度，提高了系统开发的成功率。

由于用户参与了系统开发的全过程，对系统的功能和结构容易理解和接受，有利于系统的移交，有利于系统的运行与维护。

原型法的不足之处

开发的环境要求高。

管理水平要求高。

由以上的分析可以看出，原型法的优点主要在于能更有效地确认用户需求。从直观上来看，原型法适用于那些需求不明确的系统开发。事实上，对于分析层面难度大、技术层面难度不大的系统，适合于原型法开发。

从严格意义上来说，目前的原型法不是一种独立的系统开发方法，而只是一种开发思想，它只支持在系统开发早期阶段快速生成系统的原型，没有规定在原型构建过程中必须使用哪种方法。因此，它不是完整意义上的方法论体系。这就注定了原型法必须与其他信息系统开发方法结合使用。

3.面向对象方法。面向对象（Object-Oriented，OO）方法认为，客观世界是由各种对象组成的，任何事物都是对象，每一个对象都有自己的运动规律和内部状态，都属于某个对象类，是该对象类的一个元素。复杂的对象可由相对简单的各种对象以某种方式而构成，不同对象的组合及相互作用就构成了系统。

面向对象方法的特点

使用 OO 方法构造的系统具有更好的复用性，其关键在于建立一个全面、合理、统一的模型（用例模型和分析模型）。

OO 方法也划分阶段，但其中的系统分析、系统设计和系统实现三个阶段之间已经没有“缝隙”。也就是说，这三个阶段的界限变得不明确，某项工作既可以在前一个阶段完成，也可以在后一个阶段完成；前一个阶段工作做得不够细，在后一个阶段可以补充。

面向对象方法可以普遍适用于各类信息系统的开发。

面向对象方法的不足之处

必须依靠一定的面向对象技术支持，在大型项目的开发上具有一定的局限性，不能涉足系统分析以前的开发环节。

当前，一些大型信息系统的开发，通常是将结构化方法和 OO 方法结合起来。首先，使用结构化方法进行自顶向下的整体划分；然后，自底向上地采用 OO 方法进行开发。因此，结构化方法和 OO 方法仍是两种在系统开发领域中相互依存的、不可替代的方法。

4.面向服务的方法。面向服务（Service-Oriented，SO）的方法：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。

2 业务处理系统（TPS）

又可称为电子数据处理系统（EDP），最初级形式的信息系统。针对管理中具体的事务（如财会、销售、库存等）来辅助管理人员将所发生的数据进行记录、传票、记账、统计和分类，并制成报表等活动，为经营决策提供有效信息的基于计算机的信息系统。

由于 TPS 的主要功能就是对企业管理中日常事务所发生的数据进行输入、处理和输出。因此，TPS 的数据处理周期由 5 个阶段构成：数据输入、数据处理、数据库的维护、文件报表的生成和查询处理。

1. 数据输入。主要解决如何将企业经营活动中产生的大量原始数据准确、迅速地输入到计算机系统中并存储起来，这是信息系统进行信息处理的“瓶颈”。因此，数据的输入方式和进度是这个阶段的关键问题。常见的数据输入方式有 3 种，即人工、自动及二者结合。

2. 数据处理。TPS 中常见的数据处理方式有两种，一种是批处理方式（将事务数据积累到一段时间后定期进行定期处理）；另一种是联机事务处理方式（OLTP，实时处理）。

3. 数据库的维护。一个组织的数据库通过 TPS 来更新，以确保数据库中的数据能及时、正确地反映当前最新的经营状况，因此数据库的维护是 TPS 的一项主要功能。对数据库的访问形式基本有 4 种：检索、修改、存入和删除。

4. 文件报表的产生。TPS 的输出就是为终端用户提供所需的有关文件和报表，这些文件和报表根据其用途不同可分为：行动文件（该文件的接收者持有文件后可进行某项事务处理）、信息文件（类文件向其持有者表明某项业务已发生了）、周转文件（交给接受者之后通常还要返回到发送者手中）。

5. 查询处理。TPS 支持终端用户的批次查询或联机实时查询，典型的查询方式是用户通过屏幕显示获得查询结果。

特点：TPS 是其他类型信息系统的信息产生器，企业在推进全面信息化的过程中往往是从开发 TPS 入手的。许多 TPS 是处于企业系统的边界，它是将企业与外部环境联系起来的“桥梁”。因此，TPS 性能的好坏将直接影响着组织的整体形象，是提高企业市场竞争力的重要因素。由于 TPS 面对的是结构化程度很高的管理问题，因此可以采用结构化生命周期法来进行开发。

	销售市场	生产	后勤	人事	财务会计	信息处理	高层管理
战略规划							
管理控制							
运行控制							
业务处理							

(1) 销售市场子系统。它包括销售和推销，在运行控制方面包括雇用和训练销售人员、销售和推销的日常调度，还包括按区域、产品、顾客销售数量的定期分析等。

(2) 生产子系统。它包括产品设计、生产设备计划、生产设备的调度和运行、生产人员的雇用和训练、质量控制和检查等。

(3) 后勤子系统。它包括采购、收货、库存控制和分发。

(4) 人事子系统。它包括雇用、培训、考核记录、工资和解雇等。

(5) 财务和会计子系统。财务的目标是保证企业的财务要求，并使其花费尽可能的低；会计则是把财务业务分类、总结，填入标准财务报告，准备预算、成本数据的分析与分类等。

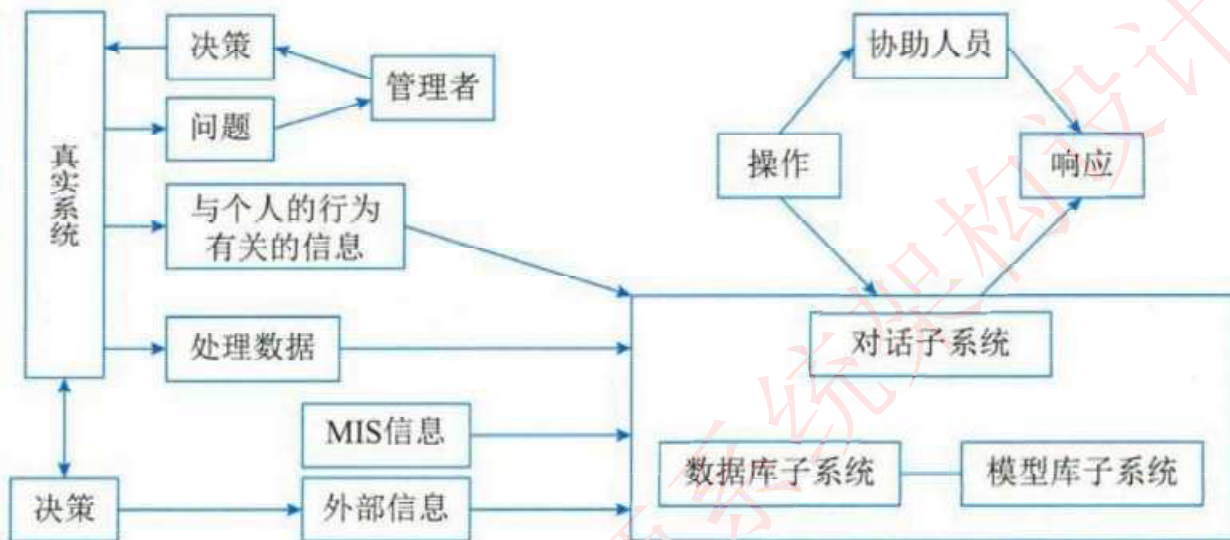
(6) 信息处理子系统。该系统的作用是保证企业的信息需要。典型的任务是处理请求、收集数据、改变数据和程序的请求、报告硬件和软件的故障及规划建议等。

(7) 高层管理子系统。为高层领导服务。业务包括查询信息和支持决策，编写文件和信件，向公司其他部门发送指令。

4 决策支持系统（DSS）

DSS 应当是一个交互式的、灵活的、适应性强的基于计算机的信息系统，能够为解决非结构化管理问题提供支持，以改善决策的质量。

DSS 的基本模式反映 DSS 的形式及其与“真实系统”、人和外部环境的关系，如图所示。其中管理者处于核心地位，运用自己的知识和经验，结合决策支持系统提供的支持，对其管理的“真实系统”进行决策。



DSS 的两种基本结构形式是两库结构和基于知识的结构，实际中的 DSS 由这两种基本结构通过分解或增加某些部件演变而来。两库结构由数据库子系统、模型库子系统和对话子系统形成三角形分布的结构：



决策支持系统的总体功能是支持各种层次的人们进行决策。具体可细分为：

- (1) 决策支持系统用来整理和提供本系统与决策问题有关的各种数据。
- (2) 决策支持系统要尽可能地收集、存储和及时提供与决策有关的外部信息。
- (3) 决策支持系统能及时收集和提供有关各项活动的反馈信息。
- (4) 决策支持系统对各种与决策有关的模型具有存储和管理的能力。
- (5) 决策支持系统提供对常用的数学方法、统计方法和运筹方法的存储和管理。
- (6) 决策支持系统能对各种数据、模型、方法进行有效管理，为用户提供查找、变更、增加、删除等操作功能。
- (7) 决策支持系统运用所提供的模型和方法对数据进行加工，并得出有效支持决策的信息。
- (8) 决策支持系统具有人-机对话接口和图形加工、输出功能，不仅用户可以对所需要的数据进行查询，而且可以输出相应的图形。
- (9) 决策支持系统应能支持分布使用方式，提供有效的传输功能，以保证分散在不同地点的用户能共享系统所提供的模型、方法和可共享的数据

决策支持系统的特点

- (1) 决策支持系统面向决策者。
- (2) 决策支持系统支持对半结构化问题的决策。

(3) 决策支持系统的作用是辅助决策者、支持决策者。

(4) 决策支持系统体现决策过程的动态性。

(5) 决策支持系统提倡交互式处理。

决策支持系统的组成

1. 数据的重组和确认。与决策支持系统相关的数据库的问题是，获得正确的数据并且可用理想的形式操作这些数据。这个问题可以通过数据仓库的概念解决。

2. 数据字典的建立。数据仓库是一个与作业层系统分离存在的数据库。通过对数据仓库的存取，管理者可以做出以事实为根据的决策来解决许多业务问题。

3. 数据挖掘和智能体。一旦建成数据仓库，管理者们需要运用工具进行数据存取和查询，使用的工具称为智能体。数据挖掘的结果类型包括：

(1) 联合：把各个事件联系在一起的过程。例如，将学生们经常同时选修的两门课程联系起来，以便这两门课程不被安排在同一时间。

(2) 定序：识别模式的过程。例如，识别学生们多个学期课程的次序。

(3) 分类：根据模式组织数据的过程。例如，以学生完成学业的时间（4 年以内，4 年以上）为标准分成几个小组。

(4) 聚类：推导特定小组与其他小组相区分的判断规则的过程。例如，通过兴趣、年龄、工作经验来划分学生。

4. 模型建立。模型管理的目的就是帮助决策者理解与选择有关的现象。建立模型的方法有穷枚举法、算法、启发式和模拟法。

5 专家系统 (ES)

基于知识的专家系统简称为专家系统，是人工智能的一个重要分支。专家系统的能力来自于它所拥有的专家知识，知识的表示及推理的方法则提供了应用的机理。

专家系统既不同于传统的应用程序，也不同于其他类型的人工智能问题求解程序。不同点主要表现在以下 5 个方面。

(1) 专家系统属于人工智能范畴，其求解的问题是半结构化或非结构化问题。

(2) 专家系统模拟的是人类专家在问题领域的推理，而不是模拟问题领域本身。

(3) 专家系统由 3 个要素组成：描述问题状态的综合数据库、存放启发式经验知识的知识库和对知识库的知识进行推理的推理机。三要素分别对应数据级、知识库级和控制级三级知识，而传统应用程序只有数据和程序两级结构。

(4) 专家系统处理的问题是实际的问题，而不是纯学术的问题。

(5) 从求解手段来看，专家系统专用性强，通用性差。

人工智能 (AI) 旨在利用机械、电子、光电或生物器件等制造的装置或机器模仿人类的智能。

AI 研究的重点放在开发具有智能行为的计算机系统上，智能行为表现出以下 5 个特点。

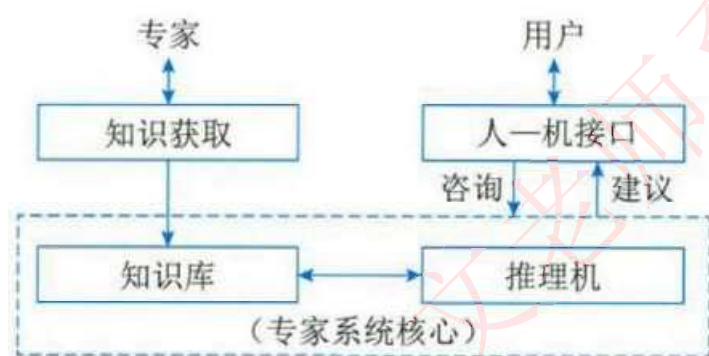
(1) 从过去的事件或情形中汲取经验，并将从经验中得到的知识应用于新的环境和场景。

- (2) 具有在缺乏重要信息时解决问题的能力。
- (3) 具有处理和操纵各种符号、理解形象化图片（图像）的能力。
- (4) 想象力和创造力。
- (5) 善于启发。

人工智能是一个极为广泛的领域， AI 的主要分支有专家系统、机器人技术、视觉系统、自然语言处理、学习系统和神经网络等。

专家系统的特点：超越时间限制、操作成本低廉、易于传递与复制、处理手段一致（不会因人而异）、善于克服难题、适用特定领域。

专家系统的组成



1. 知识库。专家系统的知识库用来存放系统求解实际问题的领域知识。一般来说，知识库中的知识可分成两类：一类为事实性知识；另一类是启发性知识。

2. 综合数据库。是专家系统在执行与推理过程中用以存放所需要和产生的各种信息的工作存储器，通常包括欲解决问题的初始状态描述、中间结果、求解过程的记录、用户对系统提问的回答等信息，因此，综合数据库又叫动态知识库，其内容在系统运行过程中是不断变化的。相应地把专家系统的知识库称为静态知识库。二者一起构成完整知识库。

3. 推理机。推理机和知识库一起构成专家系统的核心。推理机也被称为控制结构或规则解释器，通常包括推理机制和控制策略，是一组用来控制系统的运行、执行各种任务、根据知识库进行各种搜索和推理的程序模块。

4. 知识获取。主要有两方面功能：一是知识的编辑和求精；二是知识自学习。

5. 解释程序。是面向用户服务的，负责解答用户提出的各种问题。

6. 人机接口。通常包括两部分：一部分是专家系统与用户的接口；另一部分是专家系统与领域专家和知识工程师的接口。

6 办公自动化系统（OAS）

办公自动化就是办公信息处理手段的自动化。OAS 要解决的是包括数据、文字、声音、图像等信息的一体化处理问题。从本质上讲，办公自动化就是以先进的科学技术为基础，利用有关办公自动化设备协助办公人员管理各项办公信息，主要利用资源以提高办公效率和办公质量。计算机技术、通信

技术、系统科学和行为科学是它的 4 大支柱。其中以行为科学为主导，系统科学为理论基础，结合运用计算机技术和通信技术。

OAS 的主要功能

1. 事务处理。完成办公部分内的大量繁琐事情，又称为事务办公系统，分为单机处理系统和多机处理系统（通信、信息共享）。

2. 信息管理。对信息流的控制管理，主要包括信息的收集、加工、传递、交流、存取、提供、分析、判断、应用和反馈那些办公人员的综合性工作。可称为管理型办公系统，它可将事务型办公系统中各项孤立的事务处理通过信息交换和共享资源联系起来，获得准确、快捷、及时、优质的功效。管理型办公系统是一种分布式的处理系统，具有计算机通信和网络功能。

3. 辅助决策。可称为决策型办公系统，以经理型办公系统提供的大量信息作为决策工作的基础，建立起能综合分析、预测发展、判断利弊的计算机可运行的决策模型，根据原始数据信息，自动做出比较符合实际的决策方案。

办公自动化系统的组成：计算机设备、办公设备、数据通信及网络设备、软件系统。

7 企业资源规划（ERP）

企业资源规划是指建立在信息技术基础上，以系统化的管理思想，为企业提供决策和运营手段的管理平台。**ERP** 系统是将企业所有资源进行集成整合，并进行全面、一体化管理的信息系统。

演变过程：物料需求计划（物料单系统）——制造资源计划（增加库存、分销等）——企业资源计划（打通了供应链，加入财务、人力资源、销售管理等）。

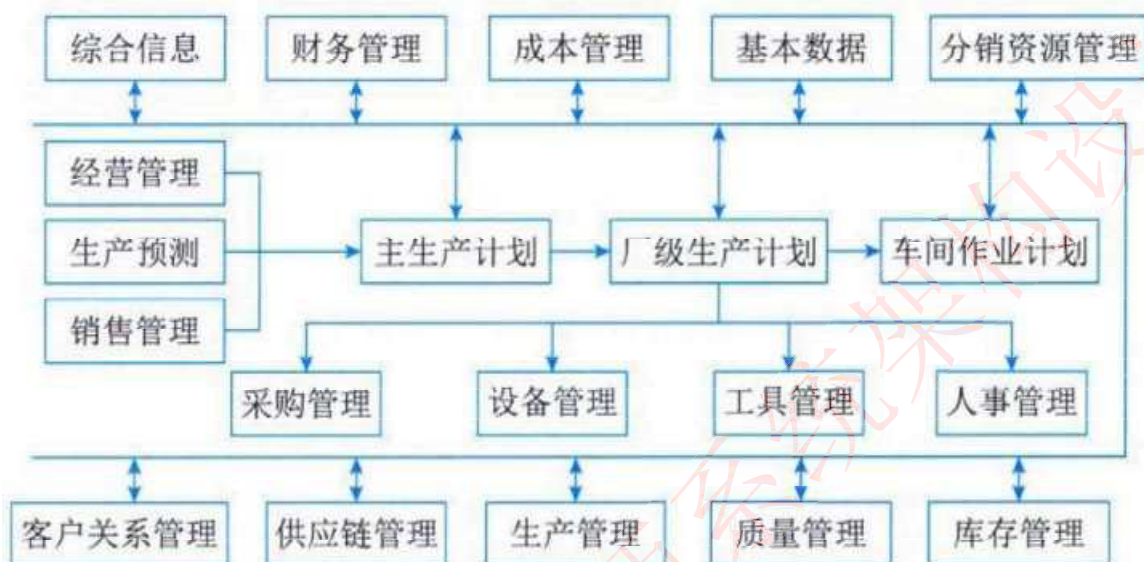
企业有三大资源：物流（物流管理）、资金流（财务管理）、信息流（生产控制管理）、现在一般认为人力资源（人力资源管理）是企业第四大资源。企业的资源计划可从下面三点来理解：

管理思想：ERP 首先是一种管理思想，将企业资源分类管理，是管理思想的变革。

软件产品：其次，ERP 是个软件产品，为企业用户提供一体化的解决方案，不是买来直接用的，需要个性化的开发和部署。

管理系统：ERP 最后是一个管理系统，存在众多的子系统，这些子系统有统一的规划，是互联互通的，便于事前事中监控。

ERP 的结构



1) 生产预测。市场需求是企业生存的基础，在 ERP 中首先需要对市场进行较准确的预测。预测主要用于计划，在 ERP 的 5 个层次的计划中，前 3 个层次计划，即经营计划、生产计划大纲和主生产计划的编制都离不开预测。

2) 销售管理（计划）。销售管理主要是针对企业的销售部门的相关业务进行管理。

3) 经营计划（生产计划大纲）。生产计划大纲（PP）是根据经营计划的生产目标制定的，是对企业经营计划的细化，用以描述企业在可用资源的条件下，在一定时期中的产量计划。生产计划大纲在企业决策层的 3 个计划中有承上启下的作用，一方面它是企业经营计划和战略规划细化，另一方面它又用于指导企业编制主生产计划，指导企业有计划地进行生产。

4) 主生产计划。主生产计划（MPS）是对企业生产计划大纲的细化，说明在一定时期内的下计划：生产什么，生产多少和什么时候交货。主生产计划的编制以生产大纲为准，其汇总结果应当等同于生产计划大纲，同时，主生产计划又是其下一层计划——物料需求计划的编制依据。

5) 物种需求计划。物料需求计划是对主生产计划的各个项目所需的全部制造件和全部采购件的网路支持计划和时间进度计划。它根据主生产计划对最终产品的需求数量和交货期，推导出构成产品的零部件及材料的需求数量和需求时期，再导出自制零部件的制作订单下达日期和采购件的采购订单发送日期，并进行需求资源和可用能力之间的进一步平衡。

6) 能力需求计划。能力需求计划（CRP）是对物料需求计划所需能力进行核算的一种计划管理方法。旨在通过分析比较 MRP 的需求和企业现有生产能力，及早发现能力的瓶颈所在，为实现企业的生产任务而提供能力方面的保障。

7) 车间作业计划。车间作业计划（PAC）是在 MRP 所产生的加工制造订单（即自制零部件生产计划）的基础上，按照交货期的前后和生产优先级选择原则以及车间的生产资源情况（如设备、人员、物料的可用性、加工能力的大小等），将零部件的生产计划以订单的形式下达给适当的车间。

8) 采购与库存管理。采购与库存管理是 ERP 的基本模块，其中采购管理模块是对采购工作，即从采购订单产生至货物收到的全过程进行组织、实施与控制，库存管理模块则是对企业物料的进、出、存进行管理。

9) 质量与设备管理。质量管理贯穿于企业经营的始终。设备管理是指依据企业的生产经营目标,通过一系列的技术、经济和组织措施,对设备寿命周期内的所有设备物资运动形态和价值运动形态进行的综合管理。

10) 财务管理。会计工作是企业管理的重要组成部分,是以货币的形式反映和监督企业的日常经济活动,并对这些经济业务的数据进行分类、汇总,以便为企业管理和决策提供必要的信息支持。企业财务管理是企业会计工作和活动的统称。

11) ERP 有关扩展应用模块。如客户关系管理、分销资源管理、供应链管理和电子商务等。这几个扩展模块本身也是一个独立的系统,在市场上它们常作为独立的软件产品进行出售和实施。

ERP 的功能: 支持决策的功能、为处于不同行业的企业提供有针对性的 IT 解决方案、从企业内部的供应链发展为全行业和跨行业的供应链。

8 典型信息系统架构模型

政府信息化和电子政务

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,将其管理和服务职能进行集成,在网络上实现政府组织结构和工作流程优化重组,超越时间、空间与部门分隔的制约,实现公务、政务、商务、事务的一体化管理与运行。电子政务主要包括 3 个组成部分:

- (1) 政府部门内部的电子化和网络化办公。
- (2) 政府部门之间通过计算机网络进行的信息共享和实时通信。
- (3) 政府部门通过网络与居民之间进行的双向信息交流。

电子政务的内容: G2G、G2B、G2C、B2G、C2G。

电子政务在世界范围内的发展有 2 个主要的特征:第 1 个特征是以互联网为基础设施,构造和发展电子政务。第 2 个特征是,就电子政务的内涵而言,更强调政府服务功能的发挥和完善,包括政府对企业、对居民的服务以及政府各部门之间的相互服务。

电子政务的发展大致经历了 4 个阶段:起步阶段、政府与用户单向互动、政府与用户双向互动、网上事务处理。

电子政务的应用领域:面向社会、政府部门之间、政府部门内部各类应用系统、涉及政府部门内部各类核心数据的应用系统、政府电子化采购、电子社区。

企业信息化和电子商务

企业信息化就是企业利用现代信息技术,通过信息资源的深入开发和广泛利用,实现企业生产过程的自动化、管理方式的网络化、决策支持的智能化和商务运营的电子化,不断提高生产、经营、管理、决策的效率和水平,进而提高企业经济效益和企业竞争力的过程。

企业信息化的具体目标是优化企业业务活动,使之更加有效,它的根本目的在于提高企业竞争能力,使得企业具有平稳和有效的运作能力,对紧急情况和机会做出快速反应,为企业内外部用户提供有价值的信息。包括技术创新、管理创新和制度创新。

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。需要从 3 个层面来实现。

(1) 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。达到战略上的融合。

(2) 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驱动力和实现这些目标的关键流程。

(3) 管理运作层面。虽然这一层面从价值链的角度上来说，是属于辅助流程，但它对企业日常管理的科学性、高效性是非常重要的。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。

企业信息化方法

(1) 业务流程重构方法。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。

(2) 核心业务应用方法。任何一家企业，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。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。

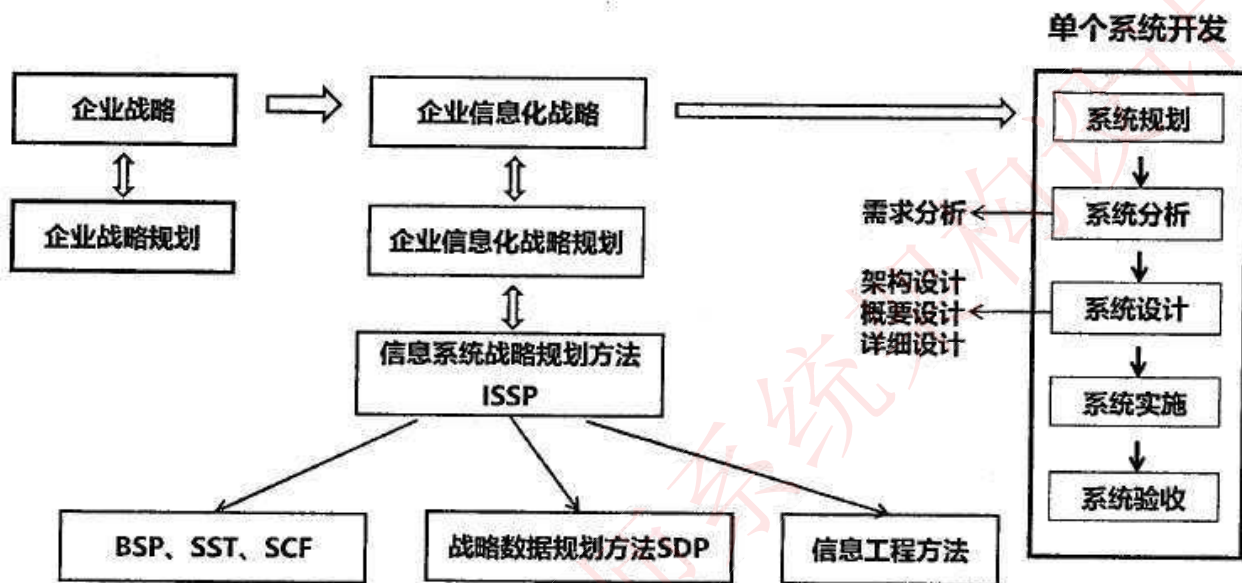
(5) 资源管理方法。目前，流行的企业信息化的资源管理方法有很多，最常见的有企业资源计划（ERP）、供应链管理（SCM）等。

(6) 人力资本投资方法。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是一种资本，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。

上面是第二版教材的内容，而以下是根据历年真题考察情况保留的原教材知识

9 信息化战略体系

体系结构图如下所示：



企业战略规划利用机会和威胁评价现在和未来的环境，用优势和劣势评价企业现状，进而选择和确定企业的总体和长远目标，制定和抉择实现目标的行动方案。

信息系统战略规划关注如何通过信息系统来支撑业务流程的运作，进而实现企业的关键业务目标，其重点在于对信息系统远景、组成架构、各部分逻辑关系进行规划。

信息技术战略规划：通常简称为 IT 战略规划，是在信息系统规划的基础上，对支撑信息系统运行的硬件、软件、支撑环境等进行具体的规划，它更关心技术层面的问题。

信息资源规划是在以上规划的基础上，为开展具体的信息化建设项目而进行的数据需求分析、信息资源标准建立、信息资源整合工作。

系统规划：单个项目的立项分析，是信息系统生命周期的第一个阶段，期任务是对企业的环境、目标及现有系统的状况进行初步调查，根据企业目标和发展战略，确定信息系统的发展战略，对建设新系统的需求做出分析和预测，同时考虑建设新系统所受的各种约束，研究建设新系统的必要性和可能性。

企业战略与信息化战略的集成主要方法有：业务与 IT 整合（Business-IT Alignment, BITA）和企业 IT 架构（Enterprise IT Architecture, EITA）。

BITA 是一种以业务为导向的、全面的 IT 管理咨询实施方法论。从制订企业战略、建立（改进）企业组织结构和业务流程，到进行 IT 管理和制订过度计划，使 IT 能够更好地为企业战略和目标服务。BITA 适用于信息系统不能满足当前管理中的业务需要，业务和 IT 之间总是有不一致的地方。

EITA 分析企业战略，帮助企业制订 IT 战略，并对其投资决策进行指导。在技术、信息系统、信息、IT 组织和 IT 流程方面，帮助企业建立 IT 的原则规范、模式和标准，指出 IT 需要改进的方面并帮助制订行动计划。EITA 适用于现有信息系统和 IT 基础架构不一致、不兼容和缺乏统一的整体管理的企业。

10 信息系统战略规划

一个企业信息系统的战略规划可分为下面三个阶段：

第一阶段：以数据处理为核心，围绕职能部门需求

企业系统规划法 **BSP**：自上而下地识别企业目标、企业过程和数据，然后对数据进行分析，自下而上地设计信息系统。重视数据的创建和使用，以数据的创建和使用归类，提供一个信息系统规划，建立 **CU** 矩阵（创建使用矩阵）。

关键成功因素法 **CSF**：重视关键因素，每个企业在某阶段都有关键因素，抓住关键信息。

战略集合转化法 **SST**：将企业的战略信息（环境、目标等）收集起来，当成一个“信息集合”，并且转换为信息系统的战略信息，全方位的注重企业的战略信息。

第二阶段：以企业内部 MIS（管理信息系统）为核心，围绕企业整体需求

战略数据规划法 **SDP**：强调建立企业模型和主题数据库（重点和关键，是面向业务主题，整个企业的），数据类基本上是稳定的，而业务和流程是多变的。

信息工程法 **IE**：第一次提出以工程的方法来建立信息系统，信息工程是面向企业计算机信息系统建设，以数据为中心的开发方法。信息工程方法认为，与企业的信息系统密切相关的三要素是：企业的各种信息、企业的业务过程和企业采用的信息技术。信息工程自上而下地将整个信息系统的开发过程划分为四个实施阶段，分别是信息规划阶段、业务领域分析阶段、系统设计阶段和系统构建阶段。

战略栅格法 **SG**：建立一个 2×2 的矩阵，每个矩阵元素代表过程对数据类的创建和使用等。栅格即划分矩阵。

第三阶段：综合考虑企业内外环境，以集成为核心，围绕企业战略需求

价值链分析法 **VCA**：将所有对企业有影响的信息作为一个个活动，其都有可能对企业造成增值，分析其中对企业增值最大的信息。

战略一致性模型 **SAM**：保证企业战略和信息系统战略要一致。

11 企业信息化与电子商务

11.1 客户关系管理 CRM

CRM 将客户看做是企业的一项重要资产，客户关怀是 **CRM** 的中心，其目的是与客户建立长期和有效的业务关系，在与客户的每一个“接触点”上都更加接近客户、了解客户，最大限度的增加利润。

CRM 的核心是客户价值管理，将客户价值分为既成价值、潜在价值和模型价值，通过“一对一”营销原则，满足不同价值客户的个性化需求，提高客户忠诚度和保有率。

CRM 的功能

客户服务：是 **CRM** 的关键内容，对客户提供的服务，可以提高客户忠诚度。

市场营销：包括商机产生、商机获取和管理、商业活动管理和电话营销等；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。

共享的客户资料库：是企业的一种重要信息资源，将市场营销和客户服务连接起来。也是 CRM 的基础和依托。

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。对上述获取的资料库进行分析。

CRM 是企业管理的前台，必须与 ERP 进行良好的集成，使得信息可以及时传递给财务、生产等部门。

市场营销和客户服务是 CRM 的支柱性功能。

一个有效的 CRM 解决方案应该具备以下要素：

畅通有效的客户交流渠道（触发中心）。

对所获信息进行有效分析（挖掘中心）。

CRM 必须能与 ERP 很好地集成。

CRM 的实现过程

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。

客户群维系，即通过与顾客的交流实现新的销售。

商机管理，即利用数据库开展销售。

11.2 供应链管理 SCM

SCM 是一种集成的管理思想和方法，它执行供应链中从供应商到最终用户的物流的计划和控制等职能。从单一的企业角度来看，是指企业通过改善上下游供应链关系，整合和优化供应链中的信息流、物流和资金流，以获得企业的竞争优势。

供应链节点：供应商、制造商、分销商、零售商、仓库、配送中心、客户等。

每个企业内部都有一条或者几条供应链，每个企业也都处于一条供应链的某个节点中，SCM 要注意的就是供应网络的构造，处于供应链中的企业是利益共同体，如何实现协作运营。SCM 包括计划、采购、制造、配送、退货五大基本内容。

计划：这是 SCM 的策略部分，企业需要有一个策略来管理所有的资源，以满足客户对产品的需求。好的计划是建立一系列的方法监控供应链。

采购：选择能为企业提供产品和服务的供应商。

制造：安排生产、测试、打包和准备送货所需的活动，是供应链中测量内容最多的部分。

配送：即物流，是调整用户的订单收据、建立仓库网络、拍递送人员提货并送货、建立产品计价系统、接收付款。

退货：是供应链中的问题处理部分。

SCM 的设计原则

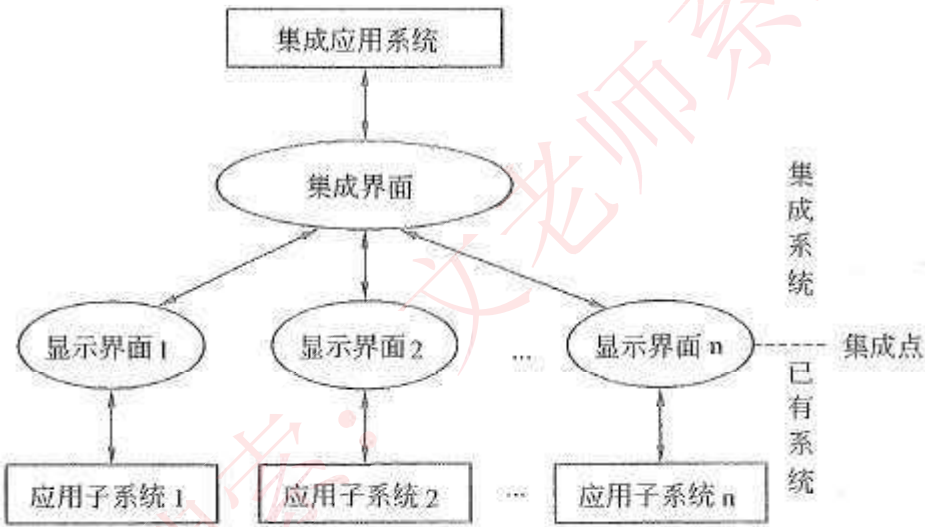
自顶向下和自底向上结合、简洁性、互补性、协调性、动态性、创新性、战略性原则。

11.3 企业应用集成 EAI

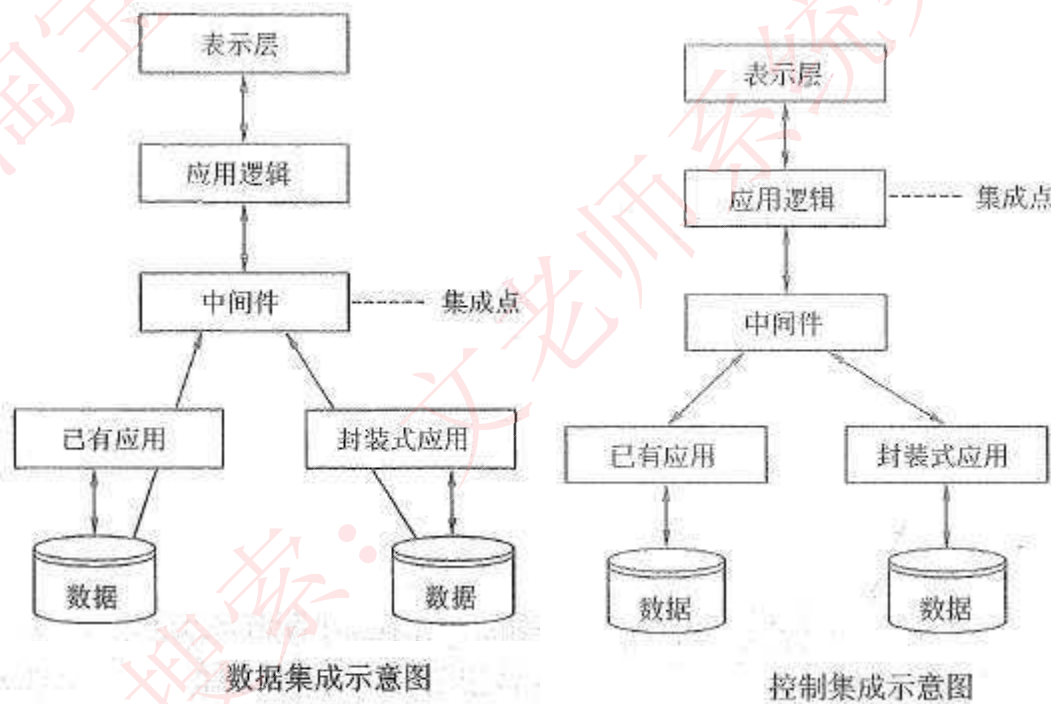
企业信息系统中容易产生信息孤岛，即在一个企业内部有多个子系统，且不互通，一个信息需要反复多次输入，为了解决信息孤岛的问题，提出了企业应用集成 EAI 的概念，可以将企业的多个信息系统联系起来，让使用者感觉在使用一个系统。企业的价值取向是推动 EAI 技术发展的原动力，而 EAI 的实现反过来也驱动企业竞争优势的提升。

企业集成分类

(1) 表示集成：即界面集成，是最原始的集成，黑盒集成。将多个信息系统的界面集成在一起，统一入口，为用户提供一个看上去统一，但是由多个系统组成的应用系统的集成，例如桌面。



(2) 数据集成：白盒集成，把不同来源、格式、特点性质的数据在逻辑上或者物理上有机的集中，从而为企业提供全面的数据共享。如数据仓库。



(3) 控制集成(功能集成、应用集成): 黑盒集成, 业务逻辑层次的集成, 可以借助于远程过程调用或远程方法调用、面向消息的中间件等技术, 将多个应用系统功能进行绑定, 使之像一个实时运行的系统一样接受信息输入和产生数据输出, 实现多个系统功能的叠加。如钉钉。

(4) 业务流程集成: 即过程集成, 最彻底的、综合的集成, 这种集成超越了数据和系统, 由一系列基于标准的、统一数据格式的工作流组成。当进行业务流程集成时, 企业必须对各种业务信息的交换进行定义、授权和管理, 以便于改进操作、减少成本、提高响应速度。它包括应用集成、B2B 集成、自动化业务流程管理、人工流程管理、企业门户, 以及对所有应用系统和流程的管理和监控等。如电子购物网站-第三方支付平台-银行-物流等流程集成。

应用集成数据交换方式

共享数据库: 在应用集成时, 让多个应用系统通过直接共享数据库的方式, 来进行数据交换, 实时性强, 可以频繁交互, 属于同步方式; 但是安全性、并发控制、死锁等问题突出。

消息传递: 消息是软件对象之间进行交互和通信时所使用的一种数据结构, 可以独立于软件平台而存在, 适用于数据量小、但要求频繁、立即、可靠、异步地数据交换场合。

文件传输: 是指在进行数据交换时, 直接将数据文件传送到相应位置, 让目标系统直接读取数据, 可以一次性传送大量信息, 但不适合频繁进行数据传送。适用于数据量大、交换频度小、即时性要求低的情况。

企业集成平台

集成平台是支持企业集成的支撑环境, 包括硬件、软件、软件工具和系统, 通过集成各种企业应用 软件形成企业集成系统。由于硬件环境和应用程序的多样性, 企业信息系统的功能和环境都非常复杂, 因此, 为了能够较好地满足企业的应用需求, 作为企业集成系统支持环境的集成平台, 其基本 功能主要有:

(1) 通信服务 它提供分布环境下透明的同步 / 异步通信服务功能, 使用户和应用程序无需关心具体的操作系统和 应用程序所处的网络物理位置, 而以透明的函数调用或对象服务方式完成它们所需的通信服务要 求。

(2) 信息集成服务 它为应用提供透明的信息访问服务, 通过实现异种数据库系统之间数据的交换、互操作、分布数据 据管理和共享信息模型定义(或共享信息数据库的建立), 使集成平台上运行的应用、服务或用户 端能够以一致的语义和接口实现对数据(数据库、数据文件、应用交互信息)的访问与控制。

(3) 应用集成服务 它通过高层应用编程接口来实现对相应应用程序的访问, 这些高层应用编程接口包含在不同的适配 器或代理中, 它们被用来连接不同的应用程序。这些接口以函数或对象服务的方式向平台的组件模 型提供信息, 使用户在无需对原有系统进行修改(不会影响原有系统的功能)的情况下, 只要在原 有系统的基础上加上相应的访问接口就可以将现有的、用不同的技术实现的系统互联起来、通过为 应用提供数据交换和访问操作, 使各种不同的系统能够相互协作。

(4) 二次开发工具 二次开发工具是集成平台提供的一组帮助用户开发特定应用程序(如实现数据转换的适配器或应用 封装服务等)的支持工具, 其目的是简化用户在企业集成平台实施过程中(特定应用程序接口)的 开发工作。

(5) 平台运行管理工具 它是企业集成平台的运行管理和控制模块，负责企业集成平台系统的静态和动态配置、集成平台应用运行管理和维护、事件管理和出错管理等。通过命名服务、目录服务、平台的动态静态配置，以及其中的关键数据的定期备份等功能来维护整个服务平台的系统配置及稳定运行。

11.4 电子商务

电子商务分三个方面：电子商情广告、电子选购和交易；电子交易凭证的交换、电子支付与结算；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、制造商和储运商）、银行（包括发卡行和收单行）及认证中心。

按照交易对象，电子商务模式包括：

- (1) B2B 模式，企业对企业。
- (2) B2C 模式，企业对消费者。
- (3) C2C，消费者对消费者。
- (4) O2O 即 Online To Offline，含义是线上购买线下的商品和服务，实体店提货或享受服务。

第三方平台结算支付模式有如下优点：

- (1) 比较安全，信用卡信息或帐户信息仅需要告知支付中介，而无需告诉每一个收款人，大大减少了信用卡信息和账户信息失密的风险；
- (2) 支付成本较低，支付中介集中了大量的电子小额交易，形成规模效应，因而支付成本较低；
- (3) 使用方便。对支付者而言，他所面对的是友好的界面，不必考虑背后复杂的技术操作过程；
- (4) 支付担保业务可以在很大程度上保障付款人的利益。

目前，我国**电子商务标准体系**包含 4 方面，分别是基础技术标准、业务标准、支撑体系标准和监督管理标准。其中，服务质量属于监督管理标准范畴；注册维护属于业务标准范畴；在线支付属于支撑体系标准范畴；信息分类编码属于业务标准范畴。

九 系统安全

1 信息安全基础知识

信息安全包括 5 个基本要素：机密性、完整性、可用性、可控性与可审查性。

- (1) 机密性：确保信息不暴露给未授权的实体或进程。
- (2) 完整性：只有得到允许的人才能修改数据，并且能够判别出数据是否已被篡改。
- (3) 可用性：得到授权的实体在需要时可访问数据，即攻击者不能占用所有的资源而阻碍授权者的工作。
- (4) 可控性：可以控制授权范围内的信息流向及行为方式。

(5) 可审查性：对出现的信息安全问题提供调查的依据和手段。

信息安全的范围包括：设备安全、数据安全、内容安全和行为安全。

(1) 信息系统设备的安全是信息系统安全的首要问题，是信息系统安全的物质基础，它包括 3 个方面：设备的稳定性、可靠性、可用性。

(2) 数据安全即采取措施确保数据免受未授权的泄露、篡改和毁坏，包括 3 个方面：数据的秘密性、完整性、可用性。

(3) 内容安全是信息安全在政治、法律、道德层次上的要求，包括 3 个方面：信息内容政治上健康、符合国家法律法规、符合道德规范。

(4) 信息系统的服务功能是指最终通过行为提供给用户，确保信息系统的行为安全，才能最终确保系统的信息安全。行为安全的特性包括：行为的秘密性、完整性、可控性。

信息的存储安全包括信息使用的安全、系统安全监控、计算机病毒防治、数据的加密和防止非法的攻击等。

(1) 信息使用的安全。包括用户的标识与验证、用户存取权限限制。

(2) 系统安全监控。系统必须建立一套安全监控系统，全面监控系统的活动，并随时检查系统的使用情况，一旦有非法入侵者进入系统，能及时发现并采取相应措施，确定和填补安全及保密的漏洞。还应当建立完善的审计系统和日志管理系统，利用日志和审计功能对系统进行安全监控。

(3) 计算机网络服务器必须加装网络病毒自动检测系统，以保护网络系统的安全，防范计算机病毒的侵袭，并且必须定期更新网络病毒检测系统。

网络安全

网络安全隐患体现在：物理安全性、软件安全漏洞、不兼容使用安全漏洞、选择合适的安全哲理。

网络安全威胁：非授权的访问、信息泄露或丢失、破坏数据完整性、拒绝服务攻击、利用网络传播病毒。

安全措施的目标：访问控制、认证、完整性、审计、保密。

2 信息安全系统的组成框架

技术体系：从实现技术上来看，信息安全系统涉及基础安全设备、计算机网络安全、操作系统安全、数据库安全、终端设备安全等多方面技术。

(1) 基础安全设备包括密码芯片、加密卡、身份识别卡等，此外还涵盖运用到物理安全的物理环境保障技术，建筑物、机房条件及硬件设备条件满足信息系统的机械防护安全，通过对电力供应设备以及信息系统组件的抗电磁干扰和电磁泄漏性能的选择性措施达到相应的安全目的。

(2) 计算机网络安全指信息在网络传输过程中的安全防范，用于防止和监控未经授权破坏、更改和盗取数据的行为。通常涉及物理隔离，防火墙及访问控制，加密传输、认证、数字签名、摘要，隧道及 VPN 技术，病毒防范及上网行为管理，安全审计等实现技术。

(3) 操作系统安全是指操作系统的无错误配置、无漏洞、无后门、无特洛伊木马等，能防止非法用户对计算机资源的非法存取，一般用来表达对操作系统的安全需求。操作系统的安全机制包括标

识与鉴别机制、访问控制机制、最小特权管理、可信通路机制、运行保障机制、存储保护机制、文件保护机制、安全审计机制，等等。

(4) 数据库安全可粗略划分为数据库管理系统安全和数据库应用系统安全两个部分，主要涉及物理数据库的完整性、逻辑数据库的完整性、元素安全性、可审计性、访问控制、身份认证、可用性、推理控制、多级保护以及消除隐通道等相关技术。

(5) 终端安全设备从电信网终端设备的角度分为电话密码机、传真密码机、异步数据密码机等。

3 信息安全技术

对称加密技术

就是对数据的加密和解密的密钥（密码）是相同的，属于不公开密钥加密算法。其缺点是加密强度不高（因为只有一个密钥），且密钥分发困难（因为密钥还需要传输给接收方，也要考虑保密性等问题）。

常见的对称密钥加密算法如下：

DES：替换+移位、56 位密钥、64 位数据块、速度快，密钥易产生。

3DES：三重 DES，两个 56 位密钥 K1、K2。

加密：K1 加密->K2 解密->K1 加密。

解密：K1 解密->K2 加密->K1 解密

AES：是美国联邦政府采用的一种区块加密标准，这个标准用来替代原先的 DES。对其的要求是“至少像 3DES 一样安全”。

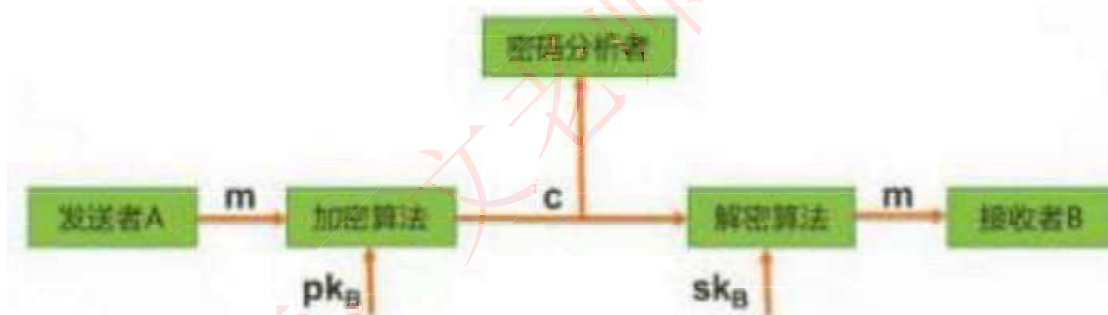
RC-5：RSA 数据安全公司的很多产品都使用了 RC-5。

IDEA：128 位密钥，64 位数据块，比 DES 的加密性好，对计算机功能要求相对低。

非对称加密技术

就是对数据的加密和解密的密钥是不同的，是公开密钥加密算法。其缺点是加密速度慢。

非对称技术的原理是：发送者发送数据时，使用接收者的公钥作加密密钥，私钥作解密密钥，这样只有接收者才能解密密文得到明文。安全性更高，因为无需传输密钥。但无法保证完整性。如下：



常见的非对称加密算法如下：

RSA：512 位（或 1024 位）密钥，计算量极大，难破解。

Elgamal、ECC（椭圆曲线算法）、背包算法、Rabin、D-H 等。

相比较可知，对称加密算法密钥一般只有 56 位，因此加密过程简单，适合加密大数据，也因此加密强度不高；而非对称加密算法密钥有 1024 位，相应的解密计算量庞大，难以破解，却不适合加密大数据，一般用来加密对称算法的密钥，这样，就将两个技术组合使用了，这也是数字信封的原理：

数字信封原理：信是对称加密的密钥，数字信封就是对此密钥进行非对称加密，具体过程：发送方将数据用对称密钥加密传输，而将对称密钥用接收方公钥加密发送给对方。接收方收到数字信封，用自己的私钥解密信封，取出对称密钥解密得原文。

数字信封运用了对称加密技术和非对称加密技术，本质是使用对称密钥加密数据，非对称密钥加密对称密钥，解决了对称密钥的传输问题。

信息摘要

所谓信息摘要，就是一段数据的特征信息，当数据发生了改变，信息摘要也会发生改变，发送方会将数据和信息摘要一起传给接收方，接收方会根据接收到的数据重新生成一个信息摘要，若此摘要和接收到的摘要相同，则说明数据正确。**信息摘要是由哈希函数生成的。**

信息摘要的特点：不算数据多长，都会产生固定长度的信息摘要；任何不同的输入数据，都会产生不同的信息摘要；单向性，即只能由数据生成信息摘要，不能由信息摘要还原数据。

信息摘要算法：MD5（产生 128 位的输出）、SHA-1（安全散列算法，产生 160 位的输出，安全性更高）。

数字签名：唯一标识一个发送方。

发送者发送数据时，使用发送者的私钥进行加密，接收者收到数据后，只能使用发送者的公钥进行解密，这样就能唯一确定发送方，这也是数字签名的过程。但无法保证机密性。如下：



公钥基础设施 PKI：是以不对称密钥加密技术为基础，以数据机密性、完整性、身份认证和行为不可抵赖性为安全目的，来实施和提供安全服务的具有普适性的安全基础设施。

(1) 数字证书：一个数据结构，是一种由一个可信任的权威机构签署的信息集合。在不同的应用中有不同的证书。如 X.509 证书必须包含下列信息：(1) 版本号 (2) 序列号 (3) 签名算法标识符 (4) 认证机构 (5) 有效期限 (6) 主题信息 (7) 认证机构的数字签名 (8) 公钥信息。

公钥证书主要用于确保公钥及其与用户绑定关系的安全。这个公钥就是证书所标识的那个主体的合法的公钥。任何一个用户只要知道签证机构的公钥，就能检查对证书的签名的合法性。如果检查正确，那么用户就可以相信那个证书所携带的公钥是真实的，而且这个公钥就是证书所标识的那个主体的合法的公钥。例如驾照。

(2) 签证机构 **CA**：负责签发证书、管理和撤销证书。是所有注册用户所信赖的权威机构，**CA** 在给用户签发证书时要加上自己的数字签名，以保证证书信息的真实性。任何机构可以用 **CA** 的公钥来验证该证书的合法性。

访问控制是指主体依据某些控制策略或权限对客体本身或是其资源进行的不同授权访问。访问控制包括 3 个要素，即主体（对其他实体施加动作）、客体（接受其他实体访问）和控制策略（主体对客体的操作行为集和约束条件集）。

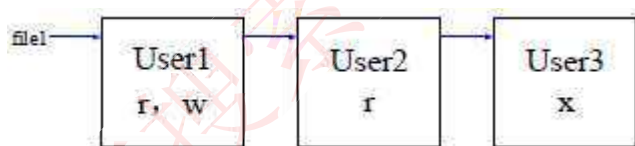
访问控制的实现首先要考虑对合法用户进行验证，然后是对控制策略的选用与管理，最后要对没有非法用户或是越权操作进行管理。所以，访问控制包括**认证、控制策略实现和审计** 3 方面的内容。

访问控制的实现技术

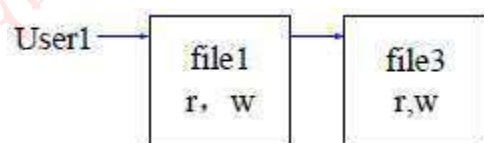
(1) 访问控制矩阵 (**ACM**)。是通过矩阵形式表示访问控制规则和授权用户权限的方法。也就是说，对每个主体而言，都拥有对哪些客体的哪些访问权限；而对客体而言，又有哪些主体对他可以实施访问；将这种关联关系加以阐述，就形成了控制矩阵。主体作为行，客体作为列。

	file1	file2	file3
User1	rw		rw
User2	r	rwX	X
User3	X	r	

(2) 访问控制表 (**ACL**)。目前最流行、使用最多的访问控制实现技术。每个客体有一个访问控制表，是系统中每一个有权访问这个客体的主体的信息。这种实现技术实际上是按列保存访问矩阵。



(3) 能力表。对应于访问控制表，这种实现技术实际上是按行保存访问矩阵。每个主体有一个能力表，是该主体对系统中每一个客体的访问权限信息。使用能力表实现的访问控制系统可以很方便地查询某一个主体的所有访问权限。



(4) 授权关系表。每一行（或者说元组）就是访问矩阵中的一个非空元素，是某一个主体对应于某一个客体的访问权限信息。如果授权关系表按主体排序，查询时就可以得到能力表的效率；如果按客体排序，查询时就可以得到访问控制表的效率。

4 信息安全的抗攻击技术

为对抗攻击者的攻击，密钥生成需要考虑 3 个方面的因素：增大密钥空间、选择强钥（复杂的）、密钥的随机性（使用随机数）。

拒绝服务攻击有许多种，网络的内外部用户都可以发动这种攻击。内部用户可以通过长时间占用系统的内存、CPU 处理时间使其他用户不能及时得到这些资源，而引起拒绝服务攻击；外部黑客也可以通过占用网络连接使其他用户得不到网络服务。本节主要讨论外部用户实施的拒绝服务攻击。

外部用户针对网络连接发动拒绝服务攻击主要有以下几种模式：消耗资源、破坏或更改配置信息、物理破坏或改变网络部件、利用服务程序中的处理错误使服务失效。

分布式拒绝服务 DDoS 攻击是对传统 DoS 攻击的发展，攻击者首先侵入并控制一些计算机，然后控制这些计算机同时向一个特定的目标发起拒绝服务攻击。克服了传统 DOS 受网络资源的限制和隐蔽性两大缺点。

拒绝服务攻击的防御方式

(1) 加强对数据包的特征识别，攻击者在传达攻击命令或发送攻击数据时，虽然都加入了伪装甚至加密，但是其数据包中还是有一些特征字符串。通过搜寻这些特征字符串，就可以确定攻击服务器和攻击者的位置。

(2) 设置防火墙监视本地主机端口的使用情况。如果发现端口处于监听状态，则系统很可能受到攻击。即使攻击者已经对端口的位置进行了一定的修改，但如果外部主机主动向网络内部高标号端口发起连接请求，则系统也很可能受到侵入。

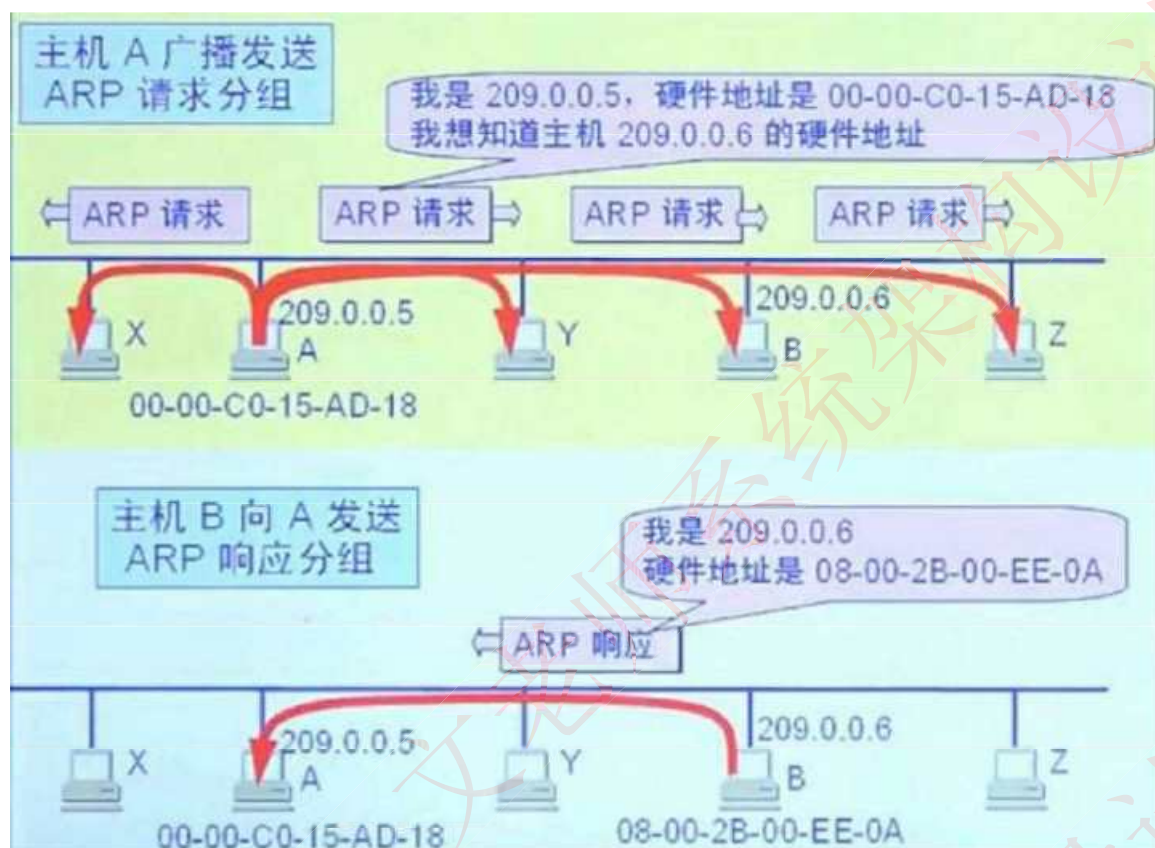
(3) 对通信数据量进行统计也可获得有关攻击系统的位置和数量信息。例如，在攻击之前，目标网络的域名服务器往往会接收到远远超过正常数量的反向和正向的地址查询。在攻击时，攻击数据的来源地址会发出超出正常极限的数据量。

(4) 尽可能的修正已经发现的问题和系统漏洞。

ARP 欺骗

正常 ARP 原理：如下图所示，主机 A 想知道局域网内主机 B 的 MAC 地址，那么主机 A 就广播发送 ARP 请求分组，局域网内主机都会收到，但只有 B 收到解析后知道是请求自己的 MAC 地址，所以只有 B 会返回单播的响应分组，告诉 A 自己的 MAC 地址。

A 收到响应分组后，会建立一个 B 的 IP 地址和 MAC 地址映射，这个映射是动态存在的，如果一定时间 AB 不再通信，那么就会清空这个地址映射，下次如果还要通信，则重复这个过程。



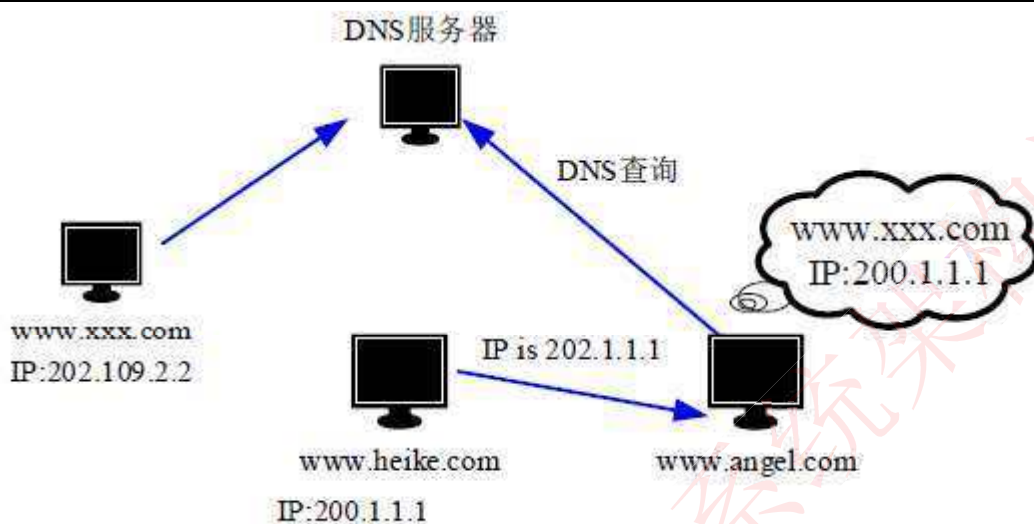
ARP 欺骗原理：上述过程主机 A 是不管其有没有发送过请求广播分组的，而是只要收到了返回的分组信息，就会刷新 IP 地址和 MAC 地址的映射关系，这样就存在安全隐患，假设有主机 C，模拟返回分组格式，构造正确的 IP 地址和自己的 MAC 地址映射，A 收到后也会刷新映射关系，那么当 A 再次向 B 发送信息时，实际就发送到了 C 的 MAC 地址，数据就被 C 监听到了。

ARP 欺骗的防范措施：

- ①在 winxp 下输入命令：`arp-s gate-way-ip gate-way-mac` 固化 arp 表，阻止 arp 欺骗。
- ②使用 ARP 服务器。通过该服务器查找自己的 ARP 转换表来响应其他机器的 ARP 广播。确保这台 ARP 服务器不被黑。
- ③采用双向绑定的方法解决并且防止 ARP 欺骗。
- ④ARP 防护软件——ARPGuard。通过系统底层核心驱动，无须安装其他任何第三方软件（如 WinPcap），以服务及进程并存的形式随系统启动并运行，不占用计算机系统资源。无需对计算机进行 IP 地址及 MAC 地址绑定，从而避免了大量且无效的工作量。也不用担心计算机会在重启后新建 ARP 缓存列表，因为此软件是以服务与进程相结合的形式存在于计算机中，当计算机重启后软件的防护功能也会随操作系统自动启动并工作。

DNS 欺骗

DNS 欺骗首先是冒充域名服务器，然后把查询的 IP 地址设为攻击者的 IP 地址，这样的话，用户上网就只能看到攻击者的主页，而不是用户想要取得的网站的主页了，这就是 DNS 欺骗的基本原理。也即改掉了域名和 IP 地址的对应关系。黑客是通过冒充 DNS 服务器回复查询 IP 的，如下图所示：



DNS 欺骗的检测:

①被动监听检测: 通过旁路监听的方式, 捕获所有 DNS 请求和应答数据包, 并为其建立一个请求应答映射表。如果在一定的时间间隔内, 一个请求对应两个或两个以上结果不同的应答包, 则怀疑受到了 DNS 欺骗攻击。

②虚假报文探测: 采用主动发送探测包的手段来检测网络内是否存在 DNS 欺骗攻击者。如果向一个非 DNS 服务器发送请求包, 正常来说不会收到任何应答, 但是由于攻击者不会验证目标 IP 是否是合法 DNS 服务器, 他就会继续实施欺骗攻击, 因此如果收到了应答包, 则说明受到了攻击。

③交叉检查查询: 在客户端收到 DNS 应答包之后, 向 DNS 服务器反向查询应答包中返回的 IP 地址所对应的 DNS 名字, 如果二者一致说明没有受到攻击, 否则说明被欺骗。

IP 欺骗

IP 欺骗的原理和流程:

- ①首先使被冒充主机 host b 的网络暂时瘫痪, 以免对攻击造成干扰;
- ②然后连接到目标机 host a 的某个端口来猜测 ISN 基值和增加规律;
- ③接下来把源地址伪装成被冒充主机 host b, 发送带有 SYN 标志的数据段请求连接;
- ④然后等待目标机 host a 发送 SYN+ACK 包给已经瘫痪的主机, 因为现在看不到这个包;
- ⑤最后再次伪装成主机 host b 向目标主机 host a 发送的 ACK, 此时发送的数据段带有预测的目标机的 ISN+1;
- ⑥连接建立, 发送命令请求。

IP 欺骗的防范: 虽然 IP 欺骗攻击有着相当难度, 但这种攻击非常广泛, 入侵往往由这里开始。预防这种攻击可以删除 UNIX 中所有的/etc/hosts.equiv、\$HOME/.rhosts 文件, 修改/etc/inetd.conf 文件, 使得 RPC 机制无法应用。另外, 还可以通过设置防火墙过滤来自外部而信源地址却是内部 IP 的报文。

端口扫描就是尝试与目标主机的某些端口建立连接, 如果目标主机该端口有回复(见三次握手中的第二次), 则说明该端口开放, 即为“活动端口”。

扫描原理分类:

(1) 全 TCP 连接。这种扫描方法使用三次握手，与目标计算机建立标准的 TCP 连接。需要说明的是，这种古老的扫描方法很容易被目标主机记录。

(2) 半打开式扫描（SYN 扫描）。在这种扫描技术中，扫描主机自动向目标计算机的指定端口发送 SYN 数据段，表示发送建立连接请求。

- 如果目标计算机的回应 TCP 报文中 SYN=1 ACK=1，则说明该端口是活动的，接着扫描主机传送给目标主机拒绝建立 TCP 连接，从而导致三次握手的过程失败。

- 如果目标计算机的回应为 RST，则表示该端口为“死端口”，这种情况下，扫描主机不用做任何回应。

(3) FIN 扫描。在前面介绍过的 TCP 报文中，有一个字段为 FIN，FIN 扫描则依靠发送 FIN 来判断目标计算机的指定端口是否是活动的。

发送一个 FIN=1 的 TCP 报文到一个关闭的端口时，该报文会被丢掉，并返回一个 RST 报文。

但是，如果当 FIN 报文到一个活动的端口时，该报文只是被简单的丢掉，不会返回任何回应。

从 FIN 扫描可以看出，这种扫描没有涉及任何 TCP 连接部分。因此，这种扫描比前两种都安全，可以称之为秘密扫描。

(4) 第三方扫描。第三方扫描又称“代理扫描”，这种扫描是利用第三方主机来代替入侵者进行扫描。这个第三方主机一般是入侵者通过入侵其他计算机而得到的，该“第三方”主机常被入侵者称之为“肉鸡”。这些“肉鸡”一般为安全防御系数极低的个人计算机。

强化 TCP/IP 堆栈以抵御拒绝服务攻击

1. 同步包风暴(SYN Flooding)：利用 TCP 协议缺陷发送大量伪造的 TCP 连接请求，使得被攻击者资源耗尽。三次握手，进行了两次，不进行第三次握手，连接队列处于等待状态，大量这样的等待，会占满全部队列空间，使得系统挂起。可以通过修改注册表防御 SYN Flooding 攻击。

2. ICMP 攻击。ICMP 协议本身的特点决定了它非常容易被用于攻击网络上的路由器和主机。比如，前面提到的“Ping of Death”攻击就是利用操作系统规定的 ICMP 数据包的最大尺寸不超过 64KB 这一规定，达到使 TCP/IP 堆栈崩溃、主机死机的效果。可以通过修改注册表防御 ICMP 攻击。

3. SNMP 攻击。SNMP 还能被用于控制这些设备和产品，重定向通信流，改变通信数据包的优先级，甚至断开通信连接。总之，入侵者如果具备相应能力，就能完全接管你的网络。可以通过修改注册表项防御。

系统漏洞扫描指对重要计算机信息系统进行检查，发现其中可能被黑客利用的漏洞。包括基于网络的漏洞扫描（通过网络远程扫描主机）、基于主机的漏洞扫描（在目标系统安装了代理扫描）。

5 信息安全的保证体系与评估方法

GB17859—999 标准规定了计算机系统安全保护能力的五个等级：

第一级 用户自主保护级：本级的计算机信息系统可信计算基通过隔离用户与数据，使用户具备自主安全保护的能力。本级实施的是自主访问控制，即计算机信息系统可信计算机定义和控制系统中命名用户对命名客体的访问。

第二级 系统审计保护级：本级的计算机信息系统可信计算基实施了粒度更细的自主访问控制，它通过登录规程、审计安全性相关事件和隔离资源，使用户对自己的行为负责。在自主访问控制的基础上控制访问权限扩散。

第三级 安全标记保护级：本级的计算机信息系统可信计算基具有系统审计保护级所有功能。此外，还提供有关安全策略模型、数据标记以及主体对客体强制访问控制的非形式化描述；具有准确地标记输出信息的能力；消除通过测试发现的任何错误。本级的主要特征是计算机信息系统可信计算基对所有主体及其所控制的客体（例如：进程、文件、段、设备）实施强制访问控制。

第四级 结构化保护级：本级的计算机信息系统可信计算基建立于一个明确定义的形式化安全策略模型之上，它要求将第三级系统中的自主和强制访问控制扩展到所有主体与客体。此外，还要考虑隐蔽通道。对外部主体能够直接或间接访问的所有资源（例如：主体、存储客体和输入输出资源）实施强制访问控制。

第五级 访问验证保护级：本级的计算机信息系统可信计算基满足访问监控器需求。访问监控器仲裁主体对客体的全部访问。访问监控器本身是抗篡改的；必须足够小，能够分析和测试。与第四级相比，自主访问控制机制根据用户指定方式或默认方式，阻止非授权用户访问客体。访问控制的粒度是单个用户。访问控制能够为每个命名客体指定命名用户和用户组，并规定他们对客体的访问模式。没有存取权的用户只允许由授权用户指定对客体的访问权。

安全风险管理

在风险评估实施前，应该考虑：

- （1）确定风险评估的范围。
- （2）确定风险评估的目标。
- （3）建立适当的组织结构。
- （4）建立系统性的风险评估方法。
- （5）获得最高管理者对风险评估策划的批准。

风险评估的基本要素为脆弱性、资产、威胁、风险和安全措施，与这些要素相关的属性分别为业务战略、资产价值、安全需求、安全事件和残余风险，这些也是风险评估要素的一部分。

风险计算模型包含信息资产、弱点 / 脆弱性、威胁等关键要素。每个要素有各自的属性，信息资产的属性是资产价值，弱点的属性是弱点被威胁利用后对资产带来的影响的严重程度，威胁的属性是威胁发生的可能性。风险计算的过程如下。

- （1）对信息资产进行识别，并对资产赋值。
- （2）对威胁进行分析，并对威胁发生的可能性赋值。
- （3）识别信息资产的脆弱性，并对弱点的严重程度赋值。
- （4）根据威胁和脆弱性计算安全事件发生的可能性。
- （5）结合信息资产的重要性和发生安全事件的可能性，计算信息资产的风险值。

6 网络安全技术

防火墙是在内部网络和外部因特网之间增加的一道安全防护措施，分为网络级防火墙和应用级防火墙。

网络级防火墙层次低，但是效率高，因为其使用包过滤和状态监测手段，一般只检验网络包外在（起始地址、状态）属性是否异常，若异常，则过滤掉，不与内网通信，因此对应用和用户是透明的。但是这样的问题是，如果遇到伪装的危险数据包就没办法过滤，此时，就要依靠**应用级防火墙**，层次高，效率低，因为应用级防火墙会将网络包拆开，具体检查里面的数据是否有问题，会消耗大量时间，造成效率低下，但是安全强度高。

入侵检测系统 IDS

防火墙技术主要是分隔来自外网的威胁，却对来自内网的直接攻击无能为力，此时就要用到入侵检测 IDS 技术，位于防火墙之后的第二道屏障，作为防火墙技术的补充。

原理：监控当前系统/用户行为，使用入侵检测分析引擎进行分析，这里包含一个知识库系统，囊括了历史行为、特定行为模式等操作，将当前行为和知识库进行匹配，就能检测出当前行为是否是入侵行为，如果是入侵，则记录证据并上报给系统和防火墙，交由它们处理。

不同于防火墙，IDS 入侵检测系统是一个监听设备，没有跨接在任何链路上，无须网络流量流经它便可以工作。因此，对 IDS 的部署，唯一的要求是：IDS 应当挂接在所有所关注流量都必须流经的链路上。因此，IDS 在交换式网络中的位置一般选择在：（1）尽可能靠近攻击源（2）尽可能靠近受保护资源

入侵防御系统 IPS

IDS 和防火墙技术都是在入侵行为已经发生后所做的检测和分析，而 IPS 是能够提前发现入侵行为，在其还没有进入安全网络之前就防御。串联接入网络，因此可以自动切换网络。

在安全网络之前的链路上挂载入侵防御系统 IPS，可以实时检测入侵行为，并直接进行阻断，这是与 IDS 的区别，要注意。

杀毒软件

用于检测和解决计算机病毒，与防火墙和 IDS 要区分，计算机病毒要靠杀毒软件，防火墙是处理网络上的非法攻击。

蜜罐系统

伪造一个蜜罐网络引诱黑客攻击，蜜罐网络被攻击不影响安全网络，并且可以借此了解黑客攻击的手段和原理，从而对安全系统进行升级和优化。

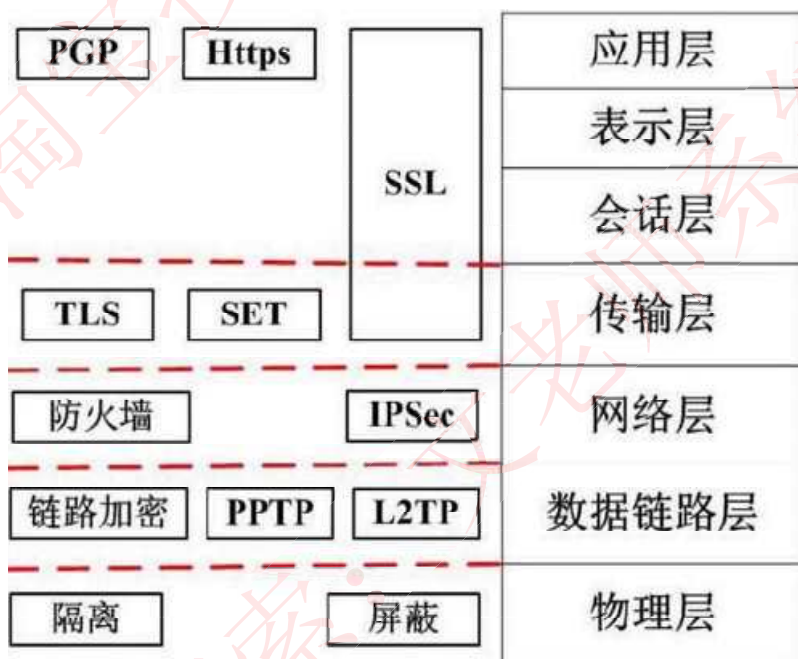
网络攻击和威胁

攻击类型	攻击名称	描述
被动攻击	窃听（网络监听）	用各种可能的合法或非法的手段窃取系统中的信息资源和敏感信息。
	业务流分析	通过对系统进行长期监听，利用统计分析方法对诸如通信频度、通信的信息流向、通信总量的变化等参数进行研究，从而发现有价值的信息和规律。
	非法登录	有些资料将这种方式归为被动攻击方式。

主动攻击	假冒身份	通过欺骗通信系统（或用户）达到非法用户冒充成为合法用户，或者特权小的用户冒充成为特权大的用户的目的。黑客大多是采用假冒进行攻击。
	抵赖	这是一种来自用户的攻击，比如：否认自己曾经发布过的某条消息、伪造一份对方来信等。
	旁路控制	攻击者利用系统的安全缺陷或安全性上的脆弱之处获得非授权的权利或特权。
	重放攻击	所截获的某次合法的通信数据拷贝，出于非法的目的而被重新发送。
	拒绝服务（DOS）	对信息或其它资源的合法访问被无条件地阻止。

7 网络安全协议

物理层主要使用物理手段，隔离、屏蔽物理设备等，其它层都是靠协议来保证传输的安全，具体如下图所示：



◆SSL 协议：安全套接字协议，被设计为加强 Web 安全传输(HTTP/HTTPS/)的协议，安全性高，和 HTTP 结合之后，形成 HTTPS 安全协议，端口号为 443。

◆ **SSH 协议**：安全外壳协议，被设计为加强 Telnet/FTP 安全的传输协议。

◆ **SET 协议**：安全电子交易协议主要应用于 B2C 模式（电子商务）中保障支付信息的安全性。SET 协议本身比较复杂，设计比较严格，安全性高，它能保证信息传输的机密性、真实性、完整性和不可否认性。SET 协议是 PKI 框架下的一个典型实现，同时也在不断升级和完善，如 SET 2.0 将支持借记卡电子交易。

◆ **Kerberos 协议**：是一种网络身份认证协议，该协议的基础是基于信任第三方，它提供了在开放型网络中进行身份认证的方法，认证实体可以是用户也可以是用户服务。这种认证不依赖宿主机的操作系统或计算机的 IP 地址，不需要保证网络上所有计算机的物理安全性，并且假定数据包在传输中可被随机窃取和篡改。

十 软件工程基础知识

1 软件工程

软件开发生命周期：

软件定义时期：包括可行性研究和详细需求分析过程，任务是确定软件开发工程必须完成的总目标，具体可分成问题定义、可行性研究、需求分析等。

软件开发时期：就是软件的设计与实现，可分成概要设计、详细设计、编码、测试等。

软件运行和维护：就是把软件产品移交给用户使用。

软件系统的文档可以分为用户文档和系统文档两类，用户文档主要描述系统功能和使用方法，并不关系这些功能是怎样实现的；系统文档描述系统设计、实现和测试等各方面的内容。

软件工程过程是指为获得软件产品，在软件工具的支持下由软件工程师完成的一系列软件工程活动，包括以下 4 个方面。

(1) **P (Plan)**——软件规格说明。规定软件的功能及其运行时的限制。

(2) **D (Do)**——软件开发。开发出满足规格说明的软件。

(3) **C (Check)**——软件确认。确认开发的软件能够满足用户的需求。

(4) **A (Action)**——软件演进。软件在运行过程中不断改进以满足客户新的需求。

软件系统工具通常可以按软件过程活动将软件工具分为软件开发工具、软件维护工具、软件管理和软件支持工具。

软件开发工具：需求分析工具、设计工具、编码与排错工具、测试工具等。

软件维护工具：版本控制工具、文档分析工具、开发信息库工具、逆向工程工具、再工程工具。

软件管理和软件支持工具：项目管理工具、配置管理工具、软件评价工具、软件开发工具的评价和选择。

软件设计包括四个既独立又相互联系的活动，即数据设计、架构（体系结构）设计、人机界面（接口）设计和过程设计，这四个活动完成以后就得到了全面的软件设计模型。

1.1 软件过程模型

瀑布模型(SDLC): 结构化方法中的模型，是结构化的开发，开发流程如同瀑布一般，一步一步的走下去，直到最后完成项目开发，**只适用于需求明确或者二次开发**（需求稳定），当需求不明确时，最终开发的项目会错误，有很大的缺陷。

原型: 与瀑布模型相反，原型针对的就是**需求不明确**的情况，首先快速构造一个功能模型，演示给用户看，并按用户要求及时修改，中间再通过不断的演示与用户沟通，最终设计出项目，就不会出现与用户要求不符合的情况，采用的是迭代的思想。不适合超大项目开发。

螺旋模型: 是多种模型的混合，针对需求不明确的项目，与原型类似，但是**增加了风险分析**，这也是其最大的特点。适合大型项目开发。四阶段迭代：目标设定、风险分析、开发和有效性验证、评审。

增量模型: 首先开发核心模块功能，而后与用户确认，之后再开发次核心模块的功能，即每次开发一部分功能，并与用户需求确认，最终完成项目开发，优先级最高的服务最先交付。由于并不是从系统整体角度规划各个模块，因此不利于模块划分。难点在于如何将客户需求划分为多个增量。与原型不同的是增量模型的每一次增量版本都可作为独立可操作的作品，而原型的构造一般是为了演示。

喷泉模型: 特点是**面向对象的模型**，而上述其他的模型都是结构化的模型，使用了迭代思想和无间隙开发。

基于构件的开发模型 CBSD: 特点是**增强了复用性**，在系统开发过程中，会构建一个**构件库**，供其他系统复用，因此可以提高可靠性，节省时间和成本。

形式化方法模型: 建立在严格**数学基础**上的一种软件开发方法，主要活动是生成计算机软件形式化的数学规格说明。

敏捷模型

开发宣言：个体和交互 胜过 过程和工具、可以工作的软件 胜过 面面俱到的文档、客户合作 胜过 合同谈判、响应变化 胜过 遵循计划。



敏捷方法区别于其他方法的两个特点：

- (1) 是“适应性”而非“预设性”。
- (2) 是“面向人的”而非“面向过程的”。

敏捷方法的核心思想：

- (1) 敏捷方法是适应型，而非可预测型。拥抱变化，适应变化。
- (2) 敏捷方法是以人为本，而非以过程为本。发挥人的特性。
- (3) 迭代增量式的开发过程。以原型开发思想为基础，采用迭代增量式开发，发行版本小型化。

主要敏捷方法：

(1) 极限编程 (XP)。基础和价值观是交流、朴素、反馈和勇气，即任何一个软件项目都可以从 4 个方面入手进行改善：加强交流：从简单做起：寻求反馈；勇于实事求是。

XP 是一种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非常清楚开发进度、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。

XP 提倡测试先行，为了将以后出现 bug 的几率降到最低。

(2) 水晶系列方法。与 XP 方法一样，都有以人为中心的理念，但在实践上有所不同。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。

(3) 并列争球法 (Scrum)。是一种迭代的增量化过程，把每段时间（如 30 天）一次的迭代称为一个“冲刺” (Sprint)，并按需求的优先级别来实现产品，多个自组织和自治的小组并行地递增实现产品。

(4) 特性驱动开发方法 (FDD)。是一个迭代的开发模型。认为有效的软件开发需要 3 个要素：人、过程和技术。有 5 个核心过程：开发整体对象模型、构造特征列表、计划特征开发、特征设计和

特征构建。其中，计划特征开发根据构造出的特征列表、特征间的依赖关系进行计划，设计出包含特征设计和特征构建过程组成的多次迭代。

统一过程模型（RUP）

RUP 描述了如何有效地利用商业的、可靠的方法开发和部署软件，是一种重量级过程。RUP 类似一个在线的指导者，它可以为所有方面和层次的程序开发提供指导方针、模版以及事例支持。

RUP 软件开发生命周期是一个二维的软件开发模型，RUP 中有 9 个核心工作流，这 9 个核心工作流如下。

- 业务建模：理解待开发系统所在的机构及其商业运作，确保所有参与人员对待开发系统所在的机构有共同的认识，评估待开发系统对所在机构的影响。
- 需求：定义系统功能及用户界面，使客户知道系统的功能，使开发人员理解系统的需求，为项目预算及计划提供基础。
- 分析与设计：把需求分析的结果转化为分析与设计模型。
- 实现：把设计模型转换为实现结果，对开发的代码做单元测试，将不同实现人员开发的模块集成成为可执行系统。
- 测试：检查各子系统之间的交互、集成，验证所有需求是否均被正确实现，对发现的软件质量上的缺陷进行归档，对软件质量提出改进建议。
- 部署：打包、分发、安装软件，升级旧系统；培训用户及销售人员的，并提供技术支持。
- 配置与变更管理：跟踪并维护系统开发过程中产生的所有制品的完整性和一致性。
- 项目管理：为软件开发项目提供计划、人员分配、执行、监控等方面的指导，为风险管理提供框架。
- 环境：为软件开发机构提供软件开发环境，即提供过程管理和工具的支持。

RUP 把软件开发生命周期划分为多个循环，每个循环生成产品的一个新的版本，每个循环依次由 4 个连续的阶段组成，每个阶段完成确定的任务。这 4 个阶段如下。

- 初始阶段：定义最终产品视图和业务模型，并确定系统范围。
- 细化阶段：设计及确定系统的体系结构，制订工作计划及资源要求。
- 构造阶段：构造产品并继续演进需求、体系结构、计划直至产品提交。
- 移交阶段：把产品提交给用户使用。

RUP 中定义了如下一些核心概念，理解这些概念对于理解 RUP 很有帮助。

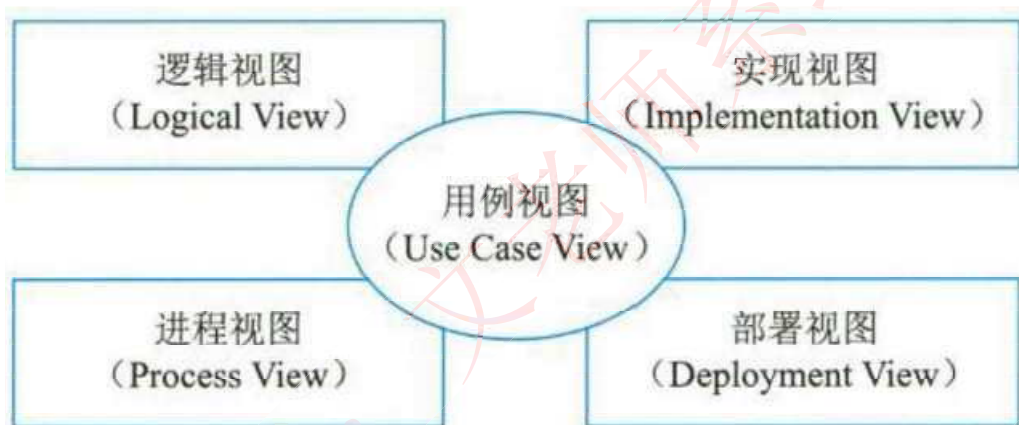
- 角色: Who 的问题。角色描述某个人或一个小组的行为与职责。RUP 预先定义了很多角色，如体系结构师、设计人员、实现人员、测试员和配置管理人员等，并对每一个角色的工作和职责都做了详尽的说明。
- 活动: How 的问题。活动是一个有明确目的的独立工作单元。
- 制品: What 的问题。制品是活动生成、创建或修改的一段信息。
- 工作流: When 的问题。工作流描述了一个有意义的连续的活动序列，每个工作流产生一些有价值的产品，并显示了角色之间的关系。

RUP 的特点：

(1) 用例驱动：需求分析、设计、实现和测试等活动都是用例驱动的。

(2) 以体系结构为中心：包括系统的总体组织和全局控制、通信协议、同步、数据存取、给设计元素分配功能、设计元素的组织、物理分布、系统的伸缩性和性能等。软件的体系结构是一个多维的结构，会采用多个视图来描述。

在典型的 4+1 视图模型中，分析人员和测试人员关心的是系统的行为，会侧重于用例视图；最终用户关心的是系统的功能，会侧重于逻辑视图；程序员关心的是系统的配置、装配等问题，会侧重于实现视图；系统集成人员关心的是系统的性能、可伸缩性、吞吐率等问题，会侧重于进程视图；系统工程师关心的是系统的发布、安装、拓扑结构等问题，会侧重于部署视图。



(3) 迭代与增量。把整个项目开发分为多个迭代过程。在每次迭代中，只考虑系统的一部分需求，进行分析、设计、实现、测试和部署等过程；每次迭代是在已完成部分的基础上进行的，每次增加一些新的功能实现，以此进行下去，直至最后项目的完成。

1.2 能力成熟度模型

能力成熟度模型 CMM

对软件组织化阶段的描述，随着软件组织定义、实施、测量、控制和改进其软件过程，软件组织的能力经过这些阶段逐步提高。针对软件研制和测试阶段。分为如下五个级别：

能力等级	特点	关键过程区域
初始级 (Initial)	软件过程的特点是杂乱无章，有时甚至很混乱，几乎没有明确定义的步骤，项目的成功完全依赖个人的努力和英雄式核心人物的作用。	
可重复级 (Repeatable)	建立了基本的项目管理过程和实践来跟踪项目费用、进度和功能特性，有必要的过程准则来重复以前在同类项目中的成功。	软件配置管理、软件质量保证、软件子合同管理、软件项目跟踪与监督、软件项目策划、软件需求管理
已定义级 (Defined)	管理和工程两方面的软件过程已经文档化、标准化，并综合成整个软件开发组织的标准软件过程。	同行评审、组间协调、软件产品工程、集成软件管理、培训大纲、组

	所有项目都采用根据实际情况修改后得到的标准软件过程来发和维护软件。	织过程定义、组织过程集点
已管理级 (Managed)	制定了软件过程和产品质量的详细度量标准。对软件过程和产品质量有定量的理解和控制。	软件质量管理和定量过程管理
优化级 (Optimized)	加强了定量分析,通过来自过程质量反馈和来自新观念、新技术的反馈使过程能不断持续地改进。	过程更改管理、技术改革管理和缺陷预防

能力成熟度模型集成 CMMI

是若干过程模型的综合和改进,不仅仅软件,而是支持多个工程学科和领域的、系统的、一致的过程改进框架,能适应现代工程的特点和需要,能提高过程的质量和工作效率。

CMMI 两种表示方法:

(1) 阶段式模型: 类似于 CMM, 它关注组织的成熟度, 五个成熟度模型如下:

能力等级	特点	关键过程区域
初始级	过程不可预测且缺乏控制	
已管理级	过程为项目服务	需求管理、项目计划、配置管理、项目监督与控制、供应商合同管理、度量和分析、过程和产品质量保证
已定义级	过程为组织服务	需求开发、技术解决方案、产品集成、验证、确认、组织级过程焦点、组织级过程定义、组织级培训、集成项目管理、风险管理、集成化的团队、决策分析和解决方案、组织级集成环境
定量管理	过程已度量和控制	组织过程性能、定量项目管理
优化级	集中于过程改进和优化	组织级改革与实施、因果分析和解决方案

其中 2-5 级对应的过程域如下图所示:

成熟度等级	过程域
已管理级	需求管理、项目计划、配置管理、项目监督与控制、供应商合同管理、度量和分析、过程和产品质量保证
已定义级	需求开发、技术解决方案、产品集成、验证、确认、组织级过程焦点、组织级过程定义、组织级培训、集成项目管理、风险管理、集成化的团队、决策分析和解决方案、组织级集成环境
定量管理级	组织级过程性能、定量项目管理
优化级	组织级改革与实施、因果分析和解决方案

2.连续式模型: 关注每个过程域的能力, 一个组织对不同的过程域可以达到不同的过程域能力等级。

1.3 逆向工程

软件复用是将已有软件的各种有关知识用于建立新的软件，以缩减软件开发和维护的花费。软件复用是提高软件生产力和质量的一种重要技术。早期的软件复用主要是代码级复用，被复用的知识专指程序，后来扩大到包括领域知识、开发经验、设计决定、体系结构、需求、设计、代码和文档等一切有关方面。

逆向工程：软件的逆向工程是分析程序，力图在比源代码更高抽象层次上建立程序的表示过程，逆向工程是设计的恢复过程。逆向工程的四个级别：

实现级：包括程序的抽象语法树、符号表、过程的设计表示。

结构级：包括反映程序分量之间相互依赖关系的信息，例如调用图、结构图、程序和数据结构。

功能级：包括反映程序段功能及程序段之间关系的信息，例如数据和控制流模型。

领域级：包括反映程序分量或程序诸实体与应用领域概念之间对应关系的信息，例如 E-R 模型。其中，领域级抽象级别最高，完备性最低，实现级抽象级别最低，完备性最高。

与逆向工程相关的概念有**重构、设计恢复、再工程和正向工程**。

(1) 重构是指在同一抽象级别上转换系统描述形式。

(2) 设计恢复是指借助工具从已有程序中抽象出有关数据设计、总体结构设计和过程设计等方面的信息。

(3) 再工程是指在逆向工程所获得信息的基础上，修改或重构已有的系统，产生系统的一个新版本。再工程是对现有系统的重新开发过程，包括逆向工程、新需求的考虑过程和正向工程三个步骤。它不仅能从已存在的程序中重新获得设计信息，而且还能使用这些信息来重构现有系统，以改进它的综合质量。在利用再工程重构现有系统的同时，一般会增加新的需求，包括增加新的功能和改善系统的性能。

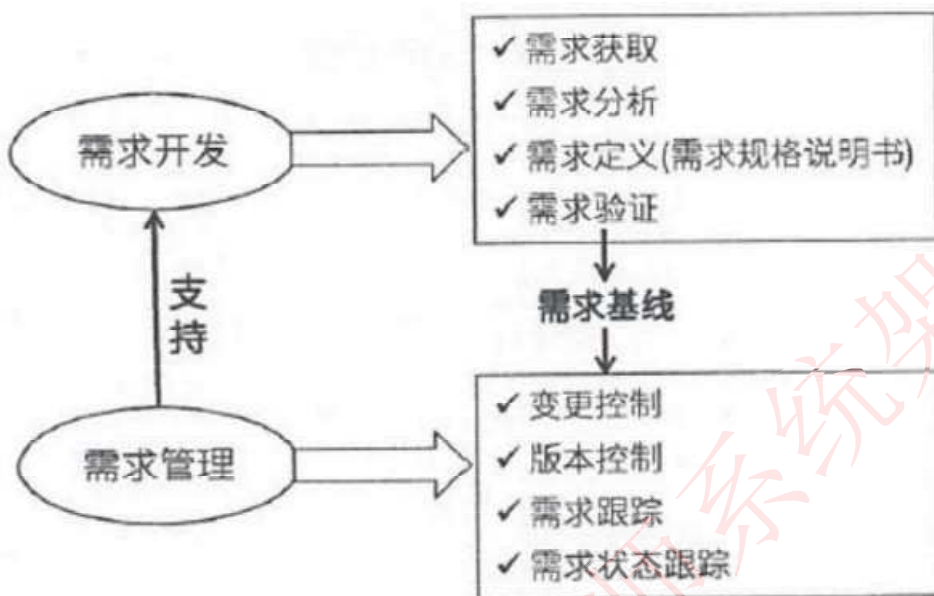
(4) 正向工程是指不仅从现有系统中恢复设计信息，而且使用该信息去改变或重构现有系统，以改善其整体质量。

2 需求工程

2.1 软件需求

是指用户对系统在功能、行为、性能、设计约束等方面的期望。是指用户解决问题或达到目标所需的条件或能力，是系统或系统部件要满足合同、标准、规范或其他正式规定文档所需具有的条件或能力，以及反映这些条件或能力的文档说明

分为需求开发和需求管理两大过程，如下所示：



需求的层次

业务需求：反映企业或客户对系统高层次的目标要求，通常来自项目投资人、客户、市场营销部门或产品策划部门。通过业务需求可以确定项目视图和范围。

用户需求：描述的是用户的具体目标，或用户要求系统必须能完成的任务。即描述了用户能使用系统来做什么。通常采取用户访谈和问卷调查等方式，对用户使用的场景进行整理，从而建立用户需求。

系统需求：从系统的角度来说明软件的需求，包括功能需求、非功能需求和设计约束等。

1) **功能需求：**也称为行为需求，规定了开发人员必须在系统中实现的软件功能，用户利用这些功能来完成任务，满足业务需要。

2) **非功能需求：**指系统必须具备的属性或品质，又可以细分为软件质量属性（如可维护性、可靠性、效率等）和其他非功能需求。

3) **设计约束：**也称为限制条件或补充规约，通常是对系统的一些约束说明，例如必须采用国有自主知识产权的数据库系统，必须运行在 UNIX 操作系统之下等。

2.2 需求获取

是一个确定和理解不同的项目干系人的需求和约束的过程。

常见的需求获取法包括：

- (1) **用户访谈：**1 对 1-3，有代表性的用户。其形式包括结构化和非结构化两种。
- (2) **问卷调查：**用户多，无法一一访谈。
- (3) **采样：**从种群中系统地选出有代表性的样本集的过程。样本数量= $0.25 \times (\text{可信度因子}/\text{错误率})^2$
- (4) **情节串联板：**一系列图片，通过这些图片来讲故事。

(5) 联合需求计划 (JRP): 通过联合各个关键用户代表、系统分析师、开发团队代表一起, 通过有组织的会议来讨论需求。

(6) 需求记录技术: 任务卡片、场景说明、用户故事、Volere 白卡。

2.3 需求分析

一个好的需求应该具有无二义性、完整性、一致性、可测试性、确定性、可跟踪性、正确性、必要性等特性, 因此, 需要分析人员把杂乱无章的用户要求和期望转化为用户需求, 这就是需求分析的工作。

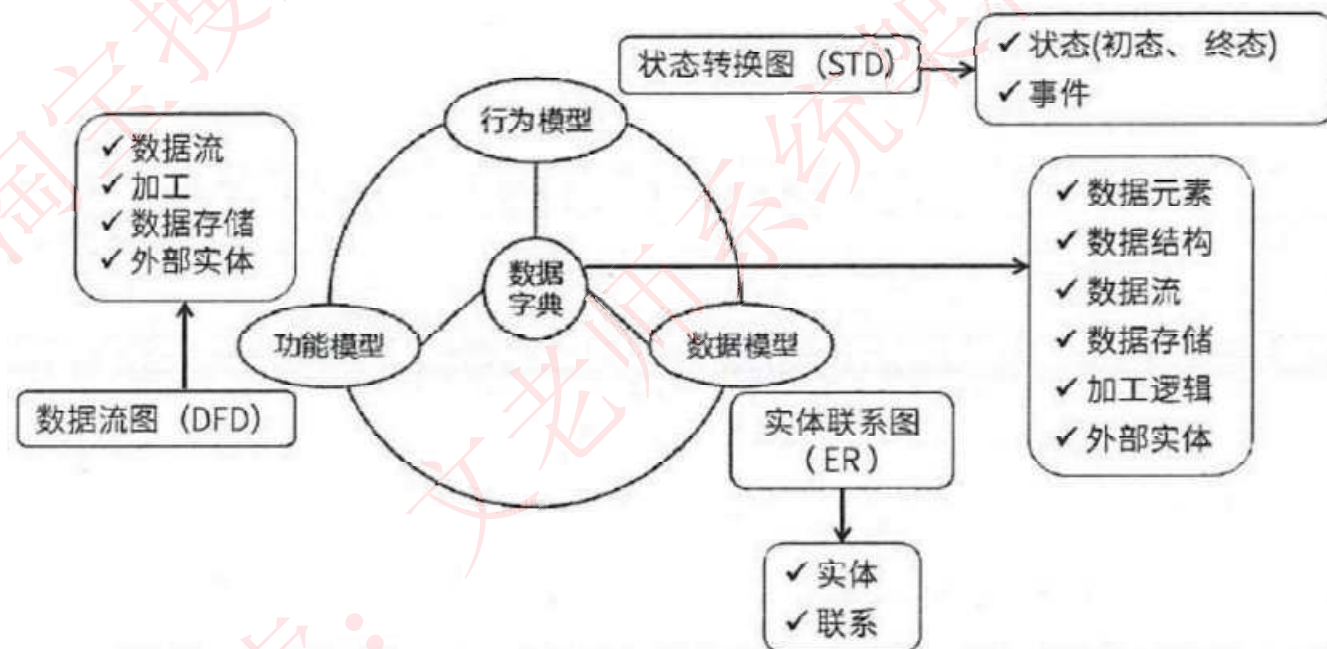
需求分析的任务

- (1) 绘制系统上下文范围关系图
- (2) 创建用户界面原型
- (3) 分析需求的可行性
- (4) 确定需求的优先级
- (5) 为需求建立模型
- (6) 创建数据字典
- (7) 使用 QFD (质量功能部署)

结构化的需求分析

结构化特点: 自顶向下, 逐步分解, 面向数据。

三大模型: 功能模型 (数据流图)、行为模型 (状态转换图)、数据模型 (E-R 图) 以及数据字典。

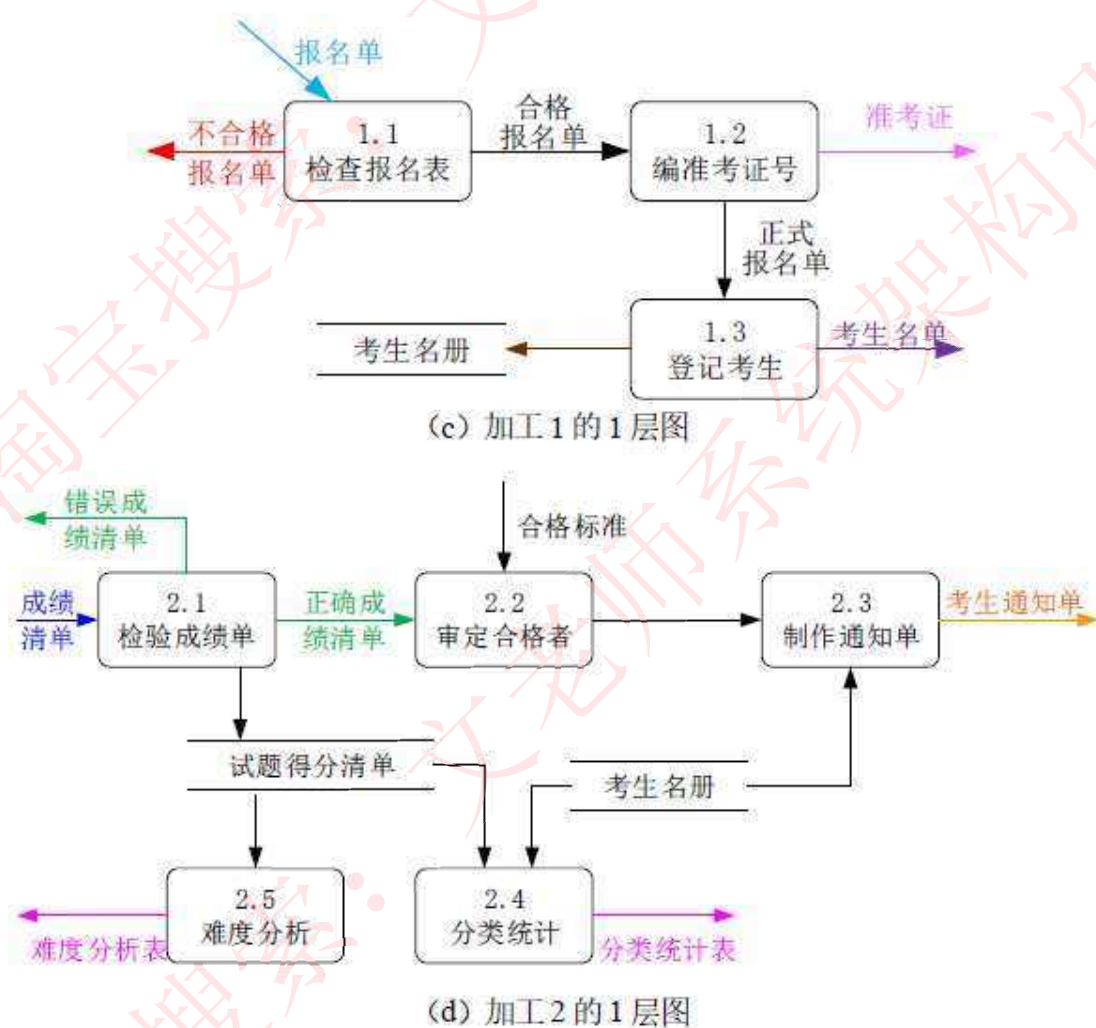
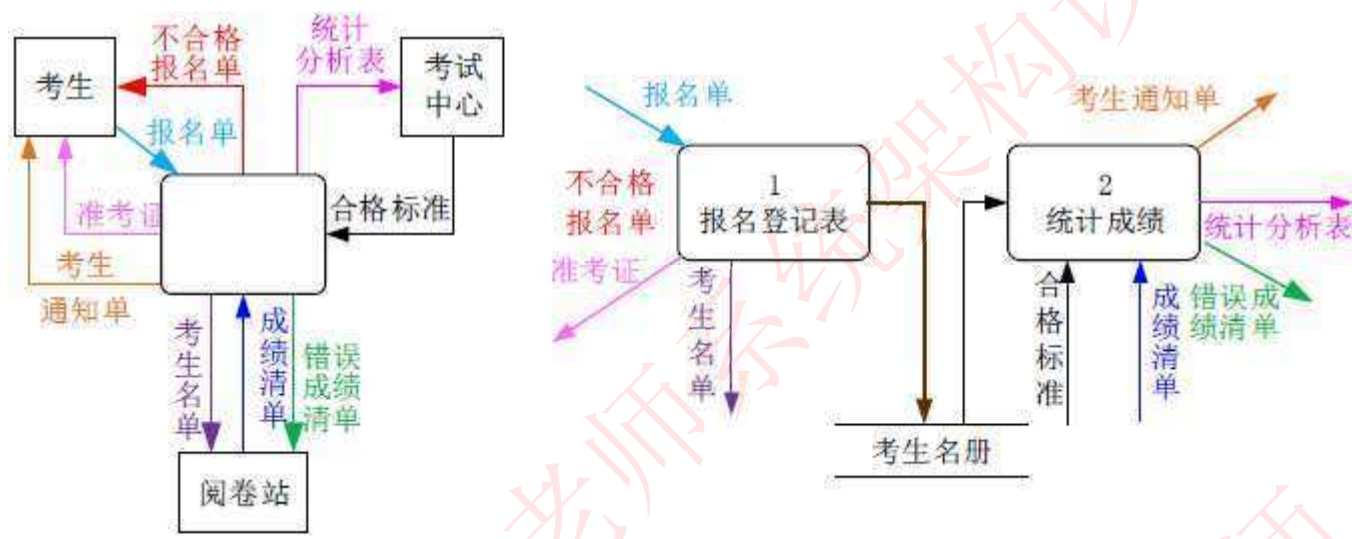


数据流图 DFD

基本图形元素: 外部实体、加工、数据存储、数据流。

4) 外部实体（外部主体）：是指存在于软件系统之外的人员或组织，它指出系统所需数据的发源地（源）和系统所产生的数据的归宿地（宿）。

分层数据流图：



数据字典 DD

数据流图描述了系统的分解，但没有对图中各成分进行说明。数据字典就是为数据流图中的每个数据流、文件、加工，以及组成数据流或文件的数据项做出说明。

数据字典有以下 4 类条目：数据流、数据项、数据存储和基本加工。

符号	含义	举例及说明
=	被定义为	
+	与	$x=a+b$ ，表示 x 由 a 和 b 组成
[... ...]	或	$x=[a b]$ ，表示 x 由 a 或 b 组成
{.....}	重复	$x=\{a\}$ ，表示 x 由 0 个或多个 a 组成

加工逻辑也称为“小说明”。常用的加工逻辑描述方法有结构化语言、判定表和判定树 3 种。

2.4 需求定义

（软件需求规格说明书 SRS）：是需求开发活动的产物，编制该文档的目的是使项目干系人与开发团队对系统的初始规定有一个共同的理解，使之成为整个开发工作的基础。SRS 是软件开发过程中最重要的文档之一，对于任何规模和性质的软件项目都不应该缺少。

需求定义方法

（1）严格定义也称为预先定义，需求的严格定义建立在以下的基本假设之上：所有需求都能够被预先定义。开发人员与用户之间能够准确而清晰地交流。采用图形（或文字）可以充分体现最终系统。

（2）原型方法，迭代的循环型开发方式，需要注意的问题：并非所有的需求都能在系统开发前被准确地说明。项目干系人之间通常都存在交流上的困难，原型提供了克服该困难的一个手段。特点：需要实际的、可供用户参与的系统模型。有合适的系统开发环境。反复是完全需要和值得提倡的，需求一旦确定，就应遵从严格的方法。

2.5 需求验证

也称为需求确认，目的是与用户一起确认需求无误，对需求规格说明书 SAS 进行评审和测试，包括两个步骤：

需求评审：正式评审和非正式评审。

需求测试：设计概念测试用例。

需求验证通过后，要请用户签字确认，作为验收标准之一，此时，这个需求规格说明书就是需求基线，不可以再随意更新，如果需要更改必须走需求变更流程。

2.6 需求管理

定义需求基线：通过了评审的需求说明书就是需求基线，下次如果需要变更需求，就需要按照流程来一步步进行。需求的流程及状态如下图所示：

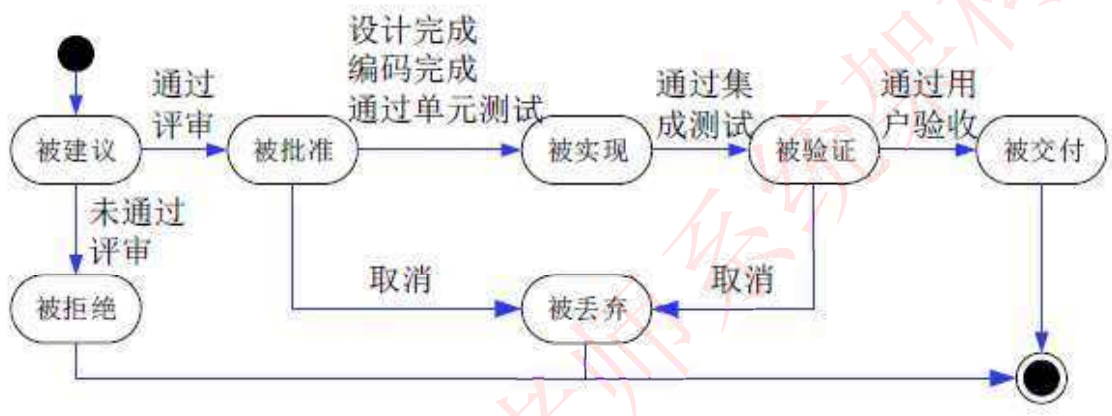


图 11-17 需求状态的变化

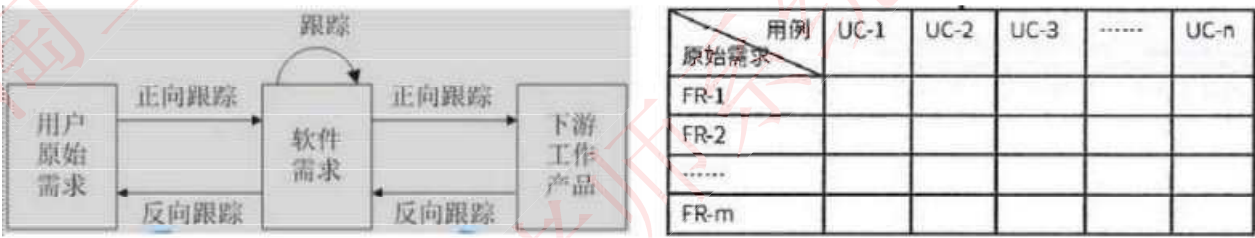
需求变更和风险

主要关心需求变更过程中的需求风险管理，带有风险的做法有：无足够用户参与、忽略了用户分类、用户需求的不断增加、模棱两可的需求、不必要的特性、过于精简的 SRS、不准确的估算。

变更产生的原因：外部环境的变化、需求和设计做的不够完整、新技术的出现、公司机构重组造成业务流程的变化。

变更控制委员会 CCB：也称为配置控制委员会，其任务时对建议的配置项变更做出评价、审批，以及监督已经批准变更的实施。

需求跟踪：双向跟踪，两个层次，如下图所示：



正向跟踪表示用户原始需求是否都实现了，反向跟踪表示软件实现的是否都是用户要求的，不多不少，可以用原始需求和用例表格（需求跟踪矩阵）来表示：

若原始需求和用例有对应，则在对应栏打对号，若某行都没有对号，表明原始需求未实现，正向跟踪发现问题；若某列都没有对号，表明有多余功能用例，软件实现了多余功能，反向跟踪发现问题。

3 系统设计

3.1 处理流程设计

流程表示工具

程序流程图（Program Flow Diagram，PFD）用一些图框表示各种操作，它独立于任何一种程序设计语言，比较直观、清晰，易于学习掌握。任何复杂的程序流程图都应该由顺序、选择和循环结构组合或嵌套而成。

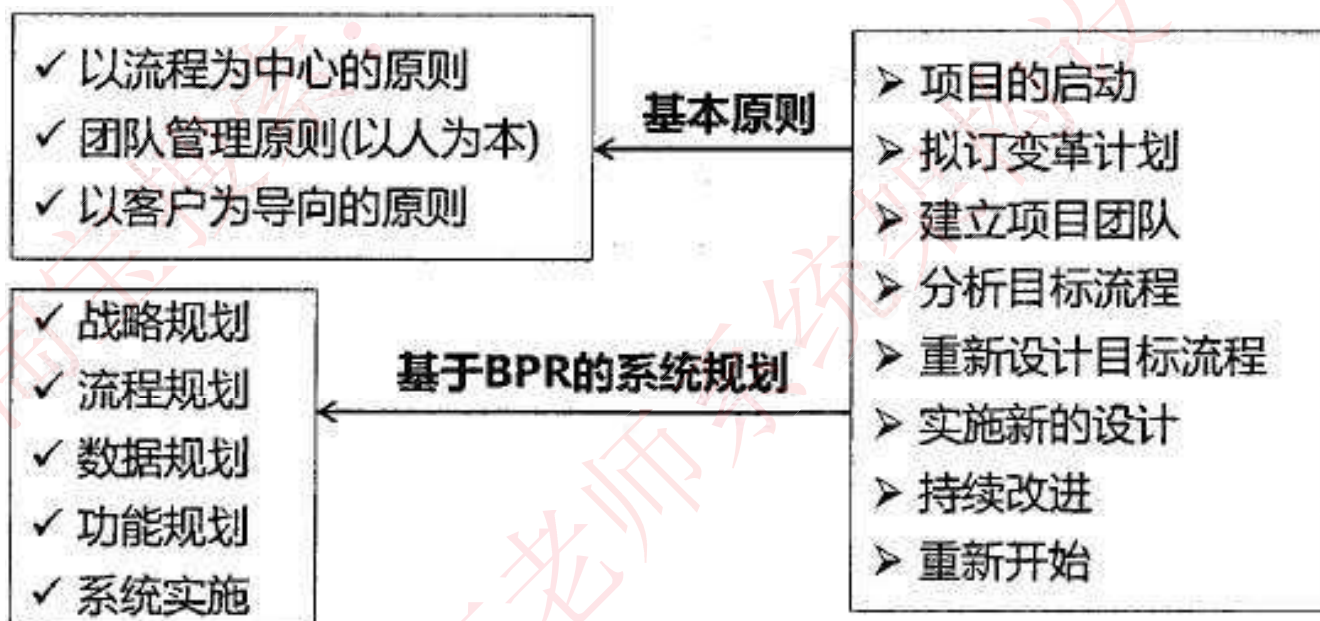
IPO 图也是流程描述工具，用来描述构成软件系统的每个模块的输入、输出和数据加工。

N-S 图容易表示嵌套和层次关系，并具有强烈的结构化特征。但是当问题很复杂时，N-S 图可能很大，因此不适合于复杂程序的设计。

问题分析图（PAD）是一种支持结构化程序设计的图形工具。PAD 具有清晰的逻辑结构、标准化的图形等优点，更重要的是，它引导设计人员使用结构化程序设计方法，从而提高程序的质量。

◆业务流程重组 BPR

BPR 是对企业的业务流程进行根本性的再思考和彻底性的再设计，从而获得可以用诸如成本、质量、服务和速度等方面的业绩来衡量的显著性的成就。BPR 设计原则、系统规划和步骤如下图所示：



◆业务流程管理 BPM

BPM 是一种以规范化的构造端到端的卓越业务流程为中心，以持续的提高组织业务绩效为目的的系统化方法。

BPM 与 BPR 管理思想最根本的不同就在于流程管理并不要求对所有的流程进行再造。构造卓越的业务流程并不是流程再造，而是根据现有流程的具体情况，对流程进行规范化的设计

流程管理包含三个层面：规范流程、优化流程和再造流程

3.2 系统设计

系统设计主要目的：为系统制定蓝图，在各种技术和实施方法中权衡利弊，精心设计，合理地使用各种资源，最终勾画出新系统的详细设计方法。

系统设计方法：结构化设计方法，面向对象设计方法。

系统设计的主要内容：概要设计、详细设计。

概要设计基本任务：又称为系统总体结构设计，是将系统的功能需求分配给软件模块，确定每个模块的功能和调用关系，形成软件的模块结构图，即系统结构图。

详细设计的基本任务：模块内详细算法设计、模块内数据结构设计、数据库的物理设计、其他设计（代码、输入/输出格式、用户界面）、编写详细设计说明书、评审。

系统设计基本原理

抽象化；

自顶而下，逐步求精；

信息隐蔽；

模块独立（高内聚，低耦合）。

系统设计原则

保持模块的大小适中；

尽可能减少调用的深度；

多扇入，少扇出；

单入口，单出口；

模块的作用域应该在模块之内；

功能应该是可预测的。

内聚程度从低到高如下表所示：

内聚分类	定义	记忆关键字
偶然内聚	一个模块内的各处理元素之间没有任何联系	无直接关系
逻辑内聚	模块内执行若干个逻辑上相似的功能，通过参数确定该模块完成哪一个功能	逻辑相似、参数决定
时间内聚	把需要同时执行的动作组合在一起形成的模块。	同时执行
过程内聚	一个模块完成多个任务，这些任务必须按指定的过程执行	指定的过程顺序
通信内聚	模块内的所有处理元素都在同一个数据结构上操作，或者各处理使用相同的输入数据或者产生相同的输出数据	相同数据结构、相同输入输出
顺序内聚	一个模块中的各个处理元素都密切相关于同	顺序执行、输入为输

	一功能且必须顺序执行，前一个功能元素的输出就是下一个功能元素的输入	出
功能内聚	最强的内聚，模块内的所有元素共同作用完成一个功能，缺一不可	共同作用、缺一不可

耦合程度从低到高如下所示：

耦合分类	定义	记忆关键字
无直接耦合	两个模块之间没有直接的关系，它们分别从属于不同模块的控制与调用，不传递任何信息	无直接关系
数据耦合	两个模块之间有调用关系，传递的是简单的数据值，相当于高级语言中的值传递	传递数据值调用
标记耦合	两个模块之间传递的是数据结构	传递数据结构
控制耦合	一个模块调用另一个模块时，传递的是控制变量，被调用模块通过该控制变量的值有选择的执行模块内的某一功能	控制变量、选择执行某一功能
外部耦合	模块间通过软件之外的环境联合（如 I/O 将模块耦合到特定的设备、格式、通信协议上）时。	软件外部环境
公共耦合	通过一个公共数据环境相互作用的那些模块间的耦合	公共数据结构
内容耦合	当一个模块直接使用另一个模块的内部数据，或通过非正常入口转入另一个模块内部时。	模块内部关联

系统结构图（SC）又称为模块结构图，它是软件概要设计阶段的工具，反映系统的功能实现和模块之间的联系与通信，包括各模块之间的层次结构，即反映了系统的总体结构。

详细设计的表示工具有图形工具、表格工具和语言工具。其中图形工具有业务流图、程序流程图、PAD 图、NS 流程图。

程序流程图又称为程序框图，是使用最广泛的一种描述程序逻辑结构的工具。它用方框表示一个处理步骤，菱形表示一个逻辑条件，箭头表示控制流向。其优点是：结构清晰，易于理解，易于修改。缺点是：只能描述执行过程而不能描述有关的数据。

NS 流程图，也称为盒图，是一种强制使用结构化构造的图示工具，也称为方框图。其具有以下特点：功能域明确、不可能任意转移控制、很容易确定局部和全局数据的作用域、很容易表示嵌套关系及模板的层次关系。

PAD 图是一种改进的图形描述方式，可以用来取代程序流程图，相比程序流程图更直观，结构更清晰。最大的优点是能够反映和描述自顶向下的历史和过程。PAD 提供了 5 种基本控制结构的图示，并允许递归使用。

3.3 人机界面设计

人机界面设计的三大原则：置于用户控制之下、减少用户的记忆负担、保持界面的一致性。

置于用户的控制之下

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方式；

提供灵活的交互；

允许用户交互可以被中断和取消；

当技能级别增加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

减少对短期记忆的要求；

建立有意义的缺省；

定义直觉性的捷径；

界面的视觉布局应该基于真实世界的隐喻；

以不断进展的方式揭示信息。

保持界面的一致性

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

如过去的交互模型已建立起了用户期望，除非有迫不得已的理由，不要去改变它。

4 测试基础知识

4.1 测试基础

测试原则

应尽早并不断的进行测试；

测试工作应该避免由元开发软件的人或小组承担；

在设计测试方案时，不仅要确定输入数据，而且要根据系统功能确定预期的输出结果；

既包含有效、合理的测试用例，也包含不合理、失效的用例；

检验程序是否做了该做的事，且是否做了不该做的事；

严格按照测试计划进行；

妥善保存测试计划和测试用例；
测试用例可以重复使用或追加测试。

测试方法的分类

测试过程中程序执行状态：动态测试、静态测试。

具体实现算法细节和系统内部结构的相关情况为根据可分黑盒测试、白盒测试和灰盒测试。

程序执行方式：人工测试、自动化测试（自动化执行测试脚本）。

动态测试：程序运行时测试，分为

黑盒测试法：功能性测试，不了解软件代码结构，根据功能设计用例，测试软件功能。

白盒测试法：结构性测试，明确代码流程，根据代码逻辑设计用例，进行用例覆盖。

灰盒测试法：即既有黑盒，也有白盒。

静态测试：程序静止时，即对代码进行人工审查，分为

桌前检查：程序员检查自己编写的程序，在程序编译后，单元测试前。

代码审查：由若干个程序员和测试人员组成评审小组，通过召开程序评审会来进行审查。

代码走查：也是采用开会来对代码进行审查，但并非简单的检查代码，而是由测试人员提供测试用例，让程序员扮演计算机的角色，手动运行测试用例，检查代码逻辑。

4.2 测试阶段

（1）单元测试：也称为模块测试，测试的对象是可独立编译或汇编的程序模块、软件构件或 OO 软件中的类（统称为模块），测试依据是软件详细设计说明书。

（2）集成测试：目的是检查模块之间，以及模块和已集成的软件之间的接口关系，并验证已集成的软件是否符合设计要求。测试依据是软件概要设计文档。

（3）确认测试：主要用于验证软件的功能、性能和其他特性是否与用户需求一致。根据用户的参与程度，通常包括以下类型：

内部确认测试：主要由软件开发组织内部按照 SRS 进行测试。

Alpha 测试：用户在开发环境下进行测试。

Beta 测试：用户在实际使用环境下进行测试，通过改测试后，产品才能交付用户。

验收测试：针对 SRS，在交付前以用户为主进行的测试。其测试对象为完整的、集成的计算机系统。验收测试的目的是，在真实的用户工作环境下，检验软件系统是否满足开发技术合同或 SRS。验收测试的结论是用户确定是否接收该软件的主要依据。除应满足一般测试的准入条件外，在进行验收测试之前，应确认被测软件系统已通过系统测试。

（4）系统测试：测试对象是完整的、集成的计算机系统；测试的目的是在真实系统工作环境下，验证完成的软件配置项能否和系统正确连接，并满足系统/子系统设计文档和软件开发合同规定的要求。测试依据是用户需求或开发合同。

主要包括功能测试、健壮性测试、性能测试、用户界面测试、安全性测试、安装与反安装测试等，其中，最重要的工作是进行功能测试与性能测试。功能测试主要采用黑盒测试方法；性能测试主要指标有响应时间、吞吐量、并发用户数和资源利用率等。

(5) 配置项测试：测试对象是软件配置项，测试目的是检验软件配置项与 SRS 的一致性。测试的依据是 SRS。在此之前，应确认被测软件配置项已通过单元测试和集成测试。

(6) 回归测试：测试目的是测试软件变更之后，变更部分的正确性和对变更需求的符合性，以及软件原有的、正确的功能、性能和其他规定的要求的不损害性。

其他测试

(1) AB 测试是为 Web 或 App 界面或流程制作两个或多个版本，在同一时间维度，分别让组成成分相同（相似）的访客群组（目标人群）随机的访问这些版本，收集各群组的用户体验数据和业务数据，最后分析、评估出最好版本，正式采用。

(2) Web 测试是软件测试的一部分，是针对 Web 应用的一类测试。由于 Web 应用与用户直接相关，又通常需要承受长时间的大量操作，因此 Web 项目的功能和性能都必须经过可靠的验证。

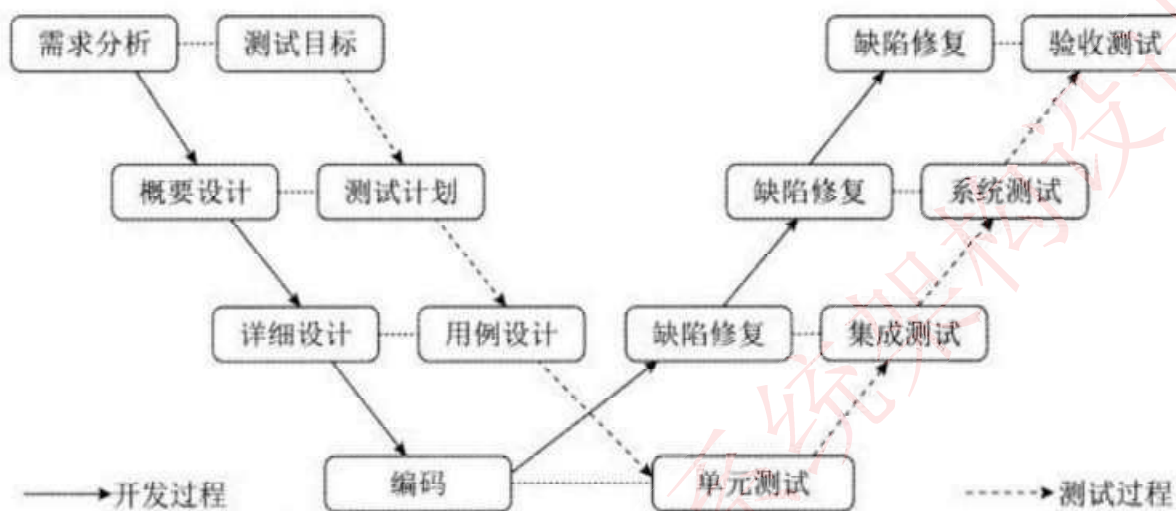
(3) 链接测试。链接是 Web 应用系统的一个主要特征，它是在页面之间切换和指导用户去一些未知地址页面的主要手段。链接测试可分为 3 个方面。首先，测试所有链接是否按指示那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在；最后，保证 Web 应用系统上没有孤立的页面。

(4) 表单测试。当用户通过表单提交信息的时候，都希望表单能正常工作。如果使用表单来进行在线注册，要确保提交按钮能正常工作，当注册完成后应返回注册成功的消息。如果使用表单收集配送信息，应确保程序能够正确处理这些数据，最后能让用户收到信息。

4.3 测试模型

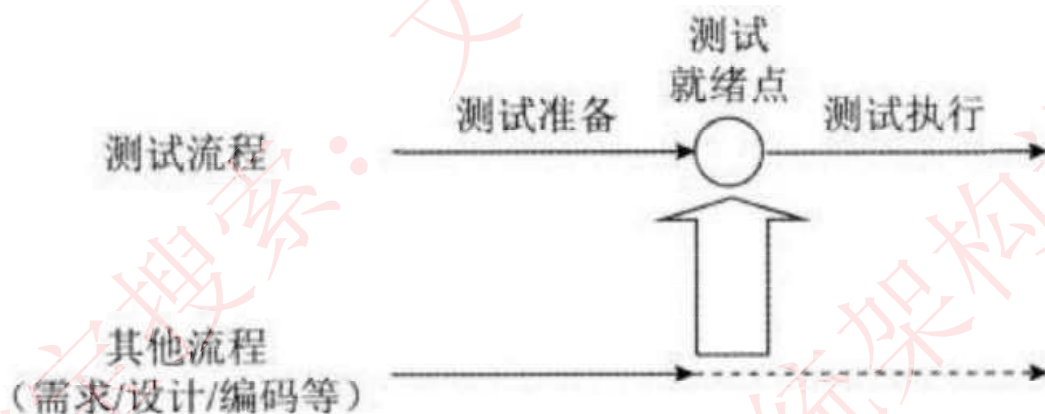
W 模型：W 模型是对 V 模型的一个重要改进，充分体现了尽早开展测试的原则，并将 V 模型中以发现缺陷为目标上升为保证软件质量为目标。W 模型如图所示。

W 模型实际上是两个 V 的叠加，一个 V 描述开发过程，另外一个 V 描述测试过程。但测试的起始时机不再是编码结束之后，而是从需求分析时开始，且与开发的每一个阶段活动同步进行。



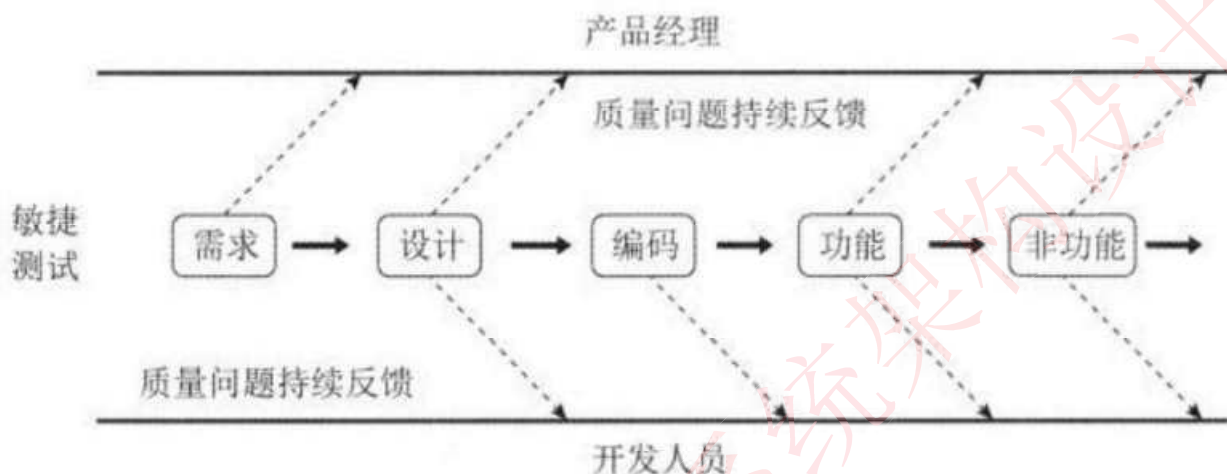
H 模型：改进了 W 和 V 模型高度依赖于开发的瀑布模型的缺陷，其特征如图所示：

H 模型把测试活动从软件开发过程中独立出来，在软件过程的任何一个时间点上，只要测试条件满足即开展测试。测试的流程与其他流程是并行的。H 模型比 W 模型更好的地方是能够兼顾测试的效率和灵活性，适合于各种规模及类型的软件项目。



敏捷测试模型：敏捷测试源于敏捷开发。敏捷测试是敏捷开发的组成部分，需要与开发流程良好融合，其特征如图所示。

敏捷测试在整个敏捷开发过程中，需要与项目的其他人员甚至用户保持紧密协作，时刻关注需求变化并实施测试，以体现测试的时效性和适应性，这对测试人员有比较高的能力要求。



4.4 测试用例设计

黑盒测试用例：将程序看做一个黑盒子，只知道输入输出，不知道内部代码，由此设计出测试用例，分为下面几类：

等价类划分：把所有的数据按照某种特性进行归类，而后在每类的数据里选取一个即可。等价类测试用例的设计原则：设计一个新的测试用例，使其尽可能多地覆盖尚未被覆盖的有效等价类，重复这一步，直到所有的有效等价类都被覆盖为止；设计一个新的测试用例，使其**仅覆盖一个尚未被覆盖的无效等价类**，重复这一步，直到所有的无效等价类都被覆盖为止。

边界值划分：将每类的边界值作为测试用例，边界值一般为范围的两端值以及在此范围之外的与此范围间隔最小的两个值，如年龄范围为 0-150，边界值为 0,150, -1,151 四个。

错误推测：没有固定的方法，凭经验而言，来推测有可能产生问题的地方，作为测试用例进行测试。

因果图：由一个结果来反推原因的方法，具体结果具体分析，没有固定方法。

白盒测试用例：知道程序的代码逻辑，按照程序的代码语句，来设计覆盖代码分支的测试用例，覆盖级别从低至高分为下面几种：

(1) 语句覆盖 SC：逻辑代码中的所有语句都要被执行一遍，覆盖层级最低，因为执行了所有的语句，不代表执行了所有的条件判断。

(2) 判定覆盖 DC：逻辑代码中的所有判断语句的条件真假分支都要覆盖一次。

语句覆盖 (SC)

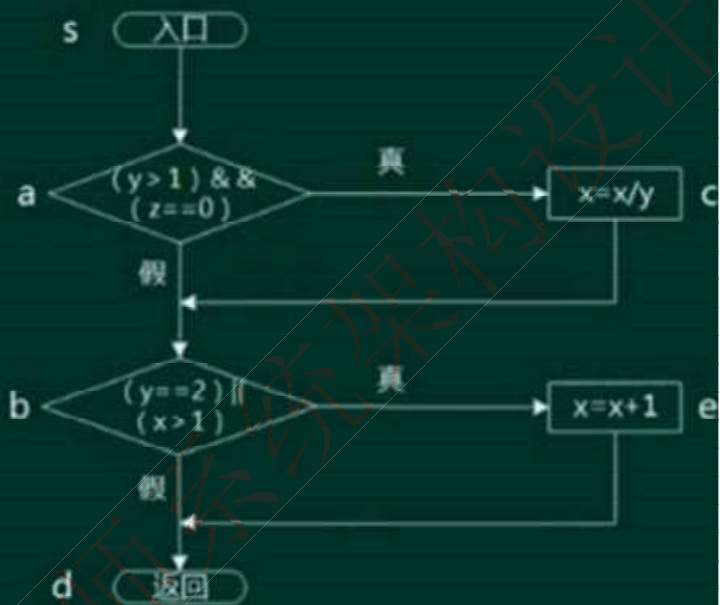
X=4, Y=2, Z=0 sacbed

判定覆盖 (DC)

又称分支覆盖

X=1, Y=3, Z=0 sacbd

X=3, Y=2, Z=1 sabed



(3) 条件覆盖 CC: 针对每一个判断条件内的每一个独立条件都要执行一遍真和假。

(4) 条件判定组合覆盖 CDC: 同时满足判定覆盖和条件覆盖。

条件覆盖 (CC)

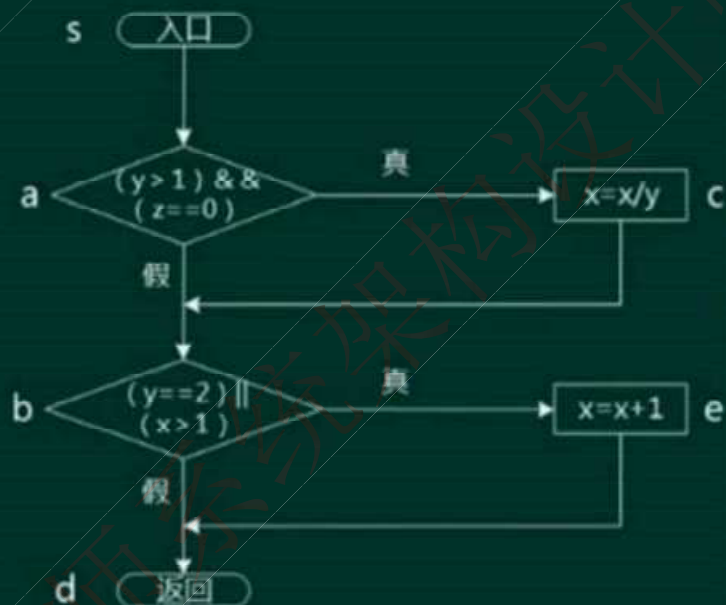
X=1, Y=2, Z=0 sacbed

X=2, Y=1, Z=1 sabed

条件判定组合覆盖 (CDC)

X=4, Y=2, Z=0 sacbed

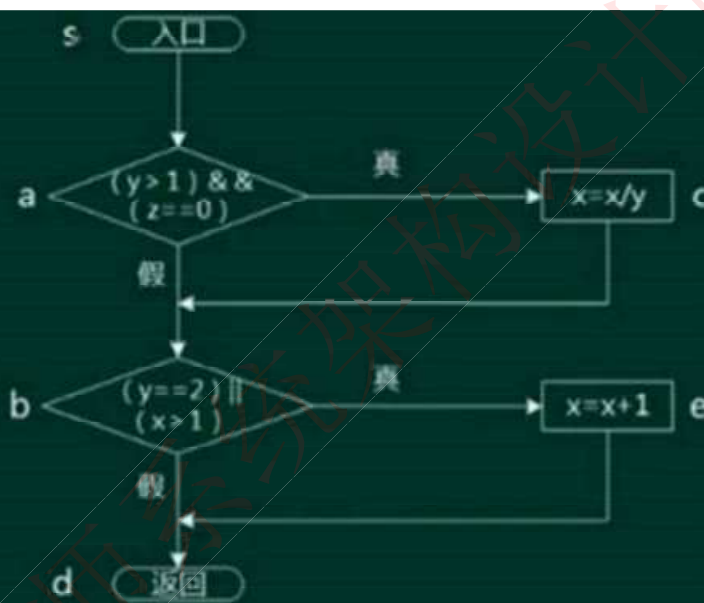
X=1, Y=1, Z=1 sabd



(5) 路径覆盖: 逻辑代码中的所有可行路径都覆盖了, 覆盖层级最高。

路径覆盖

X=1, Y=1, Z=1	sabd
X=3, Y=2, Z=0	sacbed
X=3, Y=3, Z=0	sacbd
X=1, Y=2, Z=1	sabed



4.5 调试

测试是发现错误，调试是找出错误的代码和原因。

调试需要确定错误的准确位置；确定问题的原因并设法改正；改正后要进行回归测试。

调试的方法

蛮力法：又称为穷举法或枚举法，穷举出所有可能的方法一一尝试。

回溯法：又称为试探法，按选优条件向前搜索，以达到目标，当发现原先选择并不优或达不到目标，就退回一步重新选择，这种走不通就退回再走的技术为回溯法。

演绎法：是由一般到特殊的推理方法，与“归纳法”相反，从一般性的前提出发。得出具体陈述或个别结论的过程。

归纳法：是由特殊到一般的推理方法，从测试所暴露的问题出发，收集所有正确或不正确的数据，分析它们之间的关系，提出假想的错误原因，用这些数据来证明或反驳，从而查出错误所在。

软件度量

软件的两种属性：外部属性指面向管理者和用户的属性，可直接测量，一般为性能指标。内部属性指软件产品本身的属性，如可靠性等，只能间接测量。

McCabe 度量法：又称为环路复杂度，假设有向图中有向边数为 m ，节点数为 n ，则此有向图的环路复杂度为 $m-n+2$ 。

注意 m 和 n 代表的含义不能混淆，可以用一个最简单的环路来做特殊值记忆此公式，另外，针对一个程序流程图，每一个分支边（连线）就是一条有向边，每一条语句（语句框）就是一个顶点。

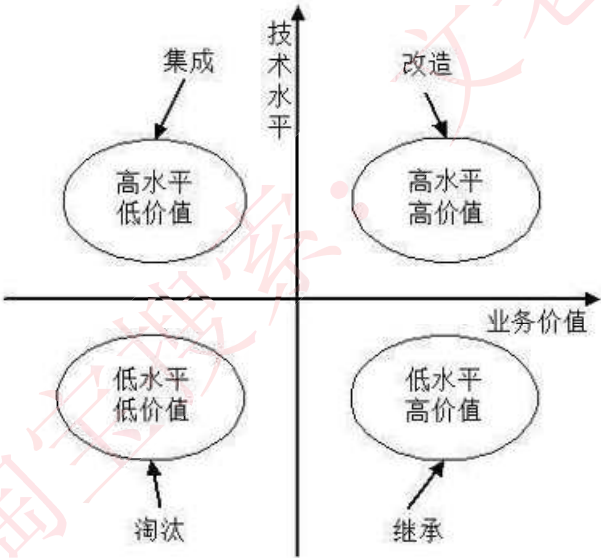
此外，推荐使用另一种简单的计算公式：环路复杂度=判定节点个数+1。

5 系统运行与维护

5.1 系统转换

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，它通常具有以下特点：

- (1) 系统虽然完成企业中许多重要的业务管理工作，但仍然不能完全满足要求。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。例如，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) 通常是大型的软件系统，已经融入企业的业务运作和决策管理机制之中，维护工作十分困难。
- (4) 没有使用现代信息系统建设方法进行管理和开发，现在基本上已经没有文档，很难理解。



系统转换是指新系统开发完毕，投入运行，取代现有系统的过程，需要考虑多方面的问题，以实现与老系统的交接，有以下三种转换计划：

直接转换：现有系统被新系统直接取代了，风险很大，适用于新系统不复杂，或者现有系统已经不能使用的情况。优点是节省成本。

并行转换：新系统和老系统并行工作一段时间，新系统经过试运行后再取代，若新系统在试运行过程中有问题，也不影响现有系统的运行，风险极小，在试运行过程中还可以比较新老系统的性能，适用于大型系统。缺点是耗费人力和时间资源，难以控制两个系统间的数据转换。

分段转换：分期分批逐步转换，是直接和并行转换的集合，将大型系统分为多个子系统，依次试运行每个子系统，成熟一个子系统，就转换一个子系统。同样适用于大型项目，只是更耗时，而且现有系统和新系统间混合使用，需要协调好接口等问题。

数据转换与迁移：将数据从旧数据库迁移到新数据库中。要在新系统中尽可能的保存旧系统中合理的数据结构，才能降低迁移的难度。也有三种方法：系统切换前通过工具迁移、系统切换前采用手工录入、系统切换后通过新系统生成。

5.2 系统维护概述

系统的可维护性可以定义为维护人员理解、改正、改动和改进这个软件的难易程度，其评价指标如下：

- (1) 易分析性。软件产品诊断软件中的缺陷或失效原因或识别待修改部分的能力。
- (2) 易改变性。软件产品使指定的修改可以被实现的能力，实现包括编码、设计和文档的更改。
- (3) 稳定性。软件产品避免由于软件修改而造成意外结果的能力。
- (4) 易测试性。软件产品使已修改软件能被确认的能力。
- (5) 维护性的依从性。软件产品遵循与维护性相关的标准或约定的能力。

系统维护包括硬件维护、软件维护和数据维护，其中软件维护类型如下：

正确性维护：发现了 bug 而进行的修改。

适应性维护：由于外部环境发生了改变，被动进行的对软件的修改和升级。

完善性维护：基于用户主动对软件提出更多的需求，修改软件，增加更多的功能，使其比之前的软件功能、性能更高，更加完善。

预防性维护：对未来可能发生的 bug 进行预防性的修改。

6 净室软件工程

净室软件工程是一种应用数学与统计学理论以经济的方式生产高质量软件的工程技术，力图通过严格的工程化的软件过程达到开发中的零缺陷或接近零缺陷。净室方法不是先制作一个产品，再去消除缺陷，而是要求在规约和设计中消除错误，然后以“净”的方式制作，可以降低软件开发中的风险，以合理的成本开发出高质量的软件。

在净室软件工程背后的哲学是：通过在第 1 次正确地书写代码增量，并在测试前验证它们的正确性，来避免对成本很高的错误消除过程的依赖。它的过程模型是在代码增量积聚到系统的过程的同时，进行代码增量的统计质量验证。它甚至提倡开发者不需要进行单元测试，而是进行正确性验证和统计质量控制。

净室软件工程（CSE）的理论基础主要是函数理论和抽样理论。

净室软件工程应用技术手段：

1. 统计过程控制下的增量式开发。
2. 基于函数的规范与设计。
3. 正确性验证。CSE 的核心
4. 统计测试和软件认证。

净室软件工程在使用过程的一些缺点：

- (1) CSE 太理论化，需要更多的数学知识。其正确性验证的步骤比较困难且比较耗时。
- (2) CSE 开发小组不进行传统的模块测试，这是不现实的。
- (3) CSE 也会带有传统软件工程的一些弊端。

7 基于构件的软件工程

基于构件的软件工程 (CBSE) 是一种基于分布对象技术、强调通过可复用构件设计与构造软件系统的软件复用途径。CBSE 体现了“购买而不是重新构造”的哲学，将软件开发的重点从程序编写转移到了基于已有构件的组装，以更快地构造系统，减轻用来支持和升级大型系统所需要的维护负担，从而降低软件开发的费用。

用于 CBSE 的构件应该具备以下特征。

- (1) 可组装型：对于可组装的构件，所有外部交互必须通过公开定义的接口进行。同时它还必须对自身信息的外部访问。
- (2) 可部署性：软件必须是自包含的，必须能作为一个独立实体在提供其构件模型实现的构件平台上运行。构件总是二进制形式，无须在部署前编译。
- (3) 文档化：构件必须是完全文档化的，用户根据文档来判断构件是否满足需求。
- (4) 独立性：构件应该是独立的，应该可以在无其他特殊构件的情况下进行组装和部署，如确实需要其他构件提供服务，则应显示声明。
- (5) 标准化：构件标准化意味着在 CBSE 过程中使用的构件必须符合某种标准化的构件模型。

构件模型定义了构件实现、文档化以及开发的标准，其包含的模型要素为：

- (1) 接口。构件通过构件接口来定义，构件模型规定应如何定义构件接口以及在接口定义中应该包含的要素，如操作名、参数以及异常等。
- (2) 使用信息。为使构件远程分布和访问，必须给构件一个特定的、全局唯一的名字或句柄。构件元数据是构件本身相关的数据，比如构件的接口和属性信息。
- (3) 部署。构件模型包括一个规格说明，指出应该如何打包构件使其部署成为一个独立的可执行实体。部署信息中包含有关包中内容的信息和它的二进制构成的信息。

构件模型提供了一组被构件使用的通用服务，这种服务包括以下两种。

- 平台服务，允许构件在分布式环境下通信和互操作。
- 支持服务，这是很多构件需要的共性服务。例如，构件都需要的身份认证服务。

中间件实现共性的构件服务，并提供这些服务的接口。

CBSE 过程是支持基于构件组装的软件开发过程，过程中的 6 个主要活动：系统需求概览、识别候选构件、根据发现的构件修改需求、体系结构设计、构件定制与适配、组装构件创建系统。

CBSE 过程与传统软件开发过程不同点：

- (1) CBSE 早期需要完整的需求，以便尽可能多地识别出可复用的构件。

(2) 在过程早期阶段根据可利用的构件来细化和修改需求。如果可利用的构件不能满足用户需求，就应该考虑由复用构件支持的相关需求。

(3) 在系统体系结构设计完成后，会有一个进一步的对构件搜索及设计精化的活动。可能需要为某些构件寻找备用构件，或者修改构件以适合功能和架构的要求。

(4) 开发就是将已经找到的构件集成在一起的组装过程。

构件组装是指构件相互直接集成或是用专门编写的“胶水代码”将它们整合在一起来创建一个系统或另一个构件的过程。常见的组装构件有以下 3 种组装方式。

1. 顺序组装。通过按顺序调用已经存在的构件，可以用两个已经存在的构件来创建一个新的构件。如上一个构件输出作为下一个构件的输入。

2. 层次组装。这种情况发生在一个构件直接调用自另一个构件所提供的服务时。被调用的构件为调用的构件提供所需的服务。二者之间接口匹配兼容。

3. 叠加组装。这种情况发生在两个或两个以上构件放在一起创建一个新构件的时候。这个新构件合并了原构件的功能，从而对外提供了新的接口。外部应用可以通过新接口来调用原有构件的接口，而原有构件不互相依赖，也不互相调用。这种组装类型适合于构件是程序单元或者构件是服务的情况。

构件组装的三种不兼容问题（通过编写适配器解决）：

(1) 参数不兼容。接口每一侧的操作有相同的名字，但参数类型或参数个数不相同。

(2) 操作不兼容。提供接口和请求接口的操作名不同。

(3) 操作不完备。一个构件的提供接口是另一个构件请求接口的一个子集，或者相反。

十一 面向对象技术

1 面向对象基础

面向对象的基本概念

(1) 对象：由数据及其操作所构成的封装体，是系统中用来描述客观事务的一个实体，是构成系统的一个基本单位。一个对象通常可以由对象名、属性和方法 3 个部分组成。

(2) 类：现实世界中实体的形式化描述，类将该实体的属性（数据）和操作（函数）封装在一起。对象是类的实例，类是对象的模板。

类可以分为三种：实体类、接口类（边界类）和控制类。实体类的对象表示现实世界中真实的实体，如人、物等。接口类（边界类）的对象为用户提供一种与系统合作交互的方式，分为人和系统两大类，其中人的接口可以是显示屏、窗口、Web 窗体、对话框、菜单、列表框、其他显示控制、条形码、二维码或者用户与系统交互的其他方法。系统接口涉及到把数据发送到其他系统，或者从其他系统接收数据。控制类的对象用来控制活动流，充当协调者。

(3) 抽象：通过特定的实例抽取共同特征以后形成概念的过程。它强调主要特征，忽略次要特征。一个对象是现实世界中一个实体的抽象，一个类是一组对象的抽象，抽象是一种单一化的描述，它强调给出与应用相关的特性，抛弃不相关的特性。

(4) 封装：是一种信息隐蔽技术，将相关的概念组成一个单元模块，并通过一个名称来引用。面向对象封装是将数据和基于数据的操作封装成一个整体对象，对数据的访问或修改只能通过对象对外提供的接口进行。

(5) 继承：表示类之间的层次关系（父类与子类），这种关系使得某类对象可以继承另外一类对象的特征，又可分为单继承和多继承。

(6) 多态：不同的对象收到同一个消息时产生完全不同的结果。包括参数多态（不同类型参数多种结构类型）、包含多态（父子类型关系）、过载多态（类似于重载，一个名字不同含义）、强制多态（强制类型转换）四种类型。多态由继承机制支持，将通用消息放在抽象层，具体不同的功能实现放在低层。

(7) 接口：描述对操作规范的说明，其只说明操作应该做什么，并没有定义操作如何做。

(8) 消息：体现对象间的交互，通过它向目标对象发送操作请求。

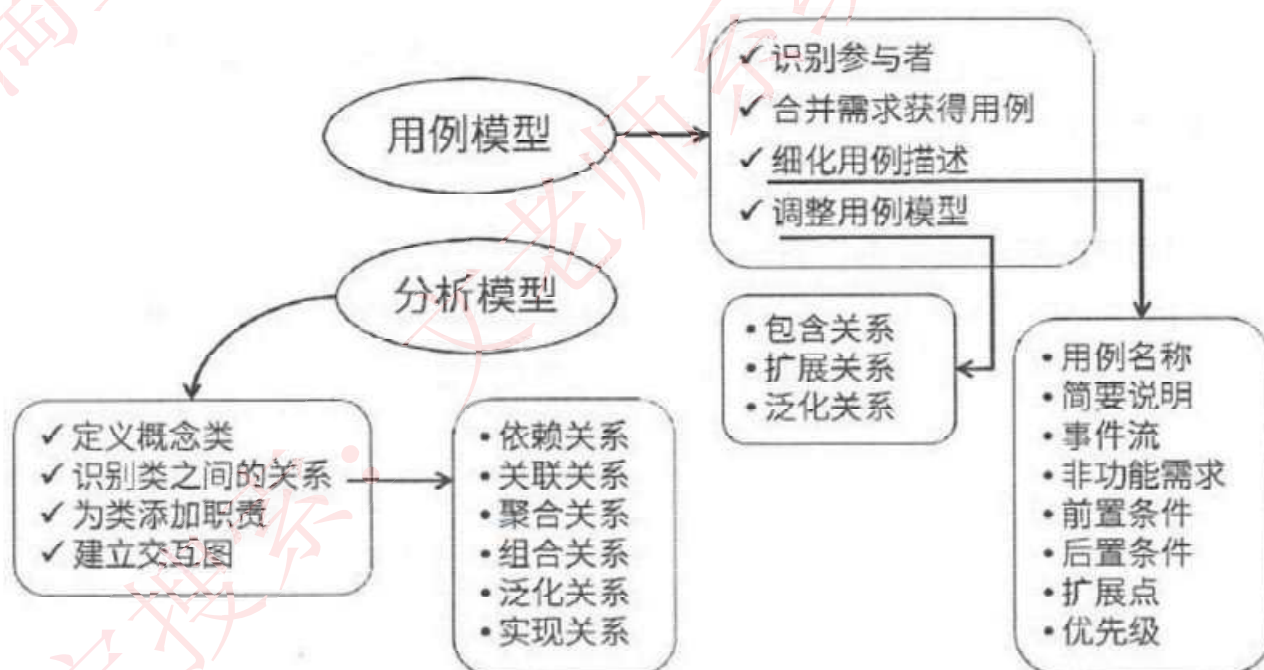
(9) 覆盖：子类在原有父类接口的基础上，用适合于自己要求的实现去置换父类中的相应实现。即在子类中重定义一个与父类同名同参的方法。

(10) 函数重载：与覆盖要区分开，函数重载与子类父类无关，且函数是同名不同参数。

(11) 绑定是一个把过程调用和响应调用所需要执行的代码加以结合的过程。在一般的程序设计语言中，绑定是在编译时进行的，叫作静态绑定。动态绑定则是在运行时进行的，因此，一个给定的过程调用和代码的结合直到调用发生时才进行。

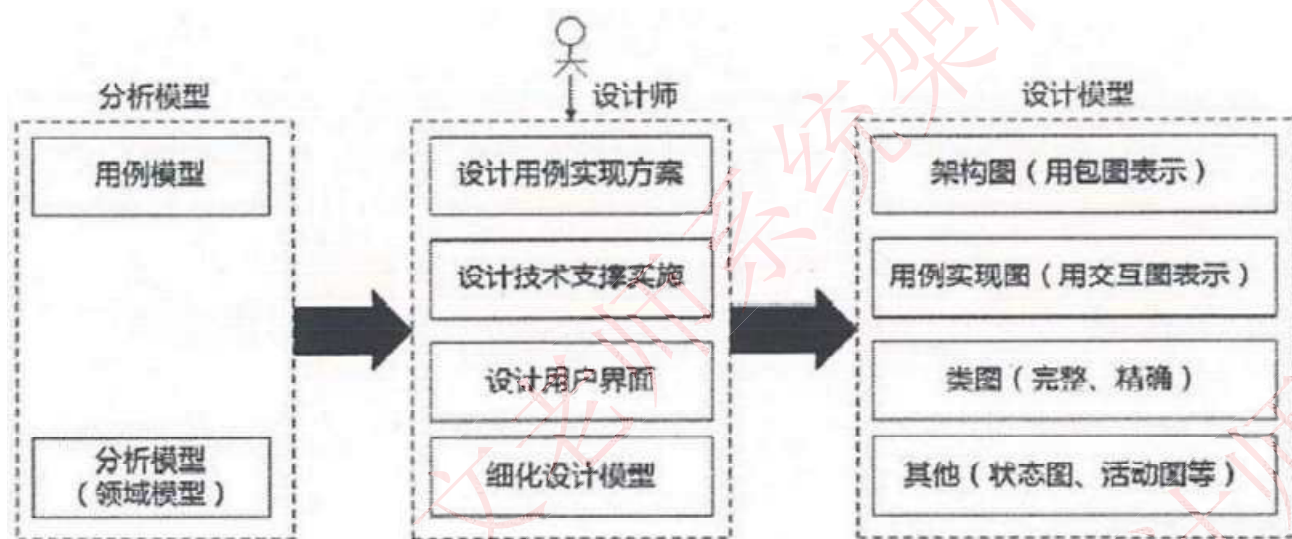
面向对象的分析：是为了确定问题域，理解问题。包含五个活动：认定对象、组织对象、描述对象间的相互作用、确定对象的操作、定义对象的内部信息。

面向对象需求建模：



面向对象的设计：是设计分析模型和实现相应源代码，设计问题域的解决方案，与技术相关。OOD 同样应遵循抽象、信息隐蔽、功能独立、模块化等设计准则。

面向对象的分析模型主要由顶层架构图、用例与用例图、领域概念模型构成；设计模型则包含以包图表示的软件体系结构图、以交互图表示的用例实现图、完整精确的类图、针对复杂对象的状态图和用以描述流程化处理过程的活动图等。



面向对象的设计原则：

(1) 单一责任原则。就一个类而言，应该仅有一个引起它变化的原因。即，当需要修改某个类的时候原因有且只有一个，让一个类只做一种类型责任。

(2) 开放—封闭原则。软件实体（类、模块、函数等）应该是可以扩展的，即开放的；但是不可修改的，即封闭的。

(3) 里氏替换原则。子类型必须能够替换掉他们的基类型。即，在任何父类可以出现的地方，都可以用子类的实例来赋值给父类型的引用。

(4) 依赖倒置原则。抽象不应该依赖于细节，细节应该依赖于抽象。即，高层模块不应该依赖于低层模块，二者都应该依赖于抽象。

(5) 接口分离原则。不应该强迫客户依赖于它们不用的方法。接口属于客户，不属于它所在的类层次结构。即：依赖于抽象，不要依赖于具体，同时在抽象级别不应该有对于细节的依赖。这样做的好处就在于可以最大限度地应对可能的变化。

上述(1)～(5)是面向对象方法中的五大原则。除了这五大原则之外，Robert C. Martin 提出的面向对象设计原则还包括以下几个。

(6) 重用发布等价原则。重用的粒度就是发布的粒度。

(7) 共同封闭原则。包中的所有类对于同一类性质的变化应该是共同封闭的。一个变化若对一个包产生影响，则将对包中的所有类产生影响，而对于其他的包不造成任何影响。

(8) 共同重用原则。一个包中的所有类应该是共同重用的。如果重用了包中的一个类，那么就要重用包中的所有类。

(9) 无环依赖原则。在包的依赖关系图中不允许存在环，即包之间的结构必须是一个直接的五环图形。

(10) 稳定依赖原则。朝着稳定的方向进行依赖。

(11) 稳定抽象原则。包的抽象程度应该和其稳定程度一致。

面向对象的测试：

一般来说，对面向对象软件的测试可分为下列 4 个层次进行。

(1) 算法层。测试类中定义的每个方法，基本上相当于传统软件测试中的单元测试。

(2) 类层。测试封装在同一个类中的所有方法与属性之间的相互作用。在面向对象软件中类是基本模块，因此可以认为这是面向对象测试中所特有的模块测试。

(3) 模板层。测试一组协同工作的类之间的相互作用，大体上相当于传统软件测试中的集成测试，但是也有面向对象软件的特点（例如，对象之间通过发送消息相互作用）。

(4) 系统层。把各个子系统组装成完整的面向对象软件系统，在组装过程中同时进行测试。

2 UML

UML（统一建模语言）：是一种可视化的建模语言，而非程序设计语言，支持从需求分析开始的软件开发的全过程。

从总体上来看，UML 的结构包括构造块、规则和公共机制三个部分。

(1) 构造块。UML 有三种基本的构造块，分别是事物(thing)、关系(relationship)和图(diagram)。事物是 UML 的重要组成部分，关系把事物紧密联系在一起，图是多个相互关联的事物的集合。

(2) 公共机制。公共机制是指达到特定目标的公共 UML 方法。

(3) 规则。规则是构造块如何放在一起的规定。

2.1 事物

结构事物：模型的静态部分，如类、接口、用例、构件等；

行为事物：模型的动态部分，如交互、活动、状态机；

分组事物：模型的组织部分，如包；

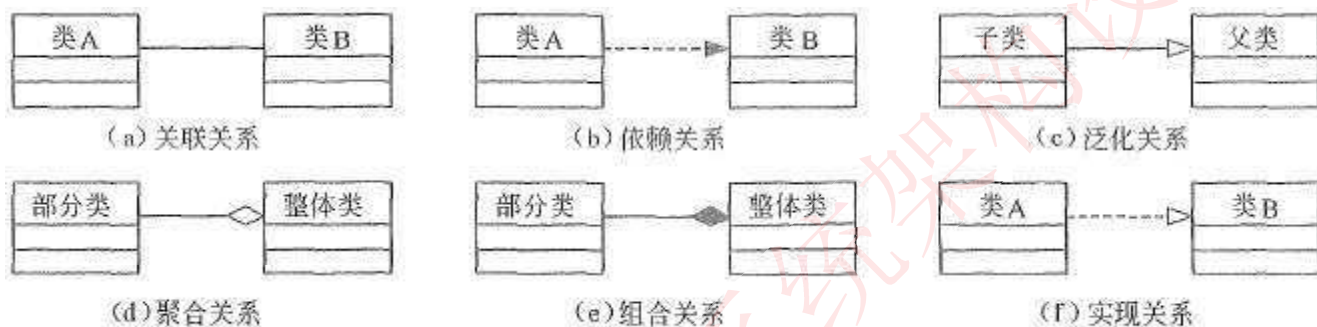
注释事物：模型的解释部分，依附于一个元素或一组元素之上对其进行约束或解释的简单符号。

2.2 关系

依赖：一个事物的语义依赖于另一个事物的语义的变化而变化

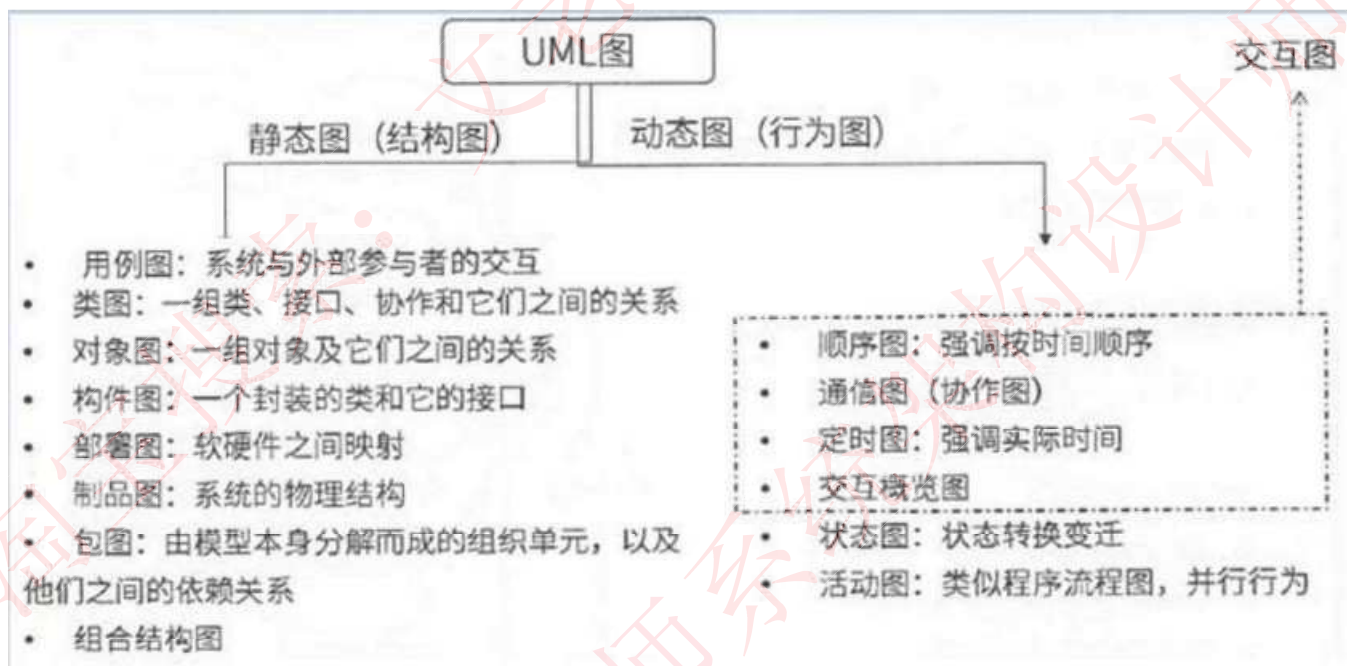
关联：是一种结构关系，描述了一组链，链是对象之间的连接。分为组合和聚合，都是部分和整体的关系，其中组合事物之间关系更强。两个类之间的关联，实际上是两个类所扮演角色的关联，因此，两个类之间可以有多个由不同角色标识的关联。

泛化：一般/特殊的关系，子类和父类之间的关系
实现：一个类元指定了另一个类元保证执行的契约。
 关系 UML 图形代号如下：



2.3 图

UML2.0 图，书上是 13 种，实际还包含制品图，一共 14 种，了解即可，总分类如下：



类图：静态图，为系统的静态设计视图，展现一组对象、接口、协作和它们之间的关系。UML 类图如下：

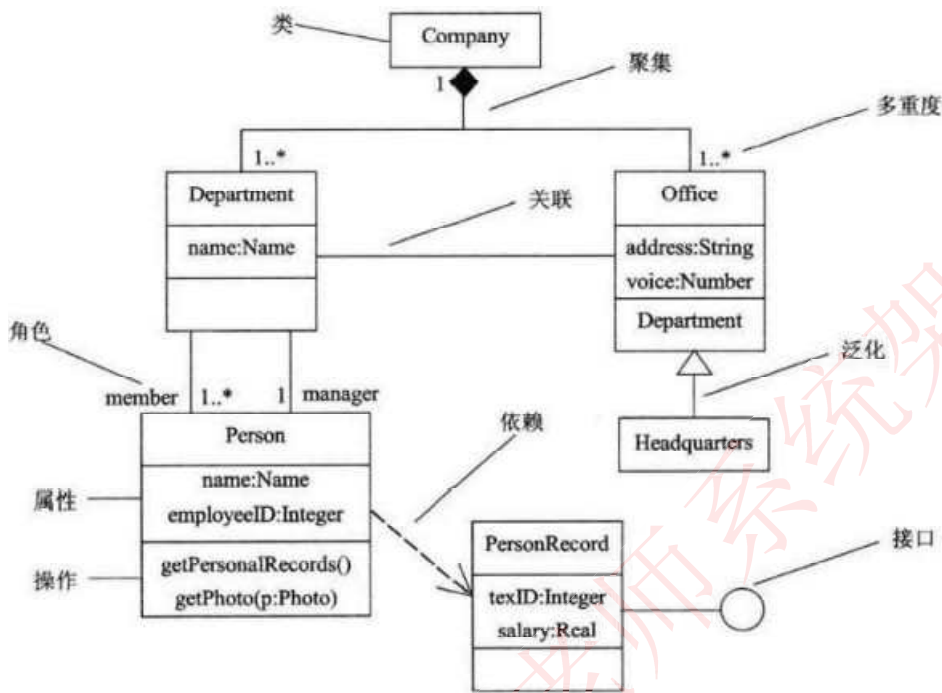
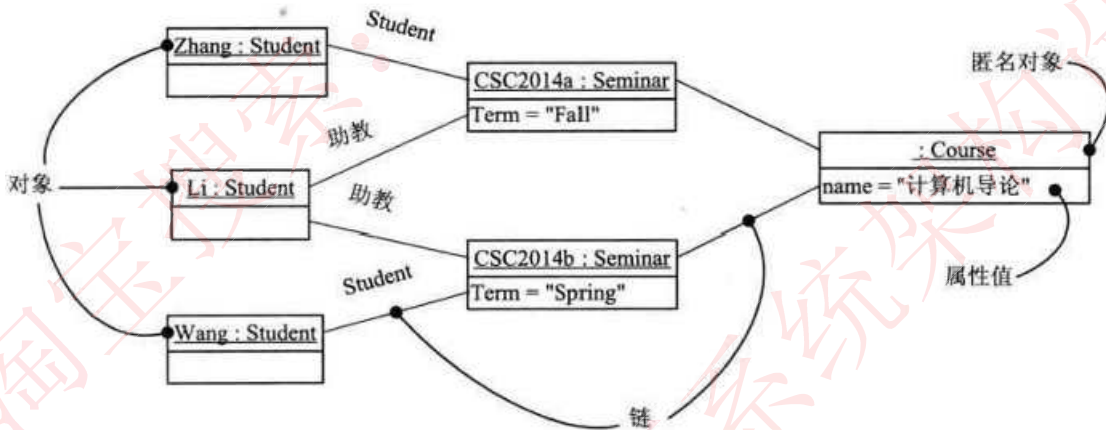
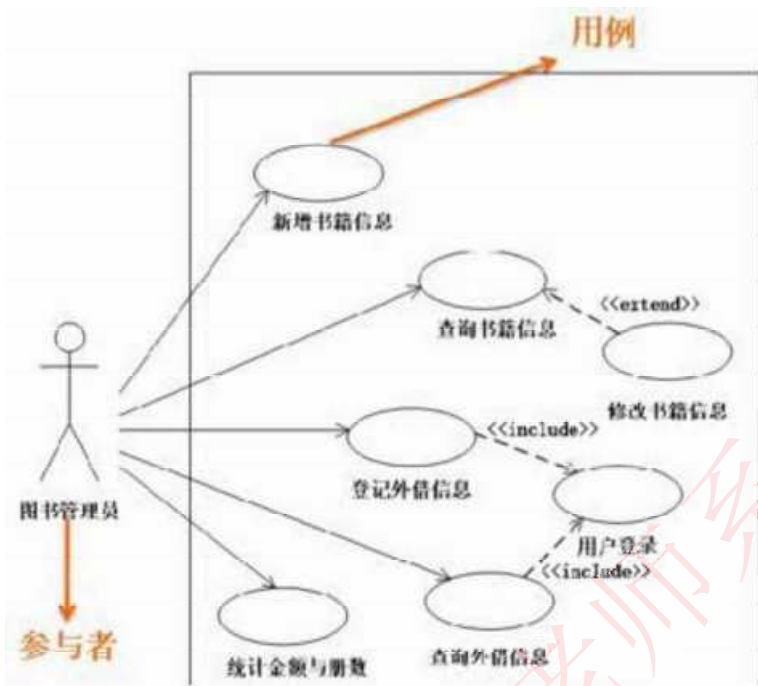


图 10-11 UML 类图

对象图：静态图，展现某一时刻一组对象及它们之间的关系，为类图的某一快照。在没有类图的前提下，对象图就是静态设计视图。如下：

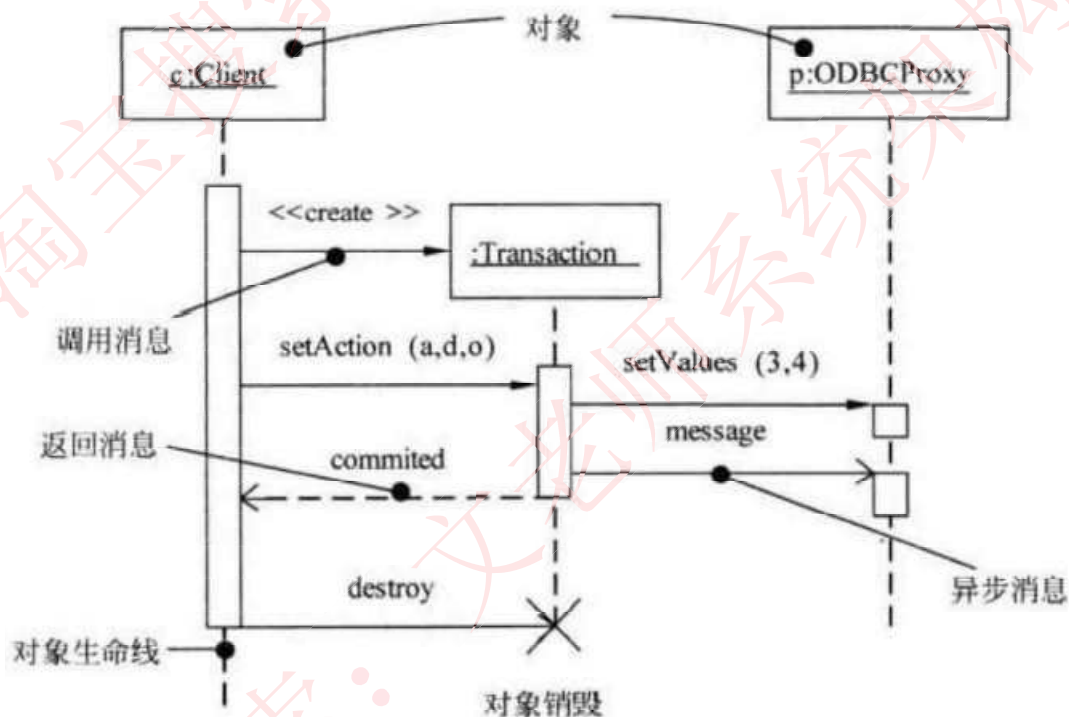


用例图：静态图，展现了一组用例、参与者以及它们之间的关系。用例图中的参与者是人、硬件或其他系统可以扮演的角色；用例是参与者完成的一系列操作，用例之间的关系有扩展、包含、泛化。如下：

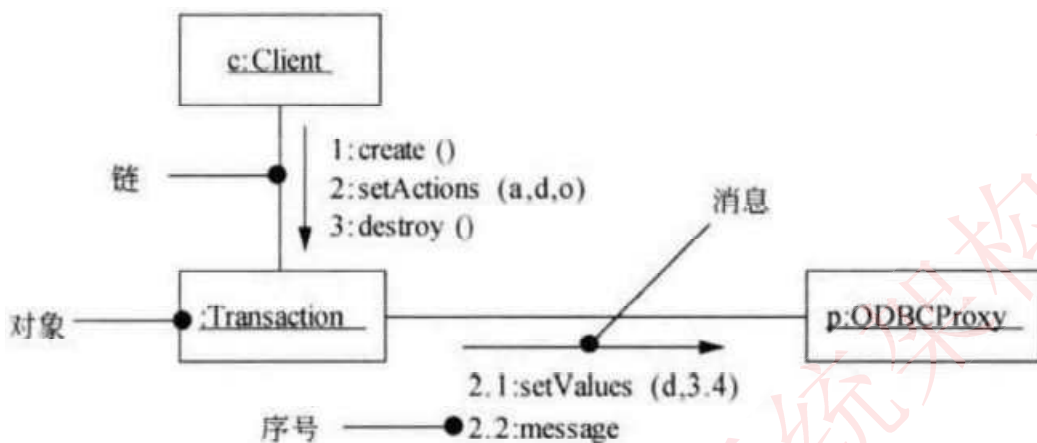


序列图：即顺序图，动态图，是场景的图形化表示，描述了以时间顺序组织的对象之间的交互活动。有**同步消息**（进行阻塞调用，调用者中止执行，等待控制权返回，需要等待返回消息，用实心三角箭头表示），**异步消息**（发出消息后继续执行，不引起调用者阻塞，也不等待返回消息，由空心箭头表示）、**返回消息**（由从右到左的虚线箭头表示）三种。如下：

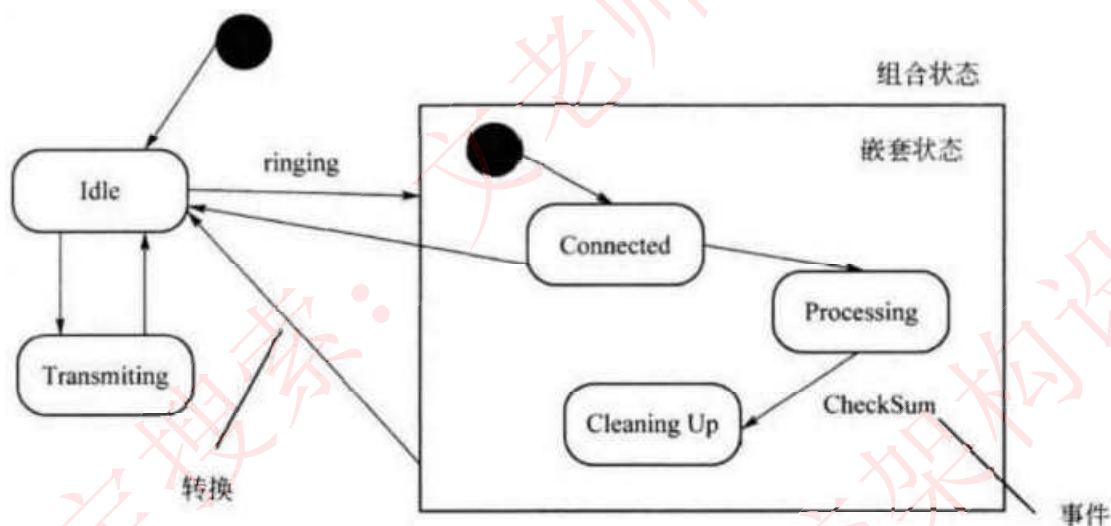
上方的对象对应下方箭头上的的成员和方法。



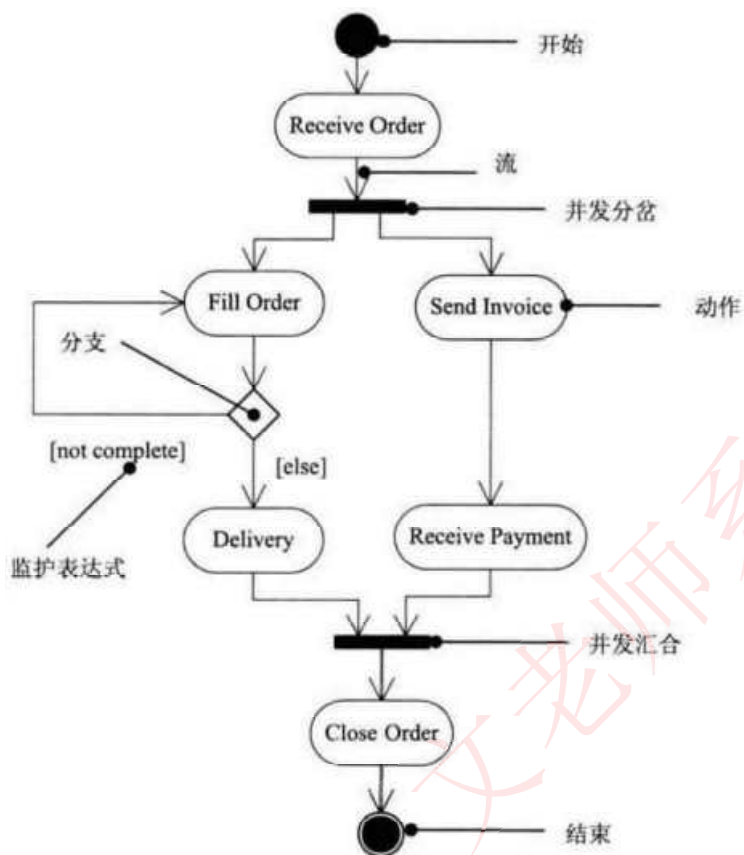
通信图：动态图，即协作图，强调参加交互的对象的组织。如下：



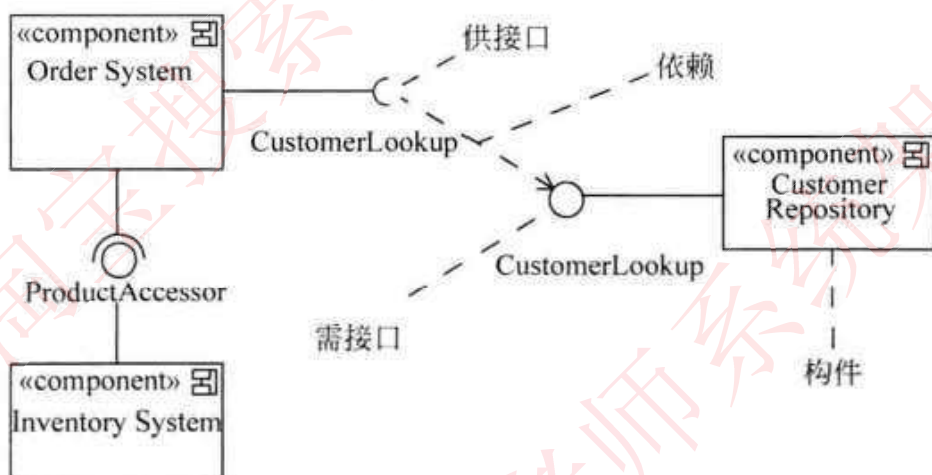
状态图：动态图，展现了一个状态机，描述单个对象在多个用例中的行为，包括简单状态和组合状态。转换可以通过事件触发器触发，事件触发后相应的监护条件会进行检查。状态图中转换和状态是两个独立的概念，如下：图中方框代表状态，箭头上的代表触发事件，实心圆点为起点和终点。



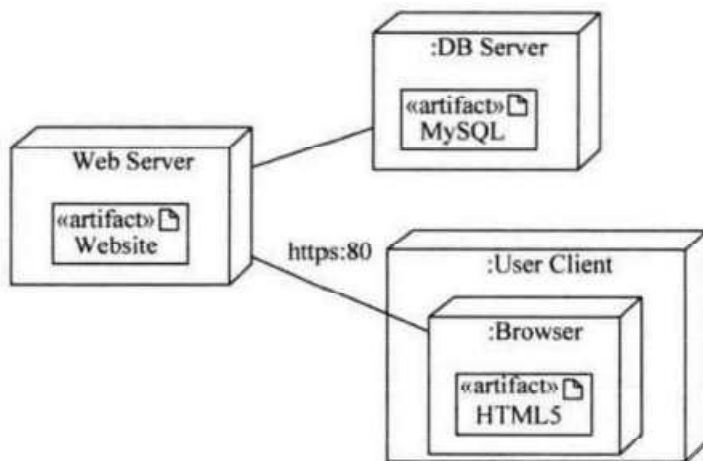
活动图：动态图，是一种特殊的状态图，展现了在系统内从一个活动到另一个活动的流程。活动的分岔和汇合线是一条水平粗线。牢记下图中并发分岔、并发汇合、监护表达式、分支、流等名词及含义。每个分岔的分支数代表了可同时运行的线程数。活动图中能够并行执行的是在一个分岔粗线下的分支上的活动，如下：



构件图（组件图）：静态图，为系统静态实现视图，展现了一组构件之间的组织和依赖。如下：

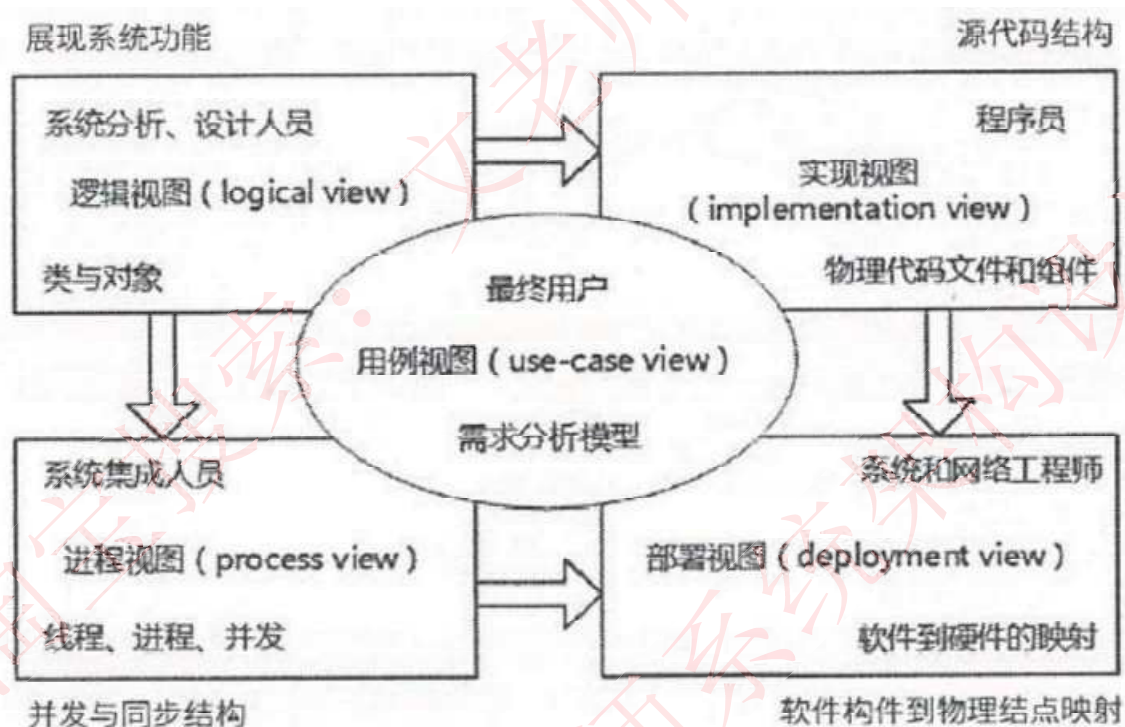


部署图：静态图，为系统静态部署视图，部署图物理模块的节点分布。它与构件图相关，通常一个结点包含一个或多个构件。其依赖关系类似于包依赖，因此部署组件之间的依赖是单向的类似于包含关系。如下：



UML 5 种视图

如下图所示：



(1) 逻辑视图。逻辑视图也称为设计视图，它表示了设计模型中在架构方面具有重要意义的一部分，即类、子系统、包和用例实现的子集。

(2) 进程视图。进程视图是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，描述了并发与同步结构。

(3) 实现视图。实现视图对组成基于系统的物理代码的文件和构件进行建模。

(4) 部署视图。部署视图把构件部署到一组物理节点上，表示软件到硬件的映射和分布结构。

(5) 用例视图。用例视图是最基本的需求分析模型。

3 SysML

SysML 是一种通用图形建模语言，用于指定，分析，设计和验证可能包括硬件，软件，信息，人员，程序和设施的复杂系统。特别是，该语言提供了图形表示，其具有用于建模系统需求，行为，结构和参数的语义基础，用于与其他工程分析模型集成。

系统建模语言（SysML）已成为基于模型的系统工程（MBSE）应用程序的事实上的标准系统架构建模语言。SysML 是 UML 2 的一种方言，它扩展了软件密集型应用程序的统一建模语言（UML）标准，使其可以成功应用于系统工程应用程序。

其实提到 SysML 不得不提另外一个概念：基于模型的系统工程（MBSE）。又称基于模型的系统开发（MBSDE），是一种系统工程过程范例，强调严格的体系结构建模原则和最佳实践应用于整个系统开发生命周期（SDLC）中的系统工程活动。这些系统工程活动包括但不限于需求分析，功能分析，性能分析，系统设计，贸易研究，系统架构规范以及系统验证和验证。

与 UML 相比，SysML 为系统工程师提供了以下优于指定系统的优势：

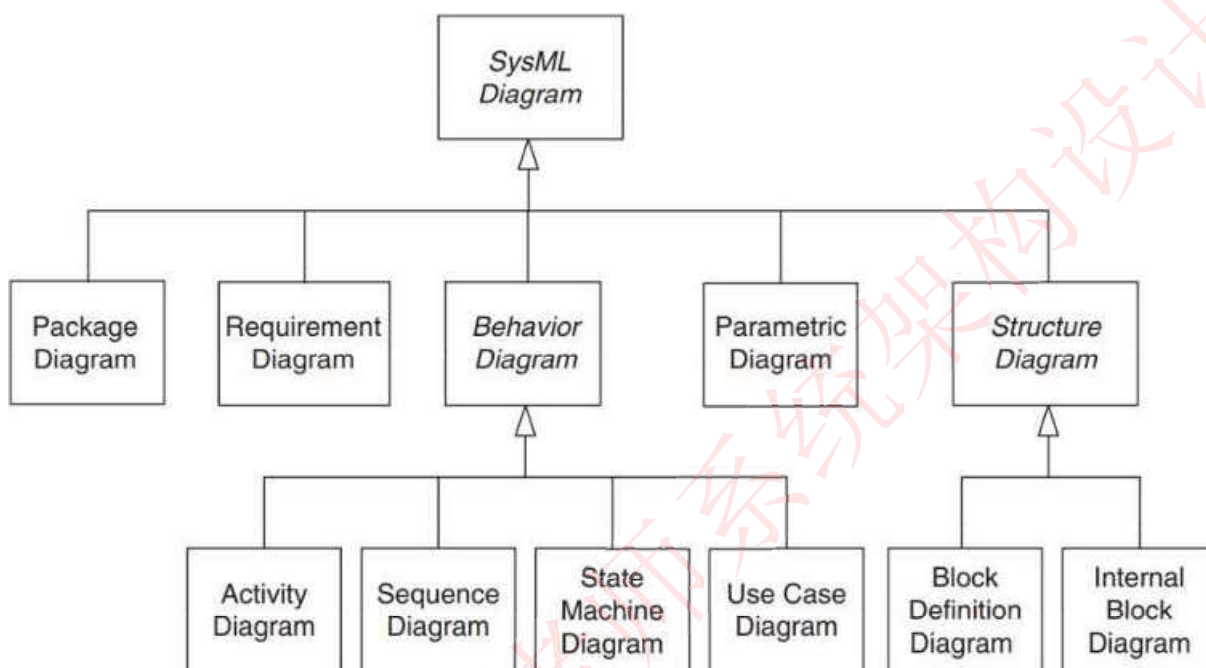
SysML 比 UML 更好地表达系统工程语义（符号解释）。它减少了 UML 的软件偏差，并为需求管理和性能分析添加了两种新的图表类型：需求图和参数图。

SysML 比 UML 更小，更容易学习。由于 SysML 删除了许多以软件为中心和无偿的构造，因此在图表类型（9 对 13）和总结构中测量的整体语言较小。

SysML 模型管理构造支持模型，视图和视点的规范，这些规范在架构上与 IEEE-Std-1471-2000（IEEE 推荐的软件密集型系统架构描述实践）保持一致。

SysML 和 UML 间存在交集，即 SysML 语言中的部分图是和 UML 中的相应图是一致的，例如用例图。同时，SysML 也有基于 UML 扩展而来的图，例如活动图。另外，还有一部分图是 SysML 所特有的，这些图与 UML 间没有关系，例如需求图。

SysML 提供了 9 种图来支持用户进行系统建模。



SysML 图	UML 图	目的
活动图（ACT 或行为）：一种行为图，主要关注控制流程，以及输入转化为输出的过程。	活动图：对系统中任何位置的流进行建模。特别是，描述正常用户交互以及替代和例外的用例中的流程由这些活动图很好地建模。	[行为图]活动图显示了作为控制和数据流的系统行为。 用于功能分析。 比较流程图（FBD）和扩展功能流程图（EFFBD），已经在系统工程师中常用。
模块定义图（BDD 或 bdd）：一种结构图，与内部模块图及参数图互补，用于描述系统的层次以及系统/组件的分类。	类图：类图表示类，它们的定义和关系问题空间中的类和实体也是解决方案空间中的详细技术实体。定义类的属性和操作包含在此类图中。类图中的关系说明了类如何与其他类交互，协作和继承。类还可以表示关系表，用户界面和控制器。	[结构图]模块定义图将系统结构显示为组件及其属性，操作和关系。 对系统分析和设计很有用。
内部模块图（IBD 或 ibd）：一种结构图，与模块定义图及参数图互补，通过组件（Parts）、端口、连接器来用于描述系统模块的内部结构。	复合结构图：在运行时模拟组件或对象行为，显示系统执行期间组件的布局，关系和实例	[结构图] 内部模块图显示了系统组件的内部结构，包括其部件和连接器。 对系统分析和设计很有用。

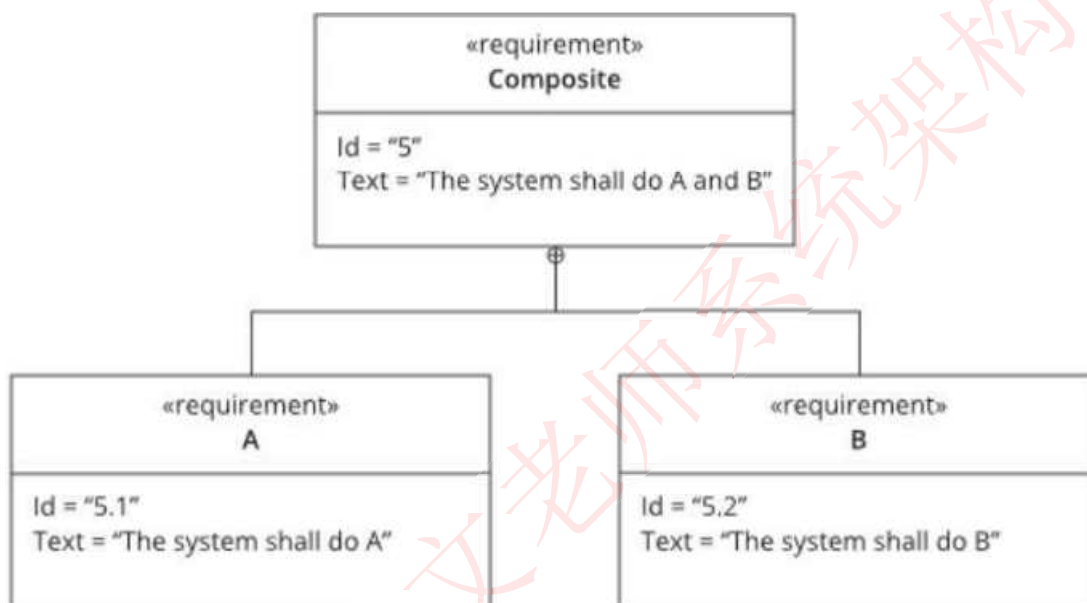
包图 (PKG 或 pkg): 一种结构图,以包的形式组织模型间的层级关系。	包图: 表示系统组织的子系统和区域。它还可以模拟包之间的依赖关系,并帮助将业务实体与用户界面,数据库,安全性和管理包分开。	结构图]包图显示了如何将模型组织到包,视图和视点中。对模型管理很有用。
参数图 (PAR 或 par): SysML 特有的图,与模块定义图及参数图互补,用于说明系统的约束。		[结构图]封装图显示结构元素之间的参数约束。对性能和定量分析很有用。
需求图 (REQ 或 req): 用于表述文字化的需求、需求间的关系,以及与之存在满足、验证等关系的其他模型元素。 SysML 还包含了分配关系的表述,包括功能到组件的分配、软件到硬件的分配以及逻辑到物理的分配		[需求图]需求图显示了系统需求及其与其他元素的关系。适用于需求工程,包括需求验证和验证 (V&V)。
序列图 (SD 或 SD): 一种行为图,主要关注并精确描述系统内部不同模块间的交互,	序列图: 根据对象的时间轴模拟对象之间的交互。对象可以在这些图上具体显示,也可以是属于类的匿名对象。	[行为图]序列图显示系统行为作为系统组件之间的交互。对系统分析和设计很有用。
状态机图 (STM 或 stm): 一种行为图,主要关注系统内部模块的一系列状态以及在事件触发下的不同状态间的转换。	状态机图: 显示内存中对象的运行时生命周期。这样的生命周期包括对象的所有状态以及状态改变的条件。	行为图]状态机图将系统行为显示为组件或交互响应事件时所经历的状态序列。对系统设计和模拟/代码生成很有用。
用例图 (UC 或 uc): 一种黑盒视图,是系统功能的高层描述,用于表达系统执行用例以及引起系统执行行为的参与者。	用例图: 从用户的角度提供系统或业务流程功能的概述。用户“使用”系统的方式是创建用例图的起点。	[行为图]用例图将系统功能需求显示为对系统用户有意义的事务。用于指定功能要求。(注意潜在的语义重叠与需求图中指定的功能需求。)

需求图: 标准 SysML 需求包括用于指定其唯一性的标识符和需求文本两个属性,如图所示。其他属性(例如,验证状态,优先级等)也可以由用户指定。

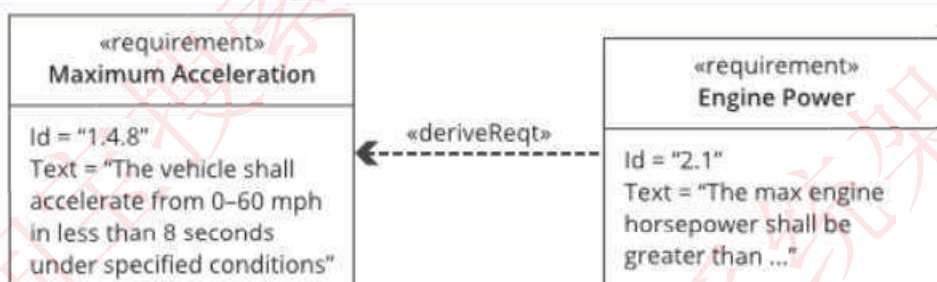
SysML 规定了七个需求关系

1) 复合关系

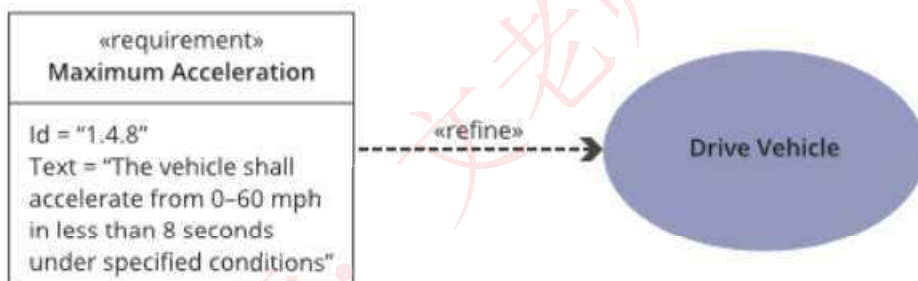
复合需求可以包含需求层次结构中的子需求。复合需求可以声明系统应执行 **A** 和 **B**，可以将其分解为系统应执行 **A** 和系统应进行 **B** 的子需求



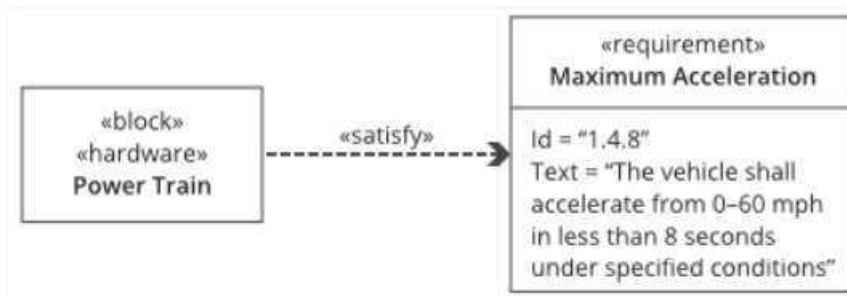
2) 派生关系：派生的需求通常对应于系统层次结构下一级的需求。一个简单的例子是车辆加速需求，该需求被分析以导出发动机动力等方面的需求



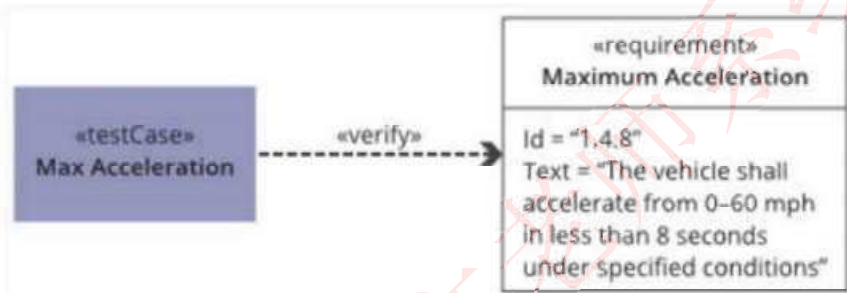
3) 细化关系：细化关系可用于描述如何使用模型元素或元素集进一步细化需求。例如，可以使用用例或活动图来细化基于文本的功能需求



4) 满足关系：满足关系描述了设计或实现模型如何满足一个或多个需求。然后，系统建模者可以指定旨在满足要求的系统设计元素



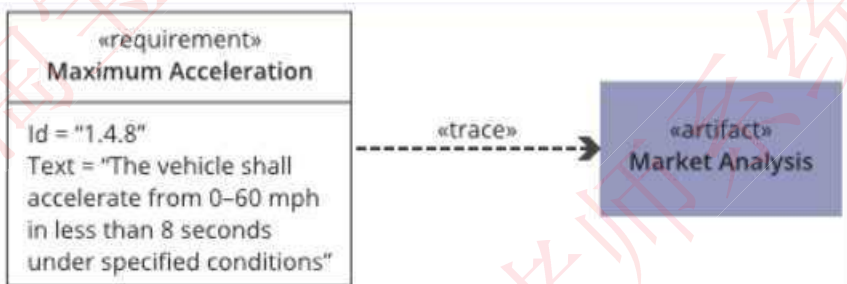
5) 验证关系：验证关系定义了测试用例或其他模型元素如何验证需求。在 SysML 中，测试用例或其他元素可以用作表示任何标准检验方法的通用机制，分析，演示或测试。



6) 复制关系：真正需要跨产品系列和项目重用需求。典型的方案是适用于产品和/或产品系列中重复使用的项目和要求的法定法规或合同要求。



7) 追溯关系：追溯关系提供了需求和任何其他模型元素之间的通用关系。追溯关系对于将需求与源文档相关联或在规范树中的规范之间建立关系可能很有用。

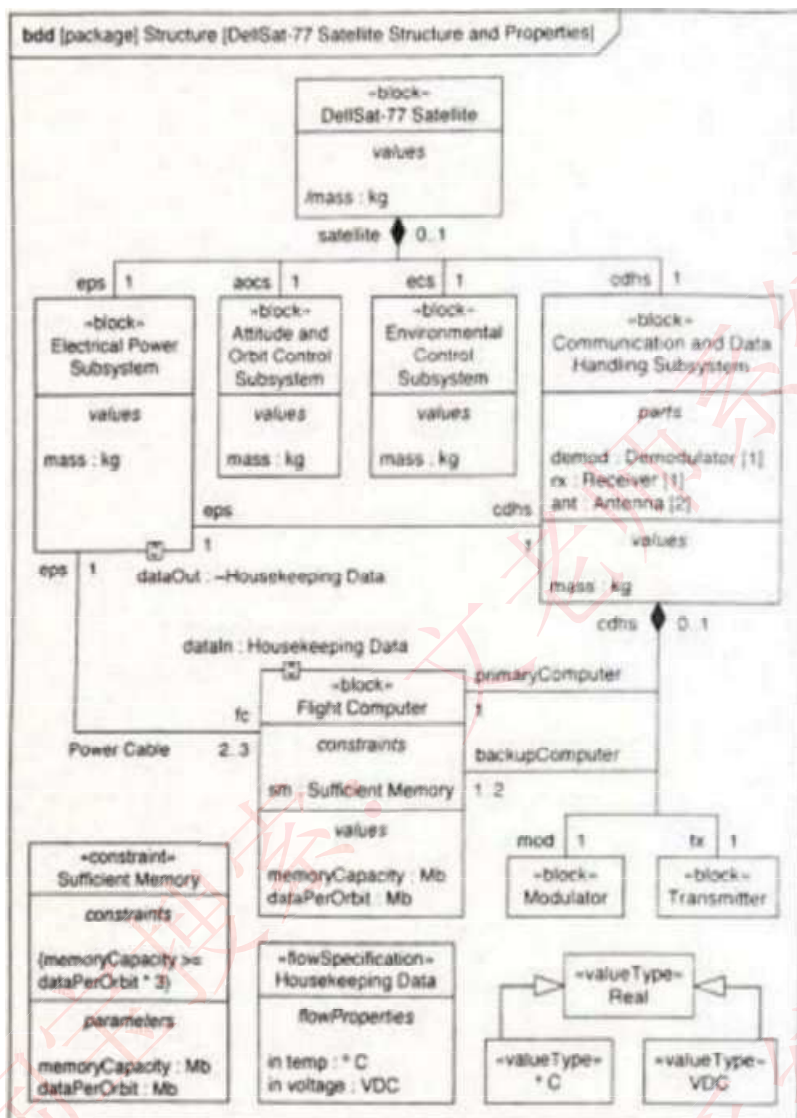


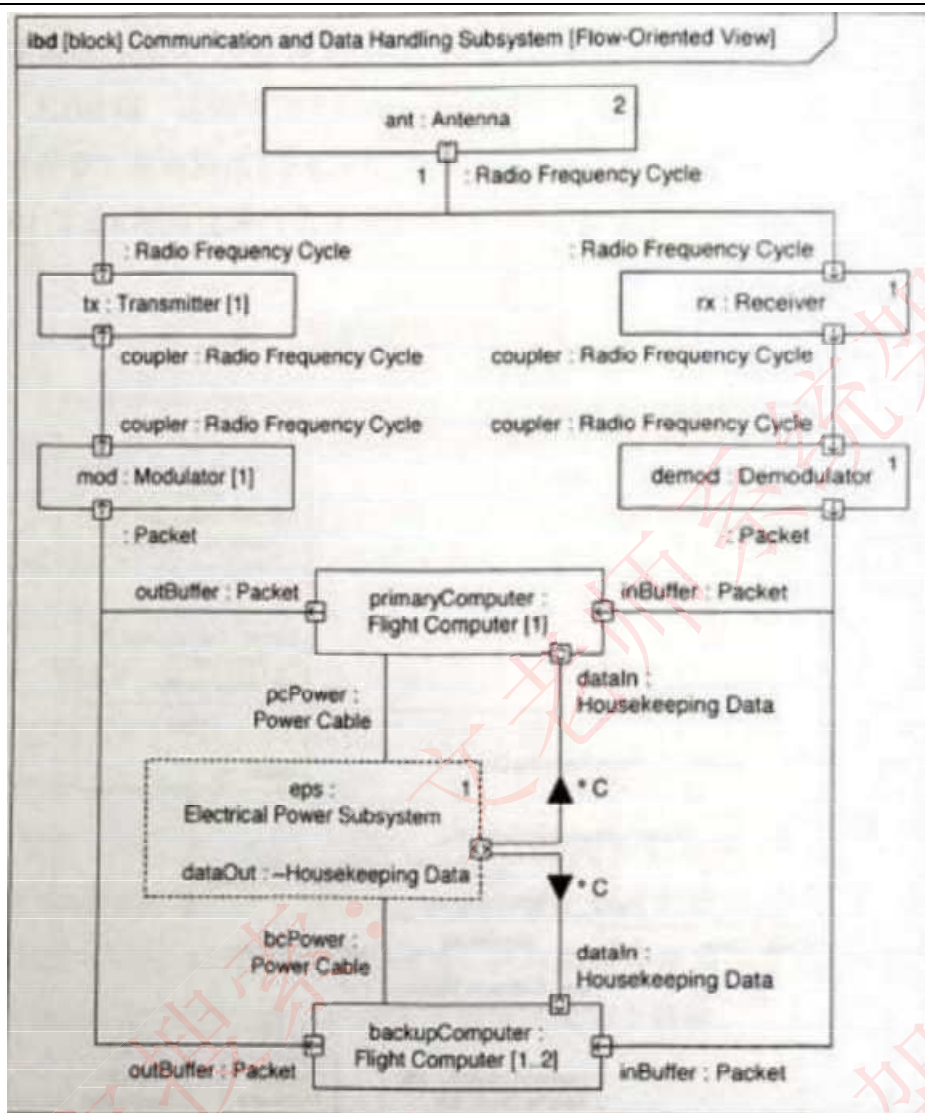
模块定义图 BDD 和内部模块图 IBD

创建 IBD 是为了指定单个模块的内部结构。和 BDD 一样，IBD 是系统或者系统一个组成部分的静态(结构化)视图。和 BDD 不同的是，IBD 不会显示模块，它会显示对模块的使用——也就是在 IBD 头部命名的模块的组成部分属性和引用属性。

在 BDD 中也可以显示组成部分属性和引用属性——或者是作为模块分隔框中的字符串，或者是作为关联一端的角色名称。但是 IBD 可以表达在 BDD 中无法表达的信息：组成部分属性和引用属性之间的连接；在连接之间 流动的事件、能量和数据的类型；以及通过连接提供和请求的服务。

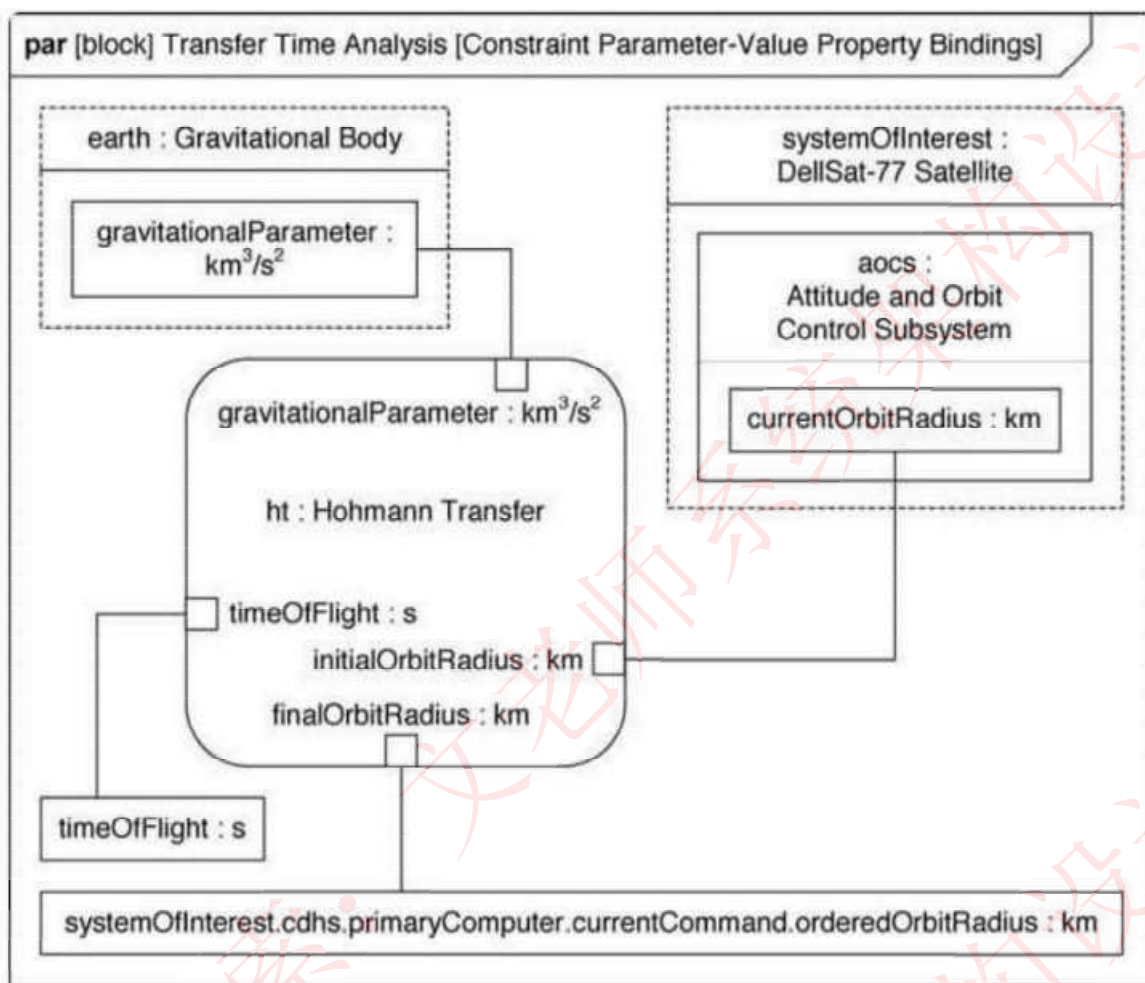
IBD 会表达模块的组成部分必须如何组合才能够创建有效实例。它还会显示模块的实例必须如何与外部实体(引用属性)连接，以在整体上创建系统的有效实例。





参数图

SysML 参数图是一种特殊的内部块图。和 IBD 一样，参数图会显示模块的内部结构，但是其关注点在于值属性和约束参数之间的绑定关系。因此，可以说，参数图和块定义图提供了模块的互补视图。



十二 项目管理

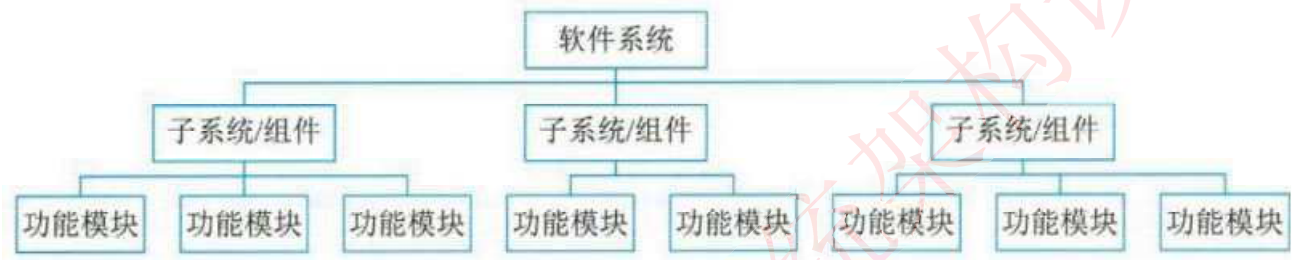
1 进度管理

进度管理就是采用科学的方法，确定进度目标，编制进度计划和资源供应计划，进行进度控制，在与质量、成本目标协调的基础上，实现工期目标。

具体来说，包括以下过程：

- (1) 活动定义：确定完成项目各项可交付成果而需要开展的具体活动。
- (2) 活动排序：识别和记录各项活动之间的先后关系和逻辑关系。
- (3) 活动资源估算：估算完成各项活动所需要的资源类型和效益。
- (4) 活动历时估算：估算完成各项活动所需要的具体时间。
- (5) 进度计划编制：分析活动顺序、活动持续时间、资源要求和进度制约因素，制订项目进度计划。
- (6) 进度控制：根据进度计划开展项目活动，如果发现偏差，则分析原因或进行调整。

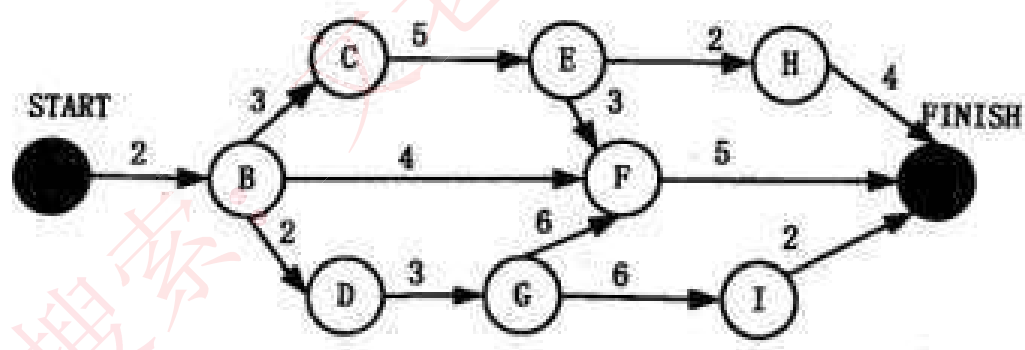
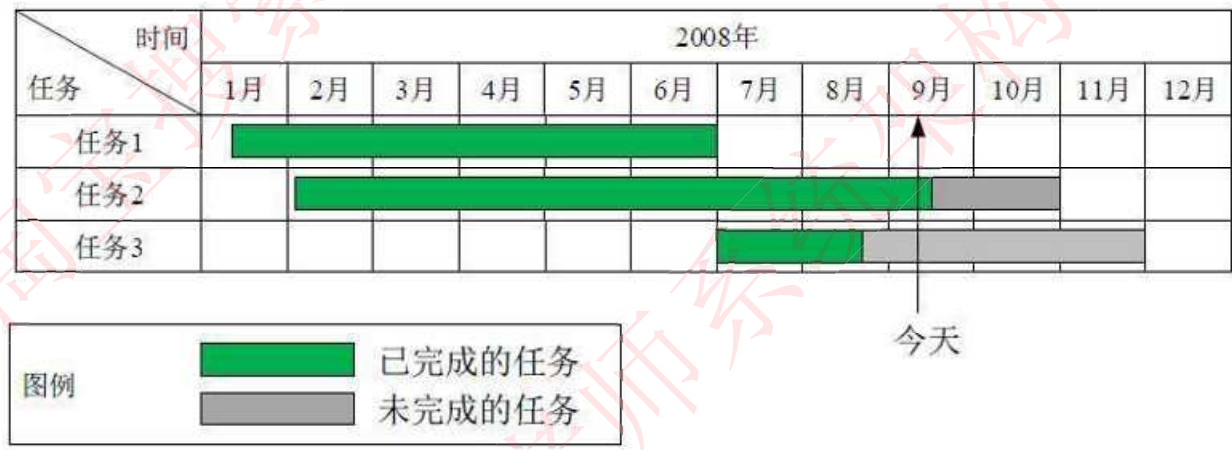
软件项目往往是比较大而复杂的，往往需要进行层层分解，将大的任务分解成一个个的单一小任务进行处理。工作分解结构（WBS）如图所示，就是把一个项目，按一定的原则分解成任务，任务再分解成一项项工作，再把一项项工作分配到每个人的日常活动中，直到分解不下去为止。



WBS 常见的分解方式包括：按产品的物理结构分解、按产品或项目的功能分解、按照实施过程分解、按照项目的实施单位分解、按照项目的目标分解、按部分或职能进行分解等。不管采用哪种分解方式，最终都要满足以下对任务分解的基本要求。

- (1) WBS 的工作包是可控和可管理的，不能过于复杂。
- (2) 任务分解也不能过细，一般原则 WBS 的树形结构不超过 6 层。
- (3) 每个工作包要有一个交付成果。
- (4) 每个任务必须有明确定义的完成标准。
- (5) WBS 必须有利于责任分配。

进度安排的常用图形描述方法有 Gantt 图（甘特图）和项目计划评审技术（Program Evaluation & Review Technique, PERT）图。



关键路径法

关键路径：是项目的最短工期，但却是从开始到结束时间最长的路径。进度网络图中可能有多条关键路径，因为活动会变化，因此关键路径也在不断变化中。

关键活动：关键路径上的活动，最早开始时间=最晚开始时间。

通常，每个节点的活动会有如下几个时间：

- (1) 最早开始时间 (ES)，某项活动能够开始的最早时间。
 - (2) 最早结束时间 (EF)，某项活动能够完成的最早时间。 $EF=ES+工期$
 - (3) 最迟结束时间 (LF)。为了使项目按时完成，某项活动必须完成的最迟时间。
 - (4) 最迟开始时间 (LS)。为了使项目按时完成，某项活动必须开始的最迟时间。 $LS=LF-工期$
- 这几个时间通常作为每个节点的组成部分，如图所示：

最早开始时间	工期	最早完成时间
活动名称		
最迟开始时间	总浮动时间	最迟完成时间

顺推：最早开始 ES=所有前置活动最早完成 EF 的最大值；最早完成 EF=最早开始 ES+持续时间。

逆推：最晚完成 LF=所有后续活动最晚开始 LS 的最小值；最晚开始 LS=最晚完成 LF-持续时间。



总浮动时间（松弛时间）：在不延误项目完工时间且不违反进度制约因素的前提下，活动可以从最早开始时间推迟或拖延的时间量，就是该活动的进度灵活性。正常情况下，关键活动的总浮动时间为零。

总浮动时间=最迟开始 LS-最早开始 ES 或 最迟完成 LF-最早完成 EF 或 关键路径-非关键路径时长。

自由浮动时间：是指在不延误任何紧后活动的最早开始时间且不违反进度制约因素的前提下，活动可以从最早开始时间推迟或拖延的时间量。

自由浮动时间=紧后活动最早开始时间的最小值-本活动的最早完成时间。

2 软件配置管理

配置管理是为了系统地控制配置变更，在系统的整个生命周期中维持配置的完整性和可跟踪性，而标识系统在不同时间点上配置的学科。在 GB/T11457-2006 中将“配置管理”正式定义为：“应用技术的和管理的指导和监控方法以标识和说明配置项的功能和物理特征，控制这些特征的变更，记录和报告变更处理和实现状态并验证与规定的需求的遵循性。”

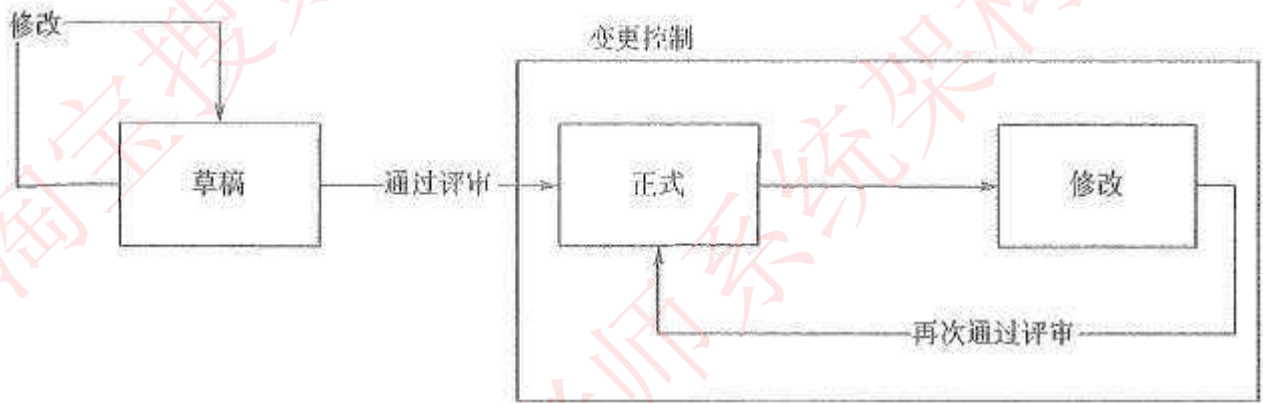
配置项：GB/T11457-2006 对配置项的定义为：“为配置管理设计的硬件、软件或二者的集合，在配置管理过程中作为一个单个实体来对待”。

以下内容都可以作为配置项进行管理：外部交付的软件产品和数据、指定的内部软件工作产品和数据、指定的用于创建或支持软件产品的支持工具、供方/供应商提供的软件和客户提供的设备/软件。典型配置项包括项目计划书、需求文档、设计文档、源代码、可执行代码、测试用例、运行软件所需的各种数据，它们经评审和检查通过后进入配置管理。

配置项可以分为基线配置项和非基线配置项两类，例如，基线配置项可能包括所有的设计文档和源程序等；非基线配置项可能包括项目的各类计划和报告等。

所有配置项的操作权限应由 CMO（配置管理员）严格管理，基本原则是：基线配置项向开发人员开放读取的权限；非基线配置项向 PM、CCB 及相关人员开放。

配置项的状态可分为“草稿”“正式”和“修改”三种。配置项刚建立时，其状态为“草稿”。配置项通过评审后，其状态变为“正式”。此后若更改配置项，则其状态变为“修改”。当配置项修改完毕并重新通过评审时，其状态又变为“正式”。如图所示



配置项版本号

(1) 处于“草稿”状态的配置项的版本号格式为 0.YZ, YZ 的数字范围为 01~99。随着草稿的修正，YZ 的取值应递增。YZ 的初值和增幅由用户自己把握。

(2) 处于“正式”状态的配置项的版本号格式为 X.Y, X 为主版本号，取值范围为 1~9。Y 为次版本号，取值范围为 0~9。

配置项第一次成为“正式”文件时，版本号为 1.0。

如果配置项升级幅度比较小,可以将变动部分制作成配置项的附件,附件版本依次为 1.0, 1.1…。当附件的变动积累到一定程度时,配置项的 Y 值可适量增加, Y 值增加一定程度时, X 值将适量增加。当配置项升级幅度比较大时,才允许直接增大 X 值。

(3) 处于“修改”状态的配置项的版本号格式为 X.YZ。配置项正在修改时,一般只增大 Z 值, X.Y 值保持不变。当配置项修改完毕,状态成为“正式”时,将 Z 值设置为 0,增加 X.Y 值。参见上述规则 (2)。

配置项版本管理:在项目开发过程中,绝大部分的配置项都要经过多次的修改才能最终确定下来。对配置项的任何修改都将产生新的版本。由于我们不能保证新版本一定比旧版本“好”,所以不能抛弃旧版本。版本管理的目的是按照一定的规则保存配置项的所有版本,避免发生版本丢失或混淆等现象,并且可以快速准确地查找到配置项的任何版本。

配置基线(常简称为基线)由一组配置项组成,这些配置项构成一个相对稳定的逻辑实体。基线中的配置项被“冻结”了,不能再被任何人随意修改。对基线的变更必须遵循正式的变更控制程序。

3 质量管理

质量是软件产品特性的综合,表示软件产品满足明确(基本需求)或隐含(期望需求)要求的能力。质量管理是指确定质量方针、目标和职责,并通过质量体系中的质量计划、质量控制、质量保证和质量改进来使其实现的所有管理职能的全部活动;

主要包括以下过程:

(1) 质量规划:识别项目及其产品的质量要求和标准,并书面描述项目将如何达到这些要求和标准的过程。

(2) 质量保证:一般是每隔一定时间(例如,每个阶段末)进行的,主要通过系统的质量审计(软件评审)和过程分析来保证项目的质量。

(3) 质量控制:实时监控项目的具体结果,以判断它们是否符合相关质量标准,制订有效方案,以消除产生质量问题的原因。

从管理角度出发,可以将影响软件质量的因素划分为 3 组,分别反映用户在使用软件产品时的 3 种不同倾向和观点。这 3 组分别是:产品运行、产品修改和产品转移。

软件质量保证(SQA)的关注点集中在于一开始就避免缺陷的产生。质量保证的主要目标是:

(1) 事前预防工作,例如,着重于缺陷预防而不是缺陷检查。

(2) 尽量在刚刚引入缺陷时即将其捕获,而不是让缺陷扩散到下一个阶段。

(3) 作用于过程而不是最终产品,因此它有可能会带来广泛的影响与巨大的收益。

(4) 贯穿于所有的活动之中,而不是只集中于一点。

软件质量保证的主要任务:SQA 审计与评审、SQA 报告、处理不符合问题。

4 风险管理

风险管理就是要对项目风险进行认真的分析和科学的管理，这样，是能够避开不利条件、少受损失、取得预期的结果并实现项目目标的，能够争取避免风险的发生或尽量减小风险发生后的影响。但是，完全避开或消除风险，或者只享受权益而不承担风险是不可能的。

风险管理计划编制：如何安排与实施项目的风险管理，制定下列各步的计划。

风险识别：识别出项目中已知和可预测的风险，确定风险的来源、产生的条件、描述风险的特征以及哪些项目可以产生风险，形成一个风险列表。

风险定性分析：对已经识别的风险进行排序，确定风险可能性与影响、确定风险优先级、确定风险类型。

风险定量分析：进一步了解风险发生的可能性具体由多大，后果具体由多严重。包括灵敏度分析、期望货币价值分析、决策树分析、蒙特卡罗模拟。

风险应对计划编制：对每一个识别出来的风险来分别制定应对措施，这些措施组成的文档称为风险应对计划。包括消极风险（避免策略、转移策略、减轻策略）；积极风险（开拓、分享、强大）。

风险监控：监控风险计划的执行，检测残余风险，识别新的风险，保证风险计划的执行，并评价这些计划对减少风险的有效性。

项目风险：作用于项目上的不确定的事件或条件，既可能产生威胁，也可能带来机会。

通过积极和合理的规划，超过 90% 的风险都可以进行提前应对和管理。

风险应该尽早识别出来，高层次风险应记录在章程里。

应由对风险最有控制力的一方承担相应的风险。

承担风险程度与所得回报相匹配原则，承担的风险要有上限。

在信息系统项目中，从宏观上来看，风险可以分为项目风险、技术风险和商业风险。

项目风险是指潜在的预算、进度、个人（包括人员和组织）、资源、用户和需求方面的问题，以及它们对项目的影响。项目复杂性、规模和结构的不确定性也构成项目的（估算）风险因素。项目风险威胁到项目计划，一旦项目风险成为现实，可能会拖延项目进度，增加项目的成本。

技术风险是指潜在的设计、实现、接口、测试和维护方面的问题。此外，规格说明的多义性、技术上的不确定性、技术陈旧、最新技术（不成熟）也是风险因素。技术风险威胁到待开发系统的质量和预定的交付时间。如果技术风险成为现实，开发工作可能会变得很困难或根本不可能。

商业风险威胁到待开发系统的生存能力，主要有以下 5 种不同的商业风险：

- （1）市场风险。开发的系统虽然很优秀但不是市场真正所想要的。
- （2）策略风险。开发的系统不再符合企业的信息系统战略。
- （3）销售风险。开发了销售部门不清楚如何推销的系统。
- （4）管理风险。由于重点转移或人员变动而失去上级管理部门的支持。
- （5）预算风险。开发过程没有得到预算或人员的保证。

十三 系统架构设计

1 软件架构的概念

一个程序和计算系统软件体系结构是指系统的一个或者多个结构。结构中包括软件的构件，构件的外部可见属性以及它们之间的相互关系。

体系结构并非可运行软件。确切地说，它是一种表达，使软件工程师能够：

- (1) 分析设计在满足所规定的需求方面的有效性；
- (2) 在设计变更相对容易的阶段，考虑体系结构可能的选择方案；
- (3) 降低与软件构造相关联的风险。

上面的定义强调在任意体系结构的表述中“软件构件”的角色。软件构件简单到可以是程序模块或者面向对象的类，也可以扩充到包含数据库和能够完成客户与服务器网络配置的“中间件”。

软件体系结构的设计两个层次：数据设计和体系结构设计。数据设计体现传统系统中体系结构的数据构件和面向对象系统中类的定义（封装了属性和操作），体系结构设计则主要关注软件构件的结构、属性和交互作用。

软件架构设计与生命周期

1. 需求分析阶段。需求分析和 SA 设计面临的是不同的对象：一个是问题空间；另一个是解空间。从软件需求模型向 SA 模型的转换主要关注两个问题：如何根据需求模型构建 SA 模型。如何保证模型转换的可追踪性。

2. 设计阶段。是 SA 研究关注的最早和最多的阶段，这一阶段的 SA 研究主要包括：SA 模型的描述、SA 模型的设计与分析方法，以及对 SA 设计经验的总结与复用等。有关 SA 模型描述的研究分为 3 个层次：SA 的基本概念（构件和连接子）、体系结构描述语言 ADL、SA 模型的多视图表示。

3. 实现阶段。最初 SA 研究往往只关注较高层次的系统设计、描述和验证。为了有效实现 SA 设计向实现的转换，实现阶段的体系结构研究表现在以下几个方面。

(1) 研究基于 SA 的开发过程支持，如项目组织结构、配置管理等。

(2) 寻求从 SA 向实现过渡的途径，如将程序设计语言元素引入 SA 阶段、模型映射、构件组装、复用中间件平台等。

(3) 研究基于 SA 的测试技术。

4. 构件组装阶段。在 SA 设计模型的指导下，可复用构件的组装可以在较高层次上实现系统，并能够提高系统实现的效率。在构件组装的过程中，SA 设计模型起到了系统蓝图的作用。研究内容包括如下两个方面。

(1) 如何支持可复用构件的互联，即对 SA 设计模型中规约的连接子的实现提供支持。

(2) 在组装过程中，如何检测并消除体系结构失配问题。

在构件组装阶段的失配问题主要包括：由构件引起的失配、由连接子引起的失配、由于系统成分对全局体系结构的假设存在冲突引起的失配等。

5. 部署阶段。SA 对软件部署作用如下。

(1) 提供高层的体系结构视图来描述部署阶段的软硬件模型。

(2) 基于 SA 模型可以分析部署方案的质量属性，从而选择合理的部署方案。

6. 后开发阶段。是指软件部署安装之后的阶段。这一阶段的 SA 研究主要围绕维护、演化、复用等方面来进行。典型的研究方向包括动态软件体系结构、体系结构恢复与重建等。

(1) 动态软件体系结构。现实中的软件具有动态性，体系结构会在运行时发生改变。

运行时变化包括两类：软件内部执行所导致的体系结构改变；软件系统外部的请求对软件进行的重配置。

包括两个部分的研究：体系结构设计阶段的支持、运行时刻基础设施的支持。

(2) 体系结构恢复与重建。对于现有系统在开发时候没有考虑 SA 的情况，从这些系统中恢复或重购体系结构。从已有的系统中获取体系结构的重建方法分为 4 类：手工体系结构重建、工具支持的手工重建、通过查询语言来自动建立聚集、使用其他技术（如数据挖掘等）。

架构设计概述

从需求分析到软件设计之间的过渡过程称为软件架构。只要软件架构设计好了，整个软件就不会出现坍塌性的错误，即不会崩溃。

架构设计就是需求分配，将满足需求的职责分配到组件上。

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，由构件的描述、构件的相互作用（连接件）、指导构件集成的模式以及这些模式的约束组成。

软件架构不仅指定了系统的组织结构和拓扑结构，并且显示了系统需求和构件之间的对应关系，提供了一些设计决策的基本原理。

解决好软件的复用、质量和维护问题，是研究软件架构的根本目的。

软件架构设计包括提出架构模型，产生架构设计和进行设计评审等活动，是一个迭代的过程。架构设计主要关注软件组件的结构、属性和交互作用，并通过多种视图全面描述特定系统的架构。

架构设计作用

软件架构能够在设计变更相对容易的阶段，考虑系统结构的可选方案，便于技术人员与非技术人员就软件设计进行交互，能够展现软件的结构、属性与内部交互关系。

软件架构是项目干系人进行交流的手段，明确了对系统实现的约束条件，决定了开发和维护组织的组织结构，制约着系统的质量属性。

软件架构使推理和控制的更改更加简单，有助于循序渐进的原型设计，可以作为培训的基础。

软件架构是可传递和可复用的模型，通过研究软件架构可能预测软件的质量。

构件

构件是一个独立可交付的功能单元，外界通过接口访问其提供的服务。

构件由一组通常需要同时部署的原子构件组成。一个原子构件是一个模块和一组资源。原子构件是部署、版本控制和替换的基本单位。原子构件通常成组地部署，但是它也能够被单独部署。

构件和原子构件之间的区别在于，大多数原子构件永远都不会被单独部署，尽管它们可以被单独部署。相反，大多数原子构件都属于一个构件家族，一次部署往往涉及整个家族。

一个模块是不带单独资源的原子构件（在这个严格定义下，Java 包不是模块——在 Java 中部署的原子单元是类文件。

一个单独的包被编译成多个单独的类文件——每个公共类都有一个）。

模块是一组类和可能的非面向对象的结构体，比如过程或者函数。

构件和对象

构件的特性是：（1）独立部署单元；（2）作为第三方的组装单元；（3）没有（外部的）可见状态。一个构件可以包含多个类元素，但是一个类元素只能属于一个构件。将一个类拆分进行部署通常没什么意义。

对象的特性是：（1）一个实例单元，具有唯一的标志。（2）可能具有状态，此状态外部可见。（3）封装了自己的状态和行为。

构件接口

接口标准化是对接口中消息的格式、模式和协议的标准化。它不是要将接口格式化为参数化操作的集合，而是关注输入输出的消息的标准化，它强调当机器在网络中互连时，标准的消息模式、格式、协议的重要性。

面向构件的编程（COP）

关注于如何支持建立面向构件的解决方案。“面向构件的编程需要下列基本的支持：

- 多态性（可替代性）；
- 模块封装性（高层次信息的隐藏）；
- 后期的绑定和装载（部署独立性）；
- 安全性（类型和模块安全性）。”

2 基于架构的软件开发方法

ABSD 方法是架构驱动，强调由业务、质量和功能需求的组合驱动架构设计。它强调采用视角和视图来描述软件架构，采用用例和质量属性场景来描述需求。进一步来说，用例描述的是功能需求，质量属性场景描述的是质量需求（或侧重于非功能需求）。

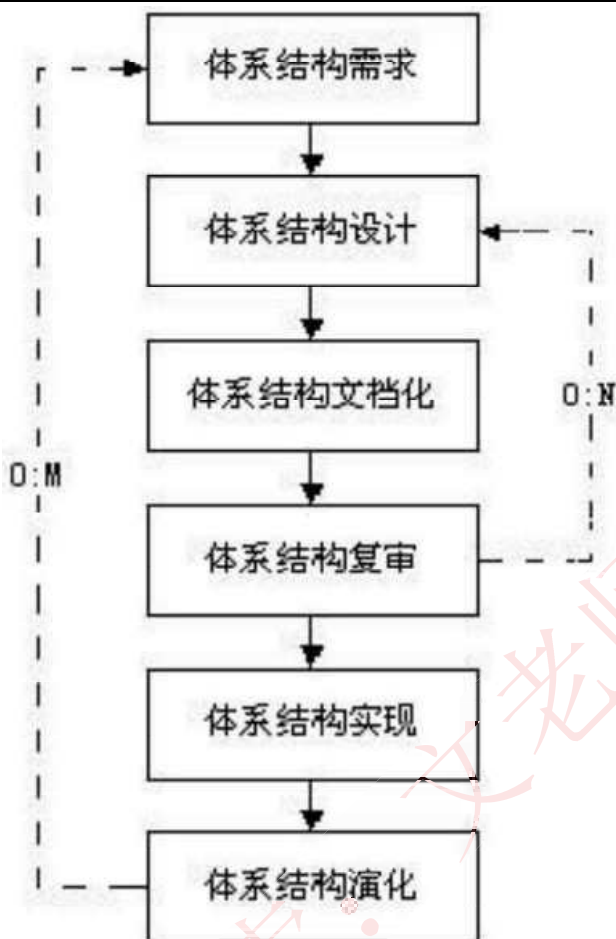
使用 ABSD 方法，设计活动可以从项目总体功能框架明确就开始，这意味着需求获取和分析还没有完成，就开始了软件设计。

ABSD 方法有三个基础。第一个基础是功能的分解，使用已有的基于模块的内聚和耦合技术；第二个基础是通过选择架构风格来实现质量和业务需求；第三个基础是软件模板的使用，软件模板利用了一些软件系统的结构。

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，有助于降低架构设计的随意性。

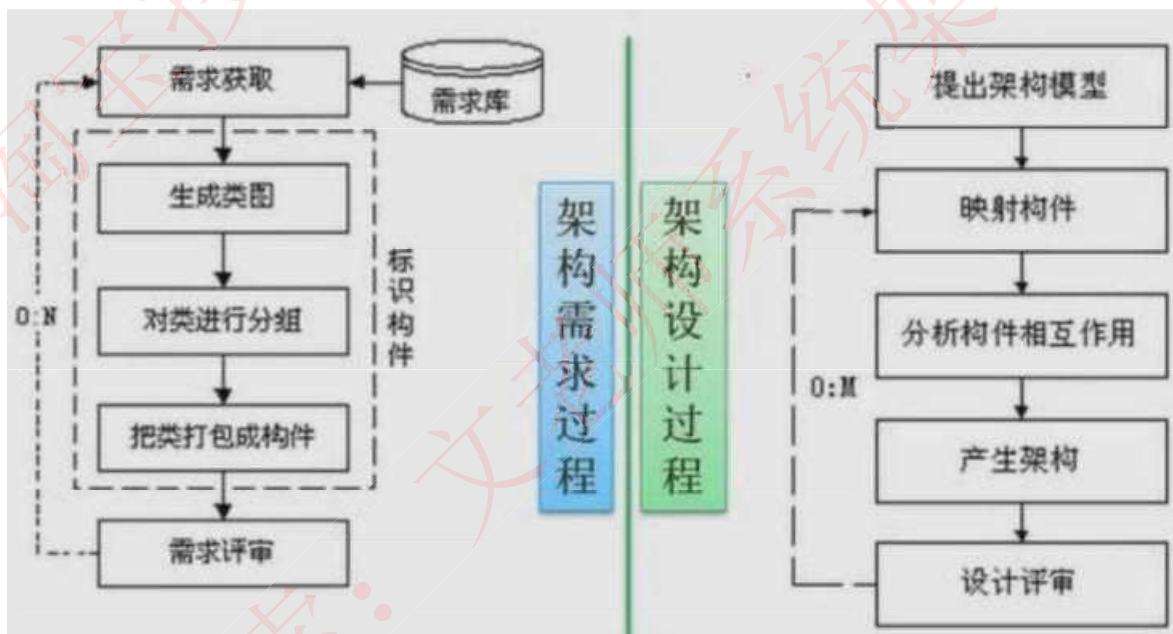
开发过程

架构设计是在需求分析之后，概要设计之前，是为了解决需求分析和软件设计之间的鸿沟问题。基于架构的软件开发过程可分为下列六步：



架构需求：重在掌握**标识构件**的三步，如下左图。

架构设计：将需求阶段的标识构件映射成构件，进行分析，如下右图。

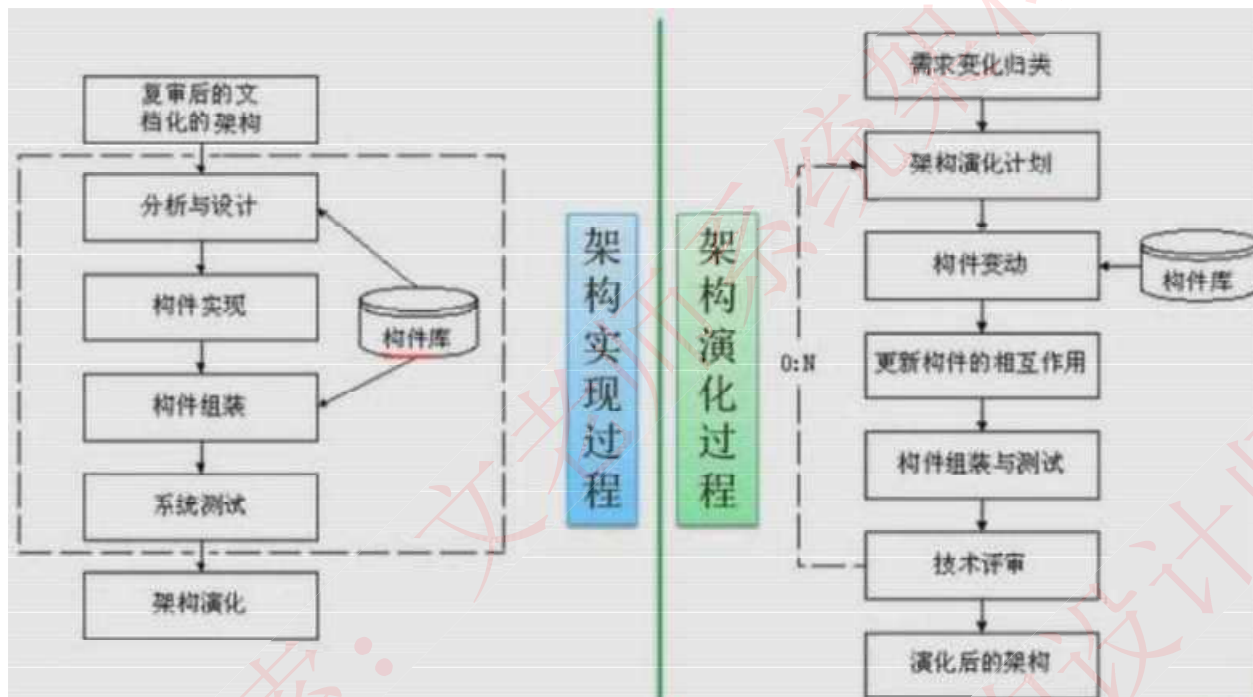


架构（体系结构）文档化：主要产出两种文档，即**架构（体系结构）规格说明**，测试**架构（体系结构）需求的质量设计说明书**。文档是至关重要的，是所有人员通信的手段，关系开发的成败。

架构复审：由外部人员（独立于开发组织之外的人，如用户代表和领域专家等）参加的复审，复审架构是否满足需求，质量问题，构件划分合理性等。若复审不过，则返回架构设计阶段进行重新设计、文档化，再复审。

架构实现：用实体来显示架构。实现构件，构件组装成系统，如下左图：

架构演化：对架构进行改变，按需求增删构件，使架构可复用，如下右图：



3 软件架构风格

软件体系结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。架构风格定义一个系统家族，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。对软件架构风格的研究和实践促进对设计的重用，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。

架构设计的一个核心问题是能否达到架构级的软件复用。

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。

3.1 基本架构风格

数据流风格：面向数据流，按照一定的顺序从前向后执行程序，代表的风格有 批处理序列、管道-过滤器。

调用/返回风格：构件之间存在互相调用的关系，一般是显式的调用，代表的风格有 主程序/子程序、面向对象、层次结构（C/S 架构风格本质属于层次风格）。

独立构件风格：构件之间是互相独立的，不存在显式的调用关系，而是通过某个事件触发、异步的方式来执行，代表的风格有 进程通信、事件驱动系统（隐式调用）。

虚拟机风格：自定义了一套规则供使用者使用，使用者基于这个规则来开发构件，能够跨平台适配，代表的风格有 解释器、基于规则的系统。

仓库风格：以数据为中心，所有的操作都是围绕建立的数据中心进行的，代表的风格有 数据库系统、超文本系统、黑板系统。

数据流风格

批处理序列：构件为一系列固定顺序的计算单元，构件之间只通过数据传递交互。每个处理步骤是一个独立的程序，每一步必须在其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。

管道-过滤器：每个构件都有一组输入和输出，构件读取输入的数据流，经过内部处理，产生输出数据流。前一个构件的输出作为后一个构件的输入，前后数据流关联。过滤器就是构件，连接件就是管道。

早期编译器就是采用的这种架构，要一步一步处理的，均可考虑此架构风格。

二者区别在于批处理前后构件不一定有关联，并且是作为整体传递，即必须前一个执行完才能执行下一个。管道-过滤器是前一个输出作为后一个输入，前面执行到部分可以开始下一个的执行。

调用/返回风格

主程序/子程序：单线程控制，把问题划分为若干个处理步骤，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，充当连接件的绝色。调用关系具有层次性，其语义逻辑表现为主程序的正确性取决于它调用的子程序的正确性。

面向对象：构件是对象，对象是抽象数据类型的实例。在抽象数据类型中，数据的表示和它们的相应操作被封装起来，对象的行为体现在其接受和请求的动作。连接件即使对象间交互的方式，对象是通过函数和过程的调用来交互的。

层次结构：构件组成一个层次结构，连接件通过决定层间如何交互的协议来定义。每层为上一层提供服务，使用下一层的服务，只能见到与自己邻接的层。通过层次结构，可以将大的问题分解为若干个渐进的小问题逐步解决，可以隐藏问题的复杂度。修改某一层，最多影响其相邻的两层（通常只能影响上层）。

层次结构优点：

- 1、支持基于可增加抽象层的设计，允许将一个复杂问题分解成一个增量步骤序列的实现。
- 2、不同的层次处于不同的抽象级别，越靠近底层，抽象级别越高；越靠近顶层，抽象级别越低。
- 3、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，同样为软件复用提供了强大的支持。

缺点：

- 1、并不是每个系统都可以很容易的划分为分层的模式。
- 2、很难找到一个合适的、正确的层次抽象方法。

独立构件风格

进程通信：构件是独立的进程，连接件是消息传递。构件通常是命名过程，消息传递的方式可以是点对点、异步或同步方式，以及远程过程（方法）调用等。

事件驱动系统（隐式调用）：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。构件中的过程在一个或多个事件中注册，当某个事件被触发时，系统自动调用在这个事件中注册的所有过程。一个事件的触发就导致了另一个模块中的过程调用。这种风格中的构件是匿名的过程，它们之间交互的连接件往往是以过程之间的隐式调用来实现的。

主要优点是软件复用提供了强大的支持，为构件的维护和演化带来了方便；缺点是构件放弃了对系统计算的控制。

虚拟机风格

解释器：通常包括一个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的进度的数据结构。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，缺点是执行效率低。

基于规则的系统：包括规则集、规则解释器、规则/数据选择器和工作内存，一般用在人工智能领域和 DSS 中。

仓库风格（数据共享风格，以数据为中心）

数据库系统：构件主要有两大类，一类是中央共享数据源，保存当前系统的数据状态；另一类是多个独立处理单元，处理单元对数据元素进行操作。

黑板系统：包括知识源、黑板和控制三部分。知识源包括若干独立计算的不同单元，提供解决问题的知识。知识源响应黑板的变化，也只修改黑板；黑板是一个全局数据库，包含问题域解空间的全部状态，是知识源相互作用的唯一媒介；知识源响应式通过黑板状态的变化来控制的。黑板系统通常应用在对于解决问题没有确定性算法的软件中（信号处理、问题规划和编译器优化等）。

超文本系统：构件以网状链接方式相互连接，用户可以在构件之间进行按照人类的联想思维方式任意跳转到相关构件。是一种非线性的网状信息组织方法，它以节点为基本单位，链作为节点之间的联想式关联。通常应用在互联网领域。

现代编译器的集成开发环境一般采用数据仓储（即以数据为中心的架构风格）架构风格进行开发，其中心数据就是程序的语法树。

闭环控制架构（过程控制）

当软件被用来操作一个物理系统时，软件与硬件之间可以粗略的表示为一个反馈循环，这个反馈循环通过接受一定的输入，确定一系列的输出，最终使环境达到一个新的状态，适合于嵌入式系统，涉及连续的动作与状态。

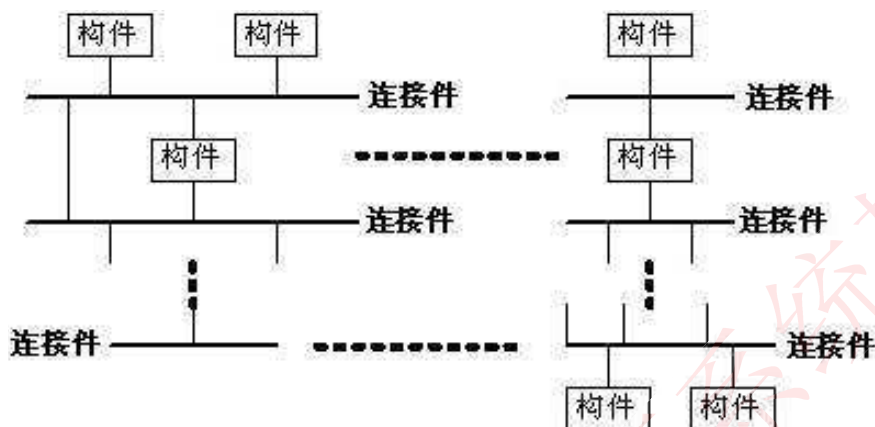
C2 架构风格

C2 体系结构风格可以概括为：通过连接件绑定在一起的按照一组规则运作的并行构件网络。C2 风格中的系统组织规则如下：

(1) 系统中的构件和连接件都有一个顶部和一个底部；

(2) 构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直接连接是不允许的；

- (3)一个连接件可以和任意数目的其它构件和连接件连接;
- (4)当两个连接件进行直接连接时,必须由其中一个的底部到另一个的顶部。



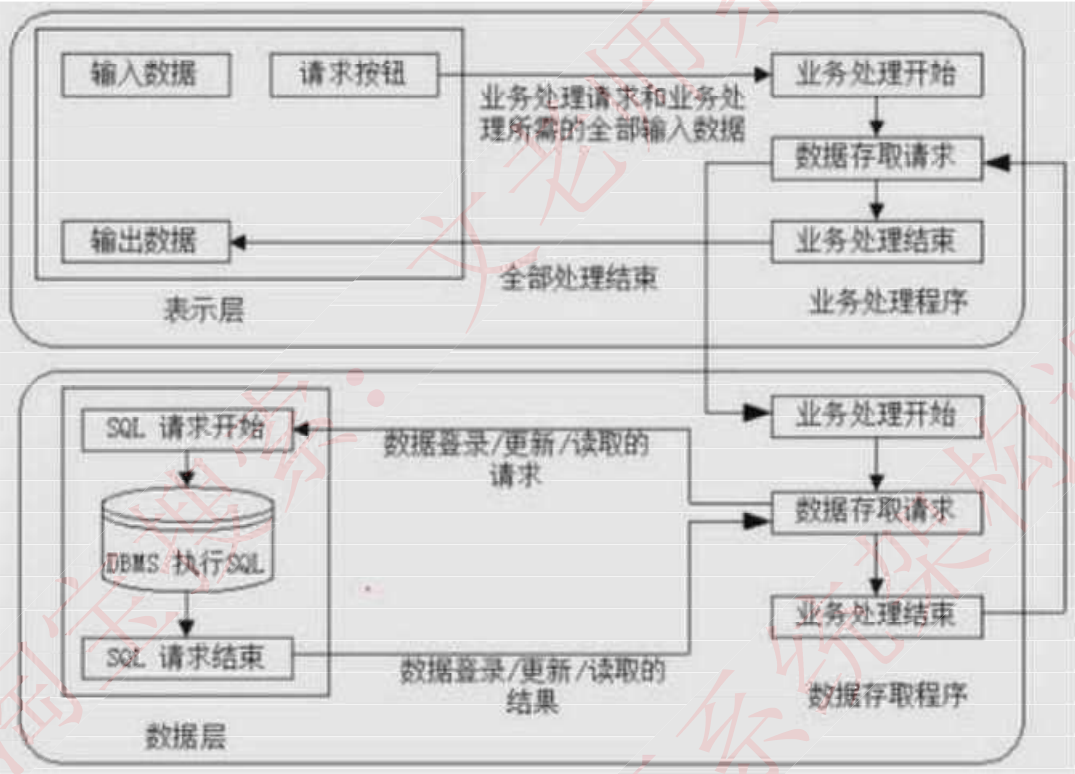
架构风格汇总（重要必背）

架构风格名	常考关键字及实例	简介
数据流-批处理	传统编译器, 每个阶段产生的结果作为下一个阶段的输入, 区别在于整体。	一个接一个, 以整体为单位。
数据流-管道-过滤器		一个接一个, 前一个输出是后一个输入。
调用/返回-主程序/子程序		显示调用, 主程序直接调用子程序。
调用/返回-面向对象		对象是构件, 通过对象调用封装的方法和属性。
调用/返回-层次结构		分层, 每层最多影响其上下两层, 有调用关系。
独立构件-进程通信		进程间独立的消息传递, 同步异步。
独立构件-事件驱动 (隐式调用)	事件触发推动动作, 如程序语言的语法高亮、语法错误提示	不直接调用, 通过事件驱动。
虚拟机-解释器	自定义流程, 按流程执行, 规则随时改变, 灵活定义, 业务灵活组合 机器人。	解释自定义的规则, 解释引擎、存储区、数据结构。
虚拟机-规则系统		规则集、规则解释器、选择器和工作内存, 用于 DSS 和人工智能、专家系统。
仓库-数据库	现代编译器的集成开发环境 IDE, 以数据为中心。 又称为数据共享风格	中央共享数据源, 独立处理单元。
仓库-超文本		网状链接, 多用于互联网。
仓库-黑板		语音识别、知识推理等问题复杂、解空间很大、求解过程不确定的这一类软件系统, 黑板、知识源、控制。
闭环-过程控制	汽车巡航定速, 空调温度调节, 设定参数, 并不断调整。	发出控制命令并接受反馈, 循环往复达到平衡。
C2 风格	构件和连接件、顶部和底部	通过连接件绑定在一起按照一组规则运作的

3.2 层次结构风格

两层 c/s 架构

客户端和服务端都有处理功能，相比较于传统的集中式软件架构，还是有不少优点的，但是现在已经不常用，原因有：开发成本较高、客户端程序设计复杂、信息内容和形式单一、用户界面风格不一、软件移植困难、软件维护和升级困难、新技术不能轻易应用、安全性问题、服务器端压力大难以复用。



三层 c/s 架构

将处理功能独立出来，表示层和数据层都变得简单。表示层在客户机上，功能层在应用服务器上，数据层在数据库服务器上。即将两层 C/S 架构中的数据从服务器中独立出来了。其优点下面四点：

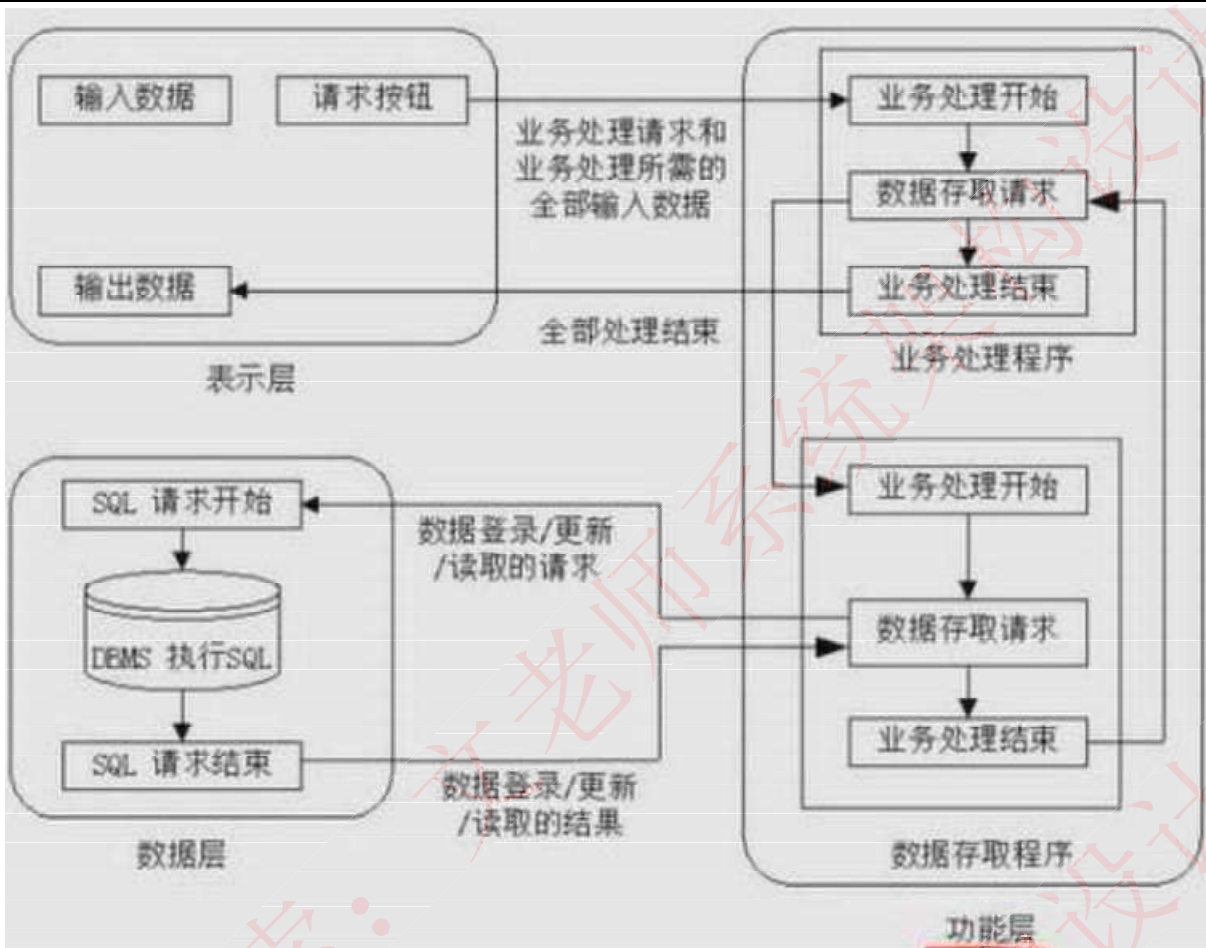
各层在逻辑上保持相对独立，整个系统的逻辑结构更为清晰，能提高系统和软件的可维护性和可扩展性；

允许灵活有效的选用相应的平台和硬件系统，具有良好的可升级性和开放性；

各层可以并行开发，各层也可以选择各自最适合的开发语言；

功能层有效的隔离表示层与数据层，为严格的安全管理奠定了坚实的基础，整个系统的管理层次也更加合理和可控制。

三层 C/S 架构设计的关键在于各层之间的通信效率，要慎重考虑三层间的通信方法、通信频度和数据量，否则即使分配给各层的硬件能力很强，性能也不高。



三层 B/S 架构

是三层 C/S 架构的变种，将客户端变为用户客户端上的浏览器，将应用服务器变为网络上的 WEB 服务器，又称为 0 客户端架构，虽然不用开发客户端，但有很多缺点，主要是数据处理能力差：

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；

安全性难以控制；

在数据查询等响应速度上，要远远低于 C/S 架构；

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 OLTP 应用。

混合架构风格

内外有别模型：企业内部使用 C/S，外部人员访问使用 B/S。

查改有别模型：采用 B/S 查询，采用 C/S 修改。

混合架构实现困难，且成本高。

富互联网应用 RIA

弥补三层 B/S 架构存在的问题，RIA 是一种用户接口，比用 HTML 实现的接口更加健壮，且有可视化内容，本质还是网站模式，其优点如下：

RIA 结合了 C/S 架构反应速度快、交互性强的优点与 B/S 架构传播范围广及容易传播的特性；

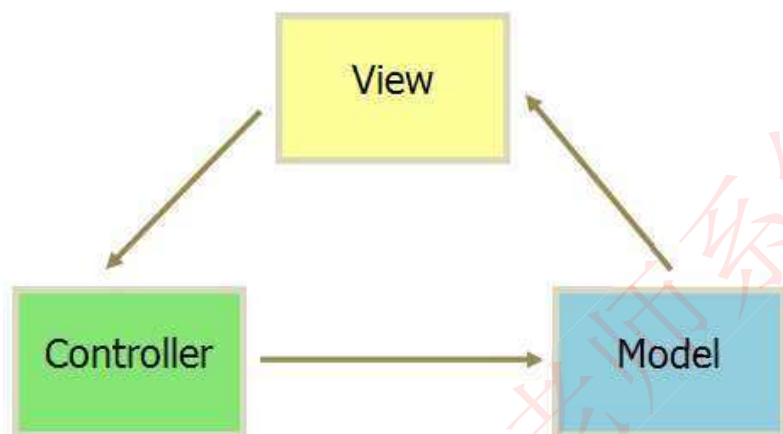
RIA 简化并改进了 B/S 架构的用户交互；

数据能够被缓存在客户端，从而可以实现一个比基于 HTML 的响应速度更快且数据往返于服务器的次数更少的用户界面。

本质还是 0 客户端，借助于高速网速实现必要插件在本地的快速缓存，增强页面对动态页面的支持能力，典型的如小程序。

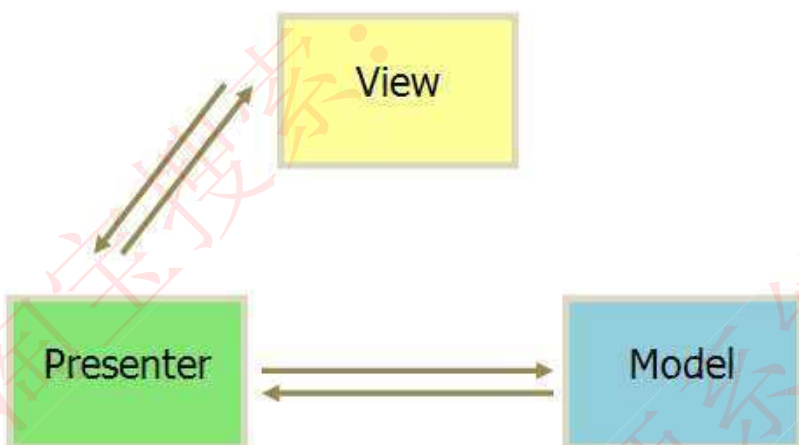
MVC

用户操作->View（负责接收用户的输入操作）->Controller（业务逻辑处理）->Model（数据持久化）->View（将结果反馈给 View）。经典 MVC 架构如下图所示：



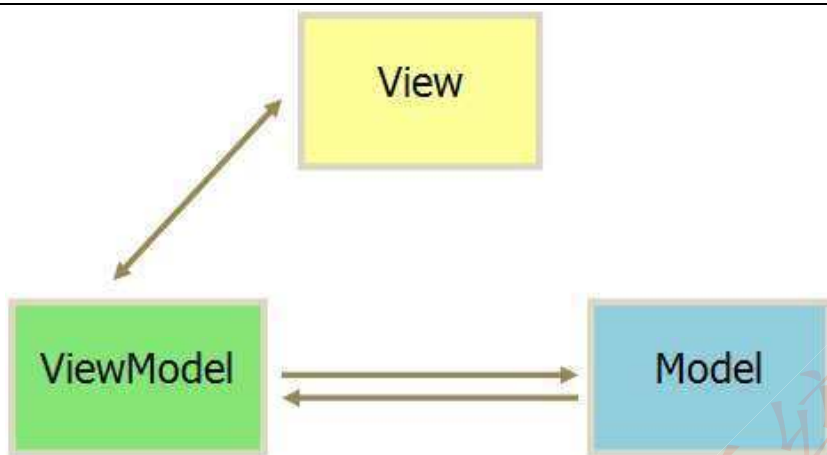
MVP

MVP 是把 MVC 中的 Controller 换成了 Presenter（呈现），目的就是为了完全切断 View 跟 Model 之间的联系，由 Presenter 充当桥梁，做到 View-Model 之间通信的完全隔离。



MVVM

如果说 MVP 是对 MVC 的进一步改进，那么 MVVM 则是思想的完全变革。它是将“数据模型数据双向绑定”的思想作为核心，因此在 View 和 Model 之间没有联系，通过 ViewModel 进行交互，而且 Model 和 ViewModel 之间的交互是双向的，因此视图的数据的变化会同时修改数据源，而数据源数据的变化也会立即反应到 View 上。



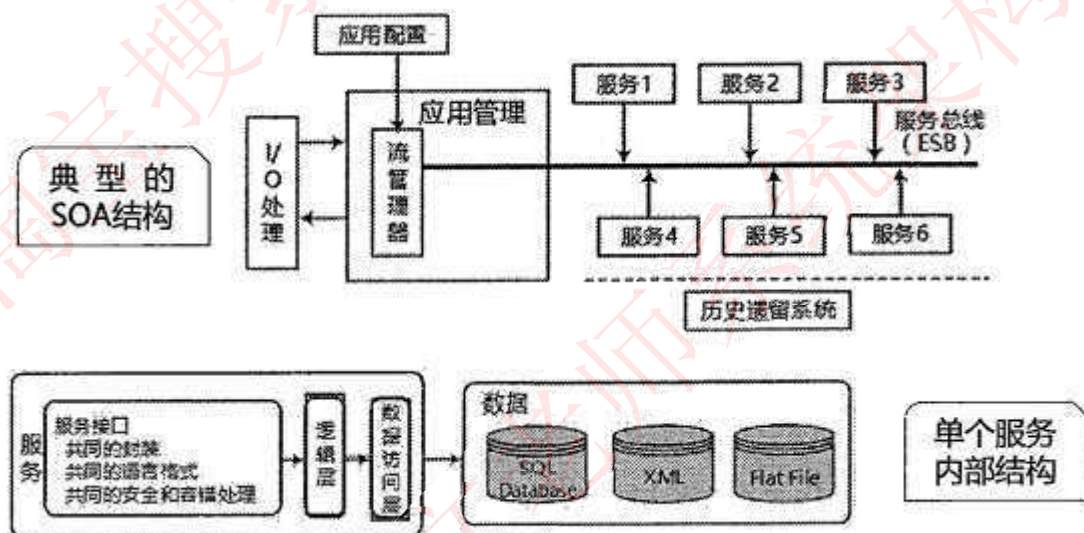
3.3 面向服务的架构 SOA

概念

SOA 是一种粗粒度、松耦合服务架构，服务之间通过简单、精确定义接口进行通信，不涉及底层编程接口和通信模型。

在 SOA 中，服务是一种为了满足某项业务需求的操作、规则等的逻辑组合，它包含一系列有序活动的交互，为实现用户目标提供支持。

SOA 并不仅仅是一种开发方法，还具有管理上的优点，管理员可直接管理开发人员所构建的相同服务。多个服务通过企业服务总线提出服务请求，由应用管理来进行处理，如下：



实施 SOA 的**关键目标是实现企业 IT 资产重用的最大化**，在实施 SOA 过程中要牢记以下**特征**：可从企业外部访问、随时可用（服务请求能被及时响应）、粗粒度接口（粗粒度提供一项特定的业务功能，而细粒度服务代表了技术构件方法）、服务分级、松散耦合（服务提供者和服务使用者分离）、可重用的服务及服务接口设计管理、标准化的接口（WSDL、SOAP、**XML 是核心**）、支持各种消息模式、精确定义的服务接口。

从基于对象到基于构件再到基于服务，架构越来越松散耦合，粒度越来越粗，接口越来越标准。

基于服务的构件与传统构件的**区别**有四点：

服务构件粗粒度，传统构件细粒度居多；

服务构件的接口是标准的，主要是 WSDL 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。

关键技术

SOA 中应用的关键技术如下表。

功能	协议
发现服务	UDDI、DISCO
描述服务	WSDL、XML Schema
消息格式层	SOAP、REST
编码格式层	XML (DOM , SAX)
传输协议层	HTTP、TCP/IP、SMTP等

发现服务

UDDI：用于 Web 服务注册和服务查找，描述了服务的概念，定义了编程的接口，供其他企业来调用。

DISCO：发现公开服务的功能及交互协议。

描述服务

WSDL（WEB 服务描述语言）协议：用于描述 Web 服务的接口和操作功能，描述网络服务。

消息格式层

SOAP 为建立 Web 服务和服务请求之间的通信提供支持。

REST (Representational State 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。

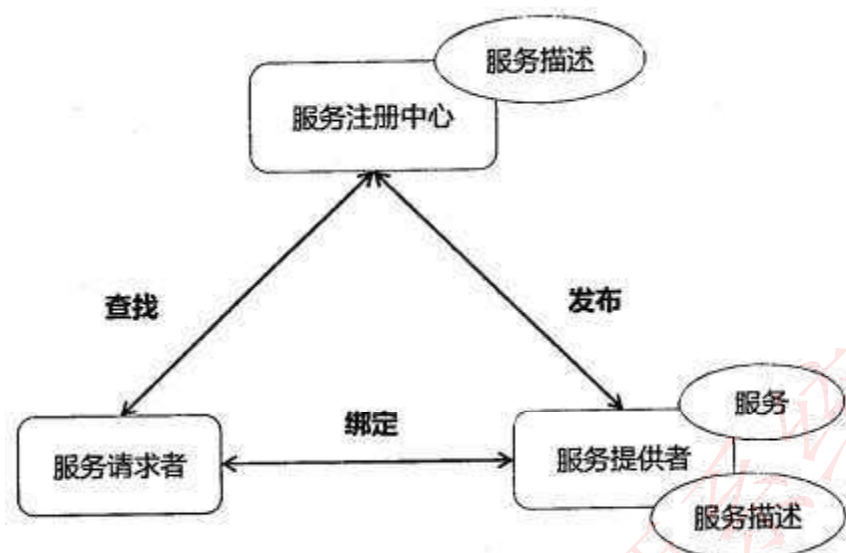
编码格式层

扩展标记语言（Extensible Markup Language, XML），用于标记电子文件使其具有结构性的标记语言，可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。

SOA 的实现方式

WEB Service

服务提供者、服务注册中心（中介，提供交易平台，可有可无）、服务请求者。服务提供者将服务描述**发布**到服务注册中心，供服务请求者**查找**，查找到后，服务请求者将**绑定**查找结果。如下图：



分为六层：底层传输层（负责底层消息的传输，采用 HTTP、JMS、SMTP 协议），服务通信协议层（描述并定义服务间通信的技术标准，采用 SOAP 和 REST 协议），服务描述层（采用 WSDL 协议），服务层（对企业应用系统进行包装，通过 WSDL 定义的标准进行调用），业务流程层（支持服务发现、服务调用和点到点的服务调用，采用 WSDPEL 标准），服务注册层（采用 UDDI 协议）。

服务注册表

服务注册：应用开发者（服务提供者）在注册表中公布服务的功能。

服务位置：服务使用者（服务应用开发者），帮助他们查询注册服务，寻找符合自身要求的服务。

服务绑定：服务使用者利用检索到的服务接口来编写代码，所编写的代码将与注册的服务绑定，调用注册的服务，以及与它们实现互动。

本质与 WEB Service 类似，只是使用一个注册表来代替服务注册中心。

企业服务总线 ESB

客户端（服务请求者）、基础架构服务（中间件）、核心集成服务（提供服务）。本质也是通过网络，有下列六点功能：

提供位置透明性的消息路由和寻址服务；

提供服务注册和命名的管理功能；

支持多种的消息传递范型；

支持多种可以广泛使用的传输协议；

支持多种数据格式及其相互转换；

提供日志和监控功能。

4 软件架构复用

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。

软件架构复用的类型包括机会复用和系统复用。机会复用是指开发过程中，只要发现有可复用的资产，就对其进行复用。系统复用是指在开发之前，就要进行规划，以决定哪些需要复用。

可复用的资产包括：需求、架构设计、元素、建模与分析、测试、项目规划、过程方法和工具、人员、样本系统、缺陷消除。

复用的基本过程主要包括 3 个阶段：首先构造 / 获取可复用的软件资产，其次管理这些资产（构件库），最后针对特定的需求，从这些资产中选择可复用的部分，以开发满足需求的应用系统。

5 特定领域软件架构 DSSA

DSSA 就是专用于一类特定类型的任务(领域)的、在整个领域中能有效地使用的、为成功构造应用系统限定了标准的组合结构的软件构件的集合。

DSSA 就是一个特定的问题领域中支持一组应用的领域模型、参考需求、参考架构等组成的开发基础，其目标就是支持在一个特定领域中多个应用的生成。

垂直域：在一个特定领域中的通用的软件架构，是一个完整的架构。

水平域：在多个不同的特定领域之间的相同的部分的小工具（如购物和教育都有收费系统，收费系统即是水平域）。

DSSA 的三个基本活动

领域分析：这个阶段的主要目标是获得领域模型（领域需求）。识别信息源，即整个领域工程过程中信息的来源，可能的信息源包括现存系统、技术文献、问题域和系统开发的专家、用户调查和市场分析、领域演化的历史记录等，在此基础上就可以分析领域中系统的需求，确定哪些需求是领域中的系统广泛共享的，从而建立领域模型。

领域设计：这个阶段的目标是获得 DSSA。DSSA 描述在领域模型中表示的需求的解决方案，它不是单个系统的表示，而是能够适应领域中多个系统的需求的一个高层次的设计。建立了领域模型之后，就可以派生出满足这些被建模的领域需求 DSSA。

领域实现：这个阶段的主要目标是依据领域模型和 DSSA 开发和组织可重用信息。这些可重用信息可能是从现有系统中提取得到，也可能需要通过新的开发得到。

以上过程是一个反复的、逐渐求精的过程。在实施领域工程的每个阶段中，都可能返回到以前的步骤，对以前的步骤得到的结果进行修改和完善，再回到当前步骤，在新的基础上进行本阶段的活动。

参与 DSSA 的四种角色人员

领域专家：包括该领域中系统的有经验的用户、从事该领域中系统的需求分析、设计、实现以及项目管理的有经验的软件工程师等。提供关于领域中系统的需求规约和实现的知识，帮助组织规范的、一致的领域字典，帮助选择样本系统作为领域工程的依据，复审领域模型、DSSA 等领域工程产品，等等。

领域分析人员：由具有知识工程背景的有经验的系统分析员来担任。控制整个领域分析过程，进行知识获取，将获取的知识组织到领域模型中。

领域设计人员：由有经验的软件设计人员来担任。根据领域模型和现有系统开发出 DSSA，并对 DSSA 的准确性和一致性进行验证。

领域实现人员：由有经验的程序设计人员来担任。根据领域模型和 DSSA，开发构件。

建立 DSSA 的过程

(1) 定义领域范围：领域中的应用要满足用户一系列的需求。

(2) 定义领域特定的元素：建立领域的字典，归纳领域中的术语，识别出领域中相同和不相同的元素。

(3) 定义领域特定的设计和实现需求的约束：识别领域中的所有约束，这些约束对领域的设计和实现会造成什么后果。

(4) 定义领域模型和架构：产生一般的架构，并描述其构件说明。

(5) 产生、搜集可复用的产品单元：为 DSSA 增加复用构件，使可用于新的系统。

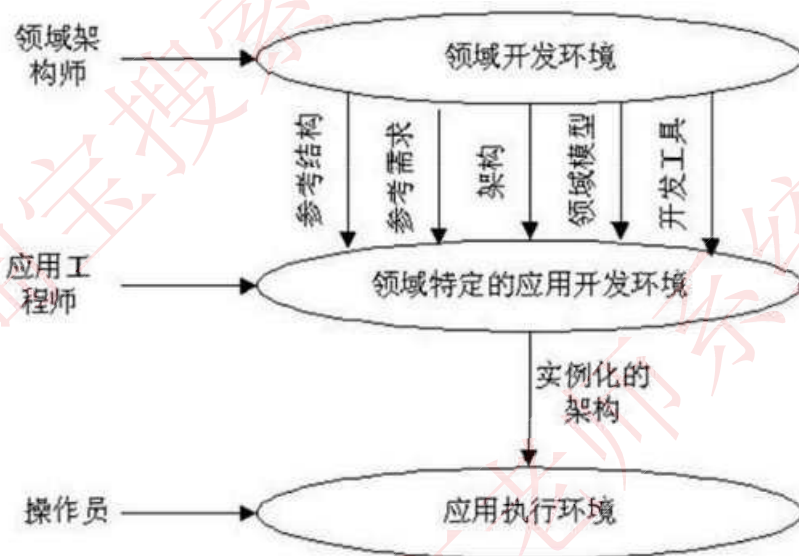
以上过程是并发的、递归的、反复的、螺旋型的。该过程的目的是将用户的需求映射为基于实现限制集合的软件需求，这些需求定义了 DSSA。

DSSA 的三层次系统模型

领域开发环境：领域架构师决定核心架构，产出参考结构、参考需求、架构、领域模型、开发工具。

领域特定的应用开发环境：应用工程师根据具体环境来将核心架构实例化。

应用执行环境：操作员实现实例化后的架构。



6 系统质量属性与架构评估

6.1 软件系统质量属性

软件系统的质量就是“软件系统与明确地和隐含地定义的需求相一致的程度”。

可以将软件系统的质量属性分为开发期质量属性和运行期质量属性 2 个部分。

1.开发期质量属性主要指在软件开发阶段所关注的质量属性，主要包含 6 个方面。

(1) 易理解性：指设计被开发人员理解的难易程度。

(2) 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。

(3) 可重用性：指重用软件系统或某一部分的难易程度。

(4) 可测试性：对软件测试以证明其满足需求规范的难易程度。

(5) 可维护性：当需要修改缺陷、增加功能、提高质量属性时，识别修改点并实施修改的难易程度。

(6) 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。

2.运行期质量属性主要指在软件运行阶段所关注的质量属性，主要包含 7 个方面。

(1) 性能：性能是指软件系统及时提供相应服务的能力，如速度、吞吐量和容量等的要求。

(2) 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。

(3) 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。

(4) 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。

(5) 可靠性：软件系统在一定的时间内持续无故障运行的能力。

(6) 可用性：指系统在一定时间内正常工作的时间所占的比例。可用性会受到系统错误，恶意攻击，高负载等问题的影响。

(7) 鲁棒性：是指软件系统在非正常情况（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。

面向架构评估的质量属性

1、性能：指系统的响应能力，即要经过多长时间才能对某个事件做出响应，或者在某段时间内系统所能处理的事件的个数。如响应时间、吞吐量。设计策略：优先级队列、增加计算资源、减少计算开销、引入并发机制、采用资源调度等。

2、可靠性：是软件系统在应用或系统错误面前，在意外或错误使用的情况下维持软件系统的功能特性的基本能力。如 MTTF、MTBF。设计策略：心跳、Ping/Echo、冗余、选举。包括容错和健壮性。

3、可用性：是系统能够正常运行的时间比例，经常用两次故障之间的时间长度或在出现故障时系统能够恢复正常的速度来表示。如故障间隔时间。设计策略：心跳、Ping/Echo、冗余、选举。

4、安全性：是指系统在向合法用户提供服务的同时能够阻止非授权用户使用的企图或拒绝服务的能力。如保密性、完整性、不可抵赖性、可控性。设计策略：入侵检测、用户认证、用户授权、追踪审计。

5、可修改性：指能够快速的以较高的性能价格比对系统进行变更的能力。通常以某些具体的变更为基准，通过考察这些变更的代价衡量。包括可维护性、可扩展性、结构重组、可移植性。设计策略：接口-实现分类、抽象、信息隐藏。

6、功能性：是系统所能完成所期望的工作的能力。一项任务的完成需要系统中许多或大多数构件的相互协作。

7、可变性：指体系结构经扩充或变更而成为新体系结构的能力。这种新体系结构应该符合预先定义的规则，在某些具体方面不同于原有的体系结构。当要将某个体系结构作为一系列相关产品的基础时，可变性是很重要的。

8、互操作性：作为系统组成部分的软件不是独立存在的，经常与其他系统或自身环境相互作用。为了支持互操作性，软件体系结构必须为外部可视的功能特性和数据结构提供精心设计的软件入口。程序和用其他编程语言编写的软件系统的交互作用就是互操作性的问题，也影响应用的软件体系结构。

质量属性场景是一种面向特定质量属性的需求。它由 6 部分组成：

- 刺激源（Source）：这是某个生成该刺激的实体（人、计算机系统或者任何其他刺激器）。
- 刺激（Stimulus）：该刺激是当刺激到达系统时需要考虑的条件。
- 环境（Environment）：该刺激在某些条件内发生。当激励发生时，系统可能处于过载、运行或者其他情况。
- 制品（Artifact）：某个制品被激励。这可能是整个系统，也可能是系统的一部分。
- 响应（Response）：该响应是在激励到达后所采取的行动。
- 响应度量（Measurement）：当响应发生时，应当能够以某种方式对其进行度量，以对需求进行测试。

可修改性质量属性场景描述实例：

场景要素	可能的情况
刺激源	最终用户、开发人员、系统管理员
刺激	希望增加、删除、修改、改变功能、质量属性、容量等
环境	系统设计时、编译时、构建时、运行时
制品	系统用户界面、平台、环境或与目标系统交互的系统
响应	查找架构中需要修改的位置，进行修改且不会影响其他功能，对所做的修改进行测试，部署所做的修改
响应度量	根据所影响元素的数量度量的成本、努力、资金；该修改对其他功能或质量属性所造成影响的程度

6.2 系统架构评估

系统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。

软件架构评估在架构设计之后，系统设计之前，因此与设计、实现、测试都没有关系。评估的目的是为了评估所采用的架构是否能解决软件系统需求，但不是单纯的确定是否满足需求。

三种常用的评估方式

基于调查问卷（检查表）的方式：类似于需求获取中的问卷调查方式，只不过是架构方面的问卷，要求评估人员对领域熟悉。

基于度量的方式：制定一些定量来度量架构，如代码行数等。要制定质量属性和度量结果之间的映射，要求评估人员对架构熟悉。涉及 3 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，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。

基于场景的方式：主要方法。首先要确定应用领域的功能和软件架构的结构之间的映射，然后要设计用于体现待评估质量属性的场景(即 4+1 视图中的场景)，最后分析软件架构对场景的支持程度。要求评估人员即对领域熟悉，也对架构熟悉。

三种评估方式的比较：

评估方式	调查问卷或检查表		场景	度量
	调查问卷	检查表		
通用性	通用	特定领域	特定系统	通用或特定领域
评估者对架构的了解程度	粗略了解	无限制	中等了解	精确了解
实施阶段	早	中	中	中
客观性	主观	主观	较主观	较客观

由上表可知，场景最不通用，度量要求对构架精确了解，调查问卷实施阶段最早，度量最客观。其中，**基于场景的方式是主流**。

架构评估中的重要概念

敏感点：是指为了实现某种特定的质量属性，一个或多个构件所具有的特性。

权衡点：是影响多个质量属性的特性，是多个质量属性的敏感点。

风险点与非风险点不是以标准专业术语形式出现的，只是一个常规概念，即可能引起风险的因素，可称为风险点。某个做法如果有隐患，有可能导致一些问题，则为风险点；而如果某件事是可行的 可接受的，则为非风险点。

从三个方面对场景进行设计：刺激（事件）；环境（事件发生的环境）；响应（架构响应刺激的过程）。

6.3 基于场景的评估方法

基于场景的架构分析方法 SAAM

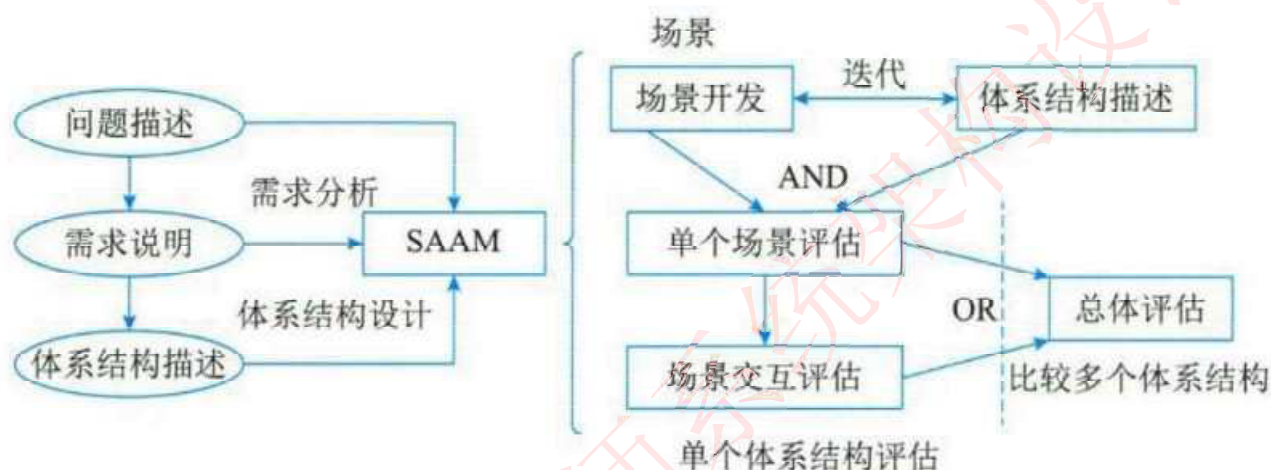
SAAM 是一种非功能质量属性的架构分析方法，是最早形成文档并得到广泛应用的软件架构分析方法。

特定目标。SAAM 的目标是对描述应用程序属性的文档，验证基本的架构假设和原则。

质量属性。这一方法的基本特点是把任何形式的质量属性都具体化为场景，但可修改性是 SAAM 分析的主要质量属性。

架构描述。SAAM 用于架构的最后版本，但早于详细设计。架构的描述形式应当被所有参与者理解。功能、结构和分配被定义为描述架构的 3 个主要方面。

方法活动。SAAM 的主要输入是问题描述、需求声明和架构描述。下图描绘了 SAAM 分析活动的相关输入及评估过程。包括 5 个步骤，即场景开发、架构描述、单个场景评估、场景交互和总体评估。



架构权衡分析法 ATAM

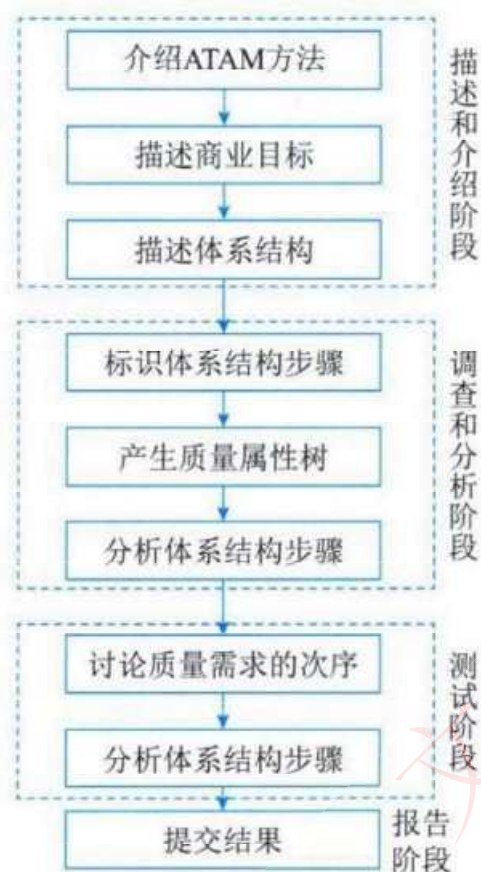
让架构师明确如何权衡多个质量目标，参与者有评估小组、项目决策者和其他项目相关人。

ATAM 被分为四个主要的活动领域，分别是场景和需求收集、体系结构视图和场景实现、属性模型构造和分析、折中。整个评估过程强调以属性作为架构评估的核心概念。主要针对性能、可用性、安全性和可修改性，在系统开发之前，对这些质量属性进行评价和折中。

ATAM 方法采用效用树（Utility tree）这一工具来对质量属性进行分类和优先级排序。效用树的结构包括：树根-质量属性-属性分类-质量属性场景（叶子节点）。需要注意的是，ATAM 主要关注 4 类质量属性：性能、安全性、可修改性和可用性，这是因为这 4 个质量属性是利益相关者最为关心的。

ATAM 方法架构评估实践

注意：这个内容选择案例不会考。但是可用于写论文的步骤，建议写这个主题论文的，重点看一下第二版教材第 8.3 节，第 289 页，书上给出了胡佛事件架构、银行事件架构这两个具体架构的详细描述，可作为大家写论文时的素材参考，不过书上写的比较多比较复杂，大家也要有选择性的参考。当然也可以直接听文老师论文直播里这块的讲解来构思。



成本效益分析法 CBAM

用来对架构建立的成本来进行设计和建模，让决策者根据**投资收益率**来选择合适的架构，可以看做**对 ATAM 的补充**，在 ATAM 确定质量合理的基础上，再对效益进行分析。有下列步骤：

- (1) 整理场景（确定场景，并确定优先级，选择三分之一优先级最高的场景进行分析）；
- (2) 对场景进行细化（对每个场景详细分析，确定最好、最坏的情况）；
- (3) 确定场景的优先级（项目干系人对场景投票，根据投票结果确定优先级）；
- (4) 分配效用（对场景响应级别确定效用表，建立策略、场景、响应级别的表格）；
- (5) 形成“策略-场景-响应级别的对应关系”；
- (6) 确定期望的质量属性响应级别的效用（根据效用表确定所对应的具体场景的效用表）；
- (7) 计算各架构策略的总收益；
- (8) 根据受成本限制影响的投资报酬率选择架构策略（估算成本，用上一步的收益减去成本，得出收益，并选择收益最高的架构策略）。

其他评估方法（仅了解）

1.SAEM 方法。将软件架构看作一个最终产品以及设计过程中的一个中间产品，从外部质量属性和内部质量属性两个角度来阐述它的评估模型，旨在为软件架构的质量评估创建一个基础框架。

外部属性指用户定义的质量属性，而内部属性指开发者决定的质量属性。该软件架构评估模型包含以下几个流程。

- (1) 对待评估的质量属性进行规约建模。

(2) 为外部和内部的质量属性创建度量准则, 先从评估目的(如软件架构比较、最终产品的质量预测), 评估角度(如开发者、用户、维护者), 评估环境(架构作为最终产品或设计中间产品)出发来定义架构评估的目标, 再根据目标相关的属性来提出问题, 然后回答每个问题并提出相应的度量准则。

(3) 评估质量属性, 包括数据收集、度量和结果分析 3 个活动。

2.SAABNet 方法。是一种用来表达和使用定性知识以辅助架构的定性评估。该方法来源于人工智能, 允许不确定、不完整知识的推理。该方法使用 **BBN** 来表示和使用开发过程中的知识, 包含定性和定量的描述, 其中定性的描述是所有结点的图, 定量的描述是每个结点状态相关的条件概率。

其中的变量可分为 3 类, 即架构质量属性变量(如可维护性、灵活性等)、质量属性的度量准则变量(如容错性、响应性等)和架构特征变量(如继承深度、编程语言等), 高层抽象的质量属性变量分解为低层抽象的度量准则变量, 度量准则变量则分解为更低层抽象的架构特征变量。

3.SACMM 方法。是一种软件架构修改的度量方法。

4.SASAM 方法。通过对预期架构(架构设计阶段的相关描述材料)和实际架构(源代码中执行的架构)进行映射和比较来静态地评估软件架构。

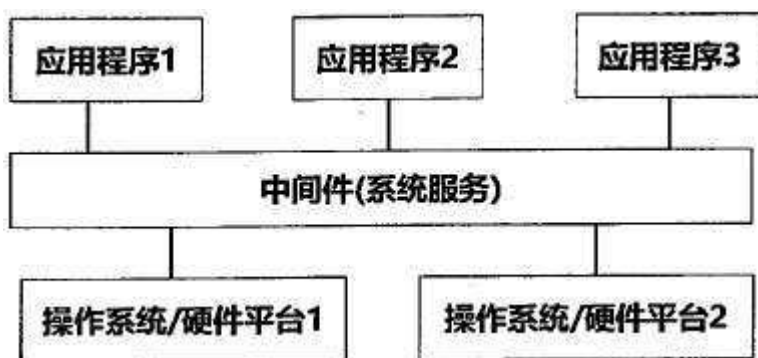
5.ALRRR 方法。是一种软件架构可靠性风险评估方法, 该方法使用动态复杂度准则和动态耦合度准则来定义组件和连接件的复杂性因素, 其中, 动态复杂度准则在某个场景的执行中分析组件的动态行为来度量组件的复杂性, 动态耦合度准则在某个场景的执行中分析连接件的消息传递协议来度量连接件的复杂性。利用失效模式和影响分析(FMEA)技术。

6.AHP(层次分析法)方法。是对定性问题进行定量分析的一种简便、灵活而又实用的多准则决策方法。**AHP** 方法的特点是把复杂问题中的各种因素通过划分为相联系的有序层次使之条理化, 并在一般情况下通过两两对比, 根据一定客观现实的主观判断结构, 把专家意见和分析者的客观判断结果直接、有效地结合起来, 将一定层次上元素的某些重要性进行定量描述, 之后利用数学方法计算反映每一层次元素的相对重要性次序的权值, 并最后通过所有层次之间的总排序计算所有元素的相对权重及对权重进行排序。

7.COSMIC+UML 方法。基于度量模型来评估软件架构可维护性的方法。针对不同表达方式的软件架构, 采用统一的软件度量 **COSMIC** 方法来进行度量和评估。该方法主要是为了辅助分析软件架构的演化方案是否可行, 并在开源软件 **DCMMS** 的软件架构 **UML** 组件图上得以验证。

7 中间件技术

中间件是一种独立的系统软件或服务程序, 可以帮助分布式应用软件在不同的技术之间共享资源。



中间件特点

负责客户机与服务器之间的连接和通信，以及客户机与应用层之间的高效率通信机制。

提供应用层不同服务之间的互操作机制，以及应用层与数据库之间的连接和控制机制。

提供多层架构的应用开发和运行的平台，以及应用开发框架，支持模块化的应用开发。

屏蔽硬件、操作系统、网络和数据库的差异。

提供应用的负载均衡和高可用性、安全机制与管理功能，以及交易管理机制，保证交易的一致性。

提供一组通用的服务去执行不同的功能，避免重复的工作和使应用之间可以协作。

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

在分布式系统中，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。

2) 事务处理（交易）中间件

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保证系统永不停机，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，并进行负载均衡的调度，实现与昂贵的可靠性机和大型计算机系统的同等功能，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。

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，不擅长做数据的写入任务，受 HTTP 协议的限制多等，就必须对其进行修改和扩充，因此出现了 Web 服务器中间件。

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、加密和认证等），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,还需要集成各结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,但是其过于庞大;**JavaBeans** 较灵活简单,很适合用于浏览器,但运行效率有待改善;**COM+**模型主要适合 **Windows** 平台,已在桌面系统广泛使用。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

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、虚拟社区和虚拟缓冲等,也是当前最热门的研发项目。

软件构件

构件又称为组件,是一个自包容、可复用的程序集。构件是一个程序集,或者说是一组程序的集合。构件的两个最重要的特性是自包容与可重用。

构件组装模型的一般开发过程:设计构件组装、建立构件库、构建应用软件、测试与发布。

商用构件标准规范-CORBA (公共对象请求代理体系结构)

公共对象请求代理架构(**CORBA**)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(**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。

COREA CCM 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。

对象管理组织(**OMG**)基于 **CORBA** 基础设施定义了**四种构件标准**。

实体(**Entity**)构件需要长期持久化并主要用于事务性行为,由容器管理其持久化。

加工(**Process**)构件同样需要容器管理其持久化,但没有客户端可访问的主键。

会话(**Session**)构件不需要容器管理其持久化,其状态信息必须由构件自己管理。

服务(**Service**)构件是无状态的。

CORBA 对象可看作是一个具有对象标识、对象接口及对象实现的抽象实体。之所以称为抽象的，是因为并没有硬性规定 CORBA 对象的实现机制。一个 CORBA 对象的引用又称可互操作的对象引用（Interoperable Object Reference）。从客户程序的角度看，IOR 中包含了对对象的标识、接口类型及其他信息以查找对象实现。

对象标识（Object ID）是一个用于在 POA 中标识一个 CORBA 对象的字符串。它既可由程序员指派，也可由对象适配器自动分配，这两种方式都要求对象标识在创建它的对象适配器中必须具有唯一性。

POA（便携式对象适配器）是对象实现与 ORB 其它组件之间的中介，支持由 Object Id 标识的对象名称空间，它将客户请求传送到伺服对象，按需创建子 POA，提供管理伺服对象的策略。

伺服对象（servant）是指具体程序设计语言的对象或实体，通常存在于一个服务程序进程之中。客户程序通过对象引用发出的请求经过 ORB 担当中介角色，转换为对特定的伺服对象的调用。在一个 CORBA 对象的生命期中，它可能与多个伺服对象相关联，因而对该对象的请求可能被发送到不同的伺服对象。

十四 软件可靠性基础知识

1 软件可靠性基本概念

软件可靠性是软件产品在规定的条件下和规定的时间区间完成规定功能的能力。

软件可靠性和硬件可靠性区别

- （1）复杂性：软件复杂性比硬件高，大部分失效来自于软件失效。
- （2）物理退化：硬件失效主要是物理退化所致，软件不存在物理退化。
- （3）唯一性：软件是唯一的，每个 COPY 版本都一样，而两个硬件不可能完全一样。
- （4）版本更新周期：硬件较慢，软件较快。

软件可靠性的定量描述

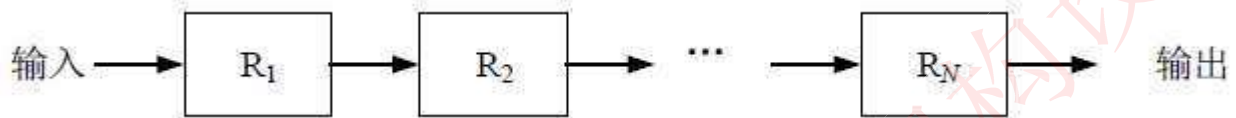
- 1.规定时间：自然时间、运行时间、执行时间（占用 CPU）。
- 2.失效概率：软件运行初始时为 0，随着时间增加单调递增，不断趋向于 1。
- 3.可靠度：软件系统在规定的条件下、规定的时间内不发生失效的概率。等于 1-失效概率。
- 4.失效强度：单位时间软件系统出现失效的概率。
- 5.平均失效前时间（MTTF）：平均无故障时间，发生故障前正常运行的时间。
- 6.平均恢复前时间（MTTR）：平均故障修复时间，发生故障后的修复时间。
- 7.平均故障间隔时间（MTBF）：失效或维护中所需的平均时间，包括故障时间以及检测和维护设备的时间。 $MTBF=MTTF+MTTR$ 。

系统可用性= $MTTF/(MTTF+MTTR)*100\%$ 。

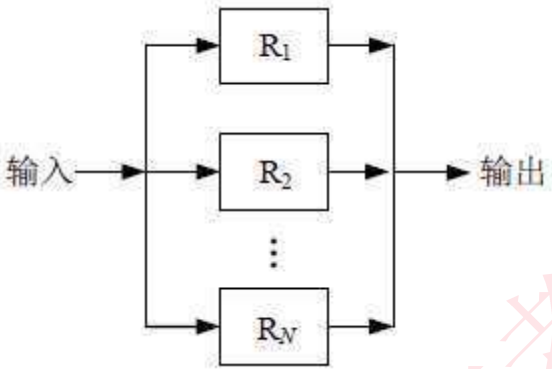
串并联系统可靠性

无论什么系统，都是由多个设备组成的，协同工作，而这多个设备的组合方式可以是串联、并联，也可以是混合模式，假设每个设备的可靠性为 $R_1, R_2, \dots, R_n$ ，则不同的系统的可靠性公式如下：

串联系统，一个设备不可靠，整个系统崩溃，整个系统可靠性 $R = R_1 * R_2 * \dots * R_n$ 。



并联系统，所有设备都不可靠，整个系统才崩溃，整个系统可靠性 $R = 1 - (1 - R_1) * (1 - R_2) * \dots * (1 - R_n)$ 。



对软件运行的影响程度不仅取决于软件失效发生的概率，还和软件失效的严重程度有很大关系。这里引出另外一个概念——失效严重程度类。

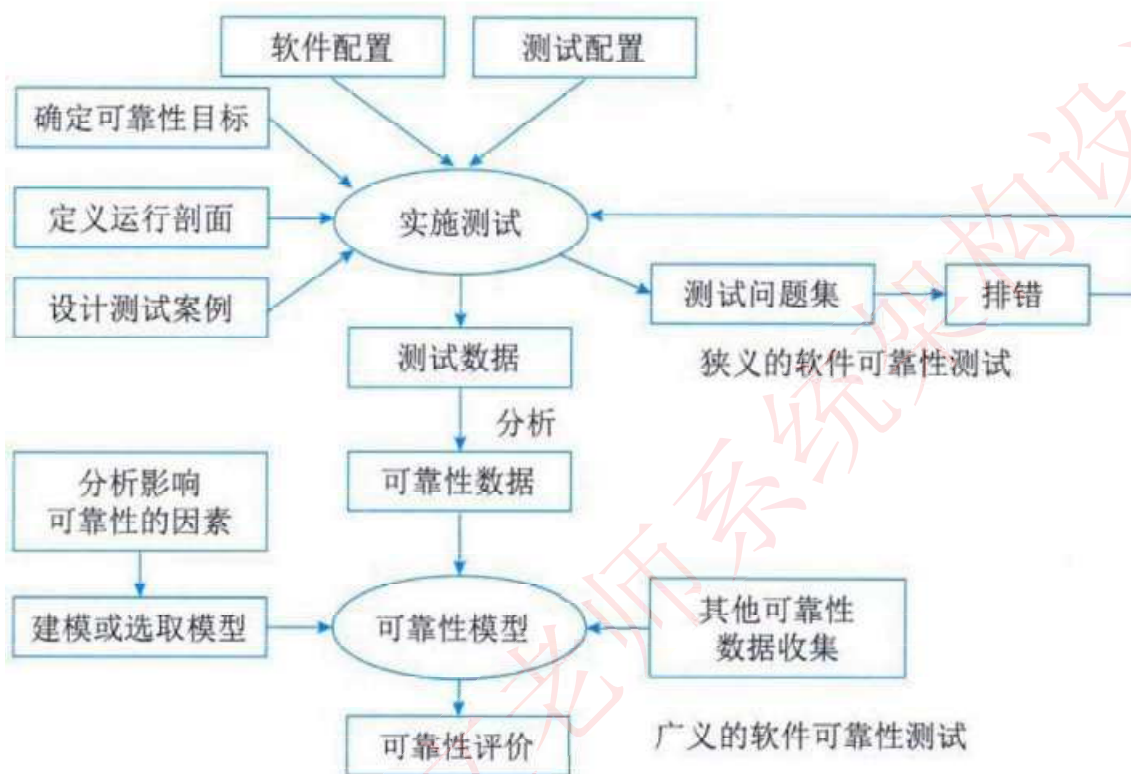
失效严重程度类就是对用户具有相同程度影响的失效集合。

对失效严重程度的分级可以按照不同的标准进行，最为常见的是按对成本影响、对系统能力的影响等标准划分软件失效的严重程度类。

可靠性测试的意义

- (1) 软件失效可能造成灾难性的后果。
- (2) 软件的失效在整个计算机系统失效中的比例较高。
- (3) 软件可靠性技术很不成熟，加剧了软件可靠性问题的重要性。
- (4) 软件可靠性问题是造成费用增长的主要原因之一。
- (5) 软件对生产活动和社会生活的影响越来越大，从而增加了软件可靠性问题在软件工程领域乃至整个计算机工程领域的重要性。

广义的软件可靠性测试是指为了最终评价软件系统的可靠性而运用建模、统计、试验、分析和评价等一系列手段对软件系统实施的一种测试。



狭义的软件可靠性测试是指为了获取可靠性数据，按预先确定的测试用例，在软件的预期使用环境中，对软件实施的一种测试。它是面向缺陷的测试，以用户将要使用的方式来测试软件。

2 软件可靠性建模

软件可靠性模型是指为预计或估算软件的可靠性所建立的可靠性框图和数学模型。

从技术的角度来看，影响软件可靠性的主要因素包括：运行环境、软件规模、软件内部结构、软件的开发方法和开发环境、软件的可靠性投入。

一个软件可靠性模型通常（但不是绝对）由以下几部分组成：

（1）模型假设。模型是实际情况的简化或规范化，总要包含若干假设，例如测试的选取代表实际运行剖面，不同软件失效独立发生等。

（2）性能度量。软件可靠性模型的输出量就是性能度量，如失效强度、残留缺陷数等。在软件可靠性模型中性能度量通常以数学表达式给出。

（3）参数估计方法。某些可靠性度量的实际值无法直接获得，例如残留缺陷数，这时需通过一定的方法估计参数的值，从而间接确定可靠性度量的值。

（4）数据要求。一个软件可靠性模型要求一定的输入数据，即软件可靠性数据。

绝大多数的模型包含 3 个共同假设：

（1）代表性假设。是指可以用测试产生的软件可靠性数据预测运行阶段的软件可靠性行为。

（2）独立性假设。此假设认为软件失效是独立发生于不同时刻，一个软件失效的发生不影响另一个软件失效的发生。

(3) 相同性假设。此假设认为所有软件失效的后果（等级）相同，即建模过程只考虑软件失效的具体发生时刻，不区分软件的失效严重等级。

人们常常通过估计或预测的方法来确定模型的参数。

确定了模型的参数后，就可以来表示失效过程的很多不同的特性。

软件的可靠性模型分类

1. 种子法模型。利用捕获一再捕获抽样技术估计程序中的错误数，在程序中预先有意“播种”一些设定的错误“种子”，然后根据测试出的原始错误数和发现的诱导错误的比例，来估计程序中残留的错误数。

2. 失效率类模型。用来研究程序的失效率。

3. 曲线拟合类模型。用回归分析的方法研究软件复杂性、程序中的缺陷数、失效率、失效间隔时间。

4. 可靠性增长模型。这类模型预测软件在检错过程中的可靠性改进，用增长函数来描述软件的改进过程。

5. 程序结构分析模型。是根据程序、子程序及其相互间的调用关系，形成一个可靠性分析网络。

6. 输入域分类模型。选取软件输入域中的某些样本“点”运行程序，根据这些样本点在“实际”使用环境中的使用概率的测试运行时的成功 / 失效率，推断软件的使用可靠性。

7. 执行路径分析方法模型。分析方法与上面的模型相似，先计算程序各逻辑路径的执行概率和程序中错误路径的执行概率，再综合出该软件的使用可靠性。

8. 非齐次泊松过程模型。是以软件测试过程中单位时间的失效次数为独立泊松随机变量，来预测在今后软件的某使用时间点的累计失效数。

9. 马尔可夫过程模型。

10. 贝叶斯模型。是利用失效率的试验前分布和当前的测试失效信息，来评估软件的可靠性。

3 软件可靠性管理

软件可靠性管理是软件工程管理的一部分，它以全面提高和保证软件可靠性为目标，以软件可靠性活动为主要对象，是把现代管理理论用于软件生命周期中的可靠性保障活动的一种管理形式。

软件可靠性管理的内容包括软件工程各个阶段的可靠性活动的目标、计划、进度、任务和修正措施等。可靠性各阶段设计任务如下：

需求分析阶段：确定可靠性目标、分析影响因素、确定验收标准、制定框架、制定文档编写规范、制定初步计划、确定数据收集规范。

概要设计阶段：确定可靠性度量、制定详细验收方案、可靠性设计、收集数据、调整计划、明确后续阶段详细计划、编制文档。

详细设计阶段：可靠性设计、预测、调整计划、收集数据、明确后续阶段详细计划、编制文档。

编码阶段：可靠性测试（单元）、排错、调整计划、收集数据、明确后续阶段详细计划、编制文档。

测试阶段：可靠性测试（集成和系统）、排错、可靠性建模、评价、调整计划、收集数据、明确后续阶段详细计划、编制文档。

实施阶段：可靠性测试（验收）、排错、收集数据、调整模型、评价、编制文档。

4 软件可靠性设计

实践证明，保障软件可靠性最有效、最经济、最重要的手段是在软件设计阶段采取措施进行可靠性控制。

可靠性设计其实就是在常规的软件设计中，应用各种方法和技术，使程序设计在兼顾用户的功能和性能需求的同时，全面满足软件的可靠性要求。

软件可靠性设计原则：

（1）软件可靠性设计是软件设计的一部分，必须在软件的总体设计框架中使用，并且不能与其他设计原则相冲突。

（2）软件可靠性设计在满足提高软件质量要求的前提下，以提高和保障软件可靠性为最终目标。

（3）软件可靠性设计应确定软件的可靠性目标，不能无限扩大化，并且排在功能度、用户需求和开发费用之后考虑。

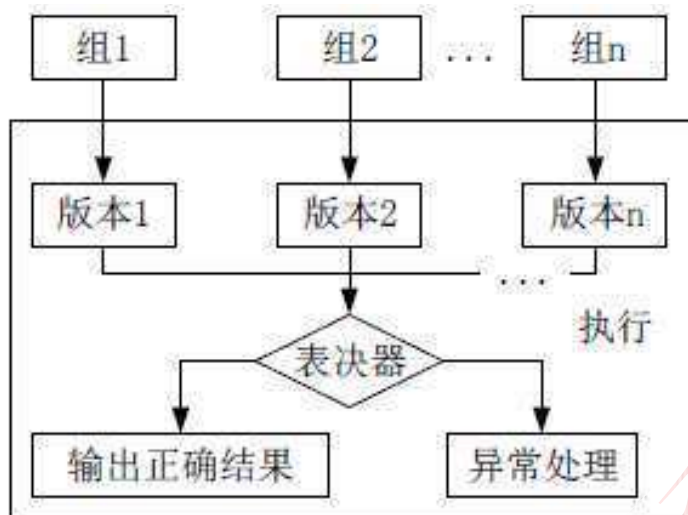
软件可靠性设计技术主要有容错设计、检错设计和降低复杂度设计等技术。

1. 容错设计技术。软件容错的主要方法是提供足够的冗余信息和算法程序，使系统在实际运行时能够及时发现程序设计错误，采取补救措施，以提高系统可靠性，保证整个系统的正常运行。软件容错技术主要有 N 版本程序设计、恢复块方法和冗余设计、防卫式程序设计等。

冗余设计

冗余是指在正常系统运行所需的基础上加上一定数量的资源，包括信息、时间、硬件和软件。冗余是容错技术的基础，通过冗余资源的加入，可以使系统的可靠性得到较大的提高。主要的冗余技术有结构冗余（硬件冗余和软件冗余）、信息冗余、时间冗余和冗余附加 4 种。

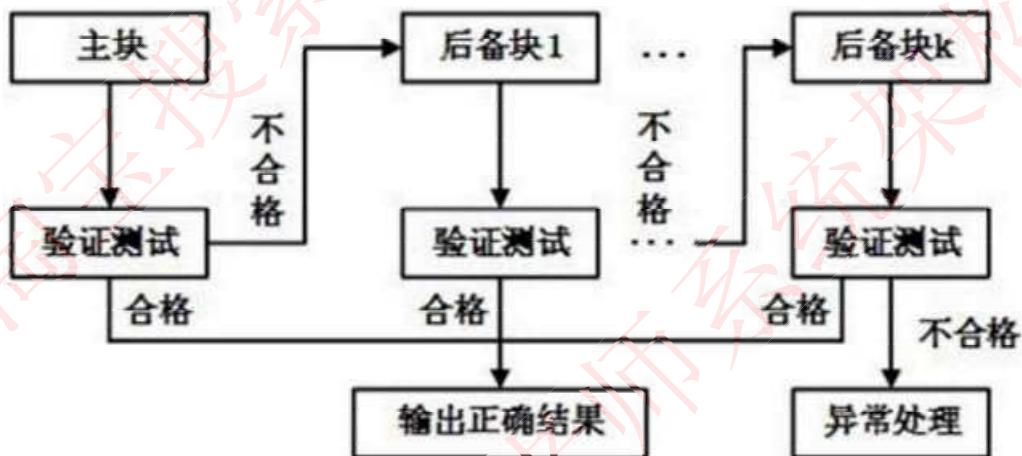
N 版本程序设计：是一种静态的故障屏蔽技术，其设计思想是用 N 个具有相同功能的程序同时执行一项计算，结果通过多数表决来选择。其中 N 个版本的程序必须由不同的人独立设计，使用不同的方法、设计语言、开发环境和工具来实现，目的是减少 N 个版本的程序在表决点上相关错误的概率。



与通常软件开发过程不同的是，N 版本程序设计增加了三个新的阶段：相异成分规范评审、相异性确认、背对背测试。

其他需要注意的问题：N 版本程序的同步、N 版本程序之间的通信、表决算法、一直比较问题、数据相异性。

恢复块设计（动态冗余）：动态冗余又称为主动冗余，它是通过故障检测、故障定位及故障恢复等手段达到容错的目的。其主要方式是多重模块待机储备，当系统检测到某工作模块出现错误时，就用一个备用的模块来替代它并重新运行。各备用模块在其待机时，可与主模块一样工作，也可以不工作。前者叫热备份系统（双重系统），后者叫冷备份系统（双工系统、双份系统）。



设计时应保证实现主块和后备块之间的独立性，避免相关错误的产生，使主块和备份块之间的共性错误降到最低程度。

二者比较

表 19-1 恢复块方法与N版本程序设计的比较

	恢复块方法	N 版本程序设计
硬件运行环境	单机	多机
错误检测方法	验证测试程序	表决
恢复策略	后向恢复	前向恢复
实时性	差	好

防卫式程序设计：是一种不采用任何传统的容错技术就能实现软件容错的方法，对于程序中存在的错误和不一致性，防卫式程序设计的基本思想是通过在程序中包含错误检查代码和错误恢复代码，使得一旦发生错误，程序就能撤销错误状态，恢复到一个已知的正确状态中去。其实现策略包括错误检测、破坏估计和错误恢复三个方面。

2. 检错技术：在软件出现故障后能及时发现并报警，提醒维护人员进行处理。代价低，但不能自动解决故障。

采用检错设计技术要着重考虑几个要素：检测对象、检测延时、实现方式和处理方式。

3. 降低复杂度设计：是在保证实现软件功能的基础上，简化软件结构，缩短程序代码长度，优化软件数据流向，降低软件复杂度，从而提高软件可靠性。

通常在**系统配置**中可以采用相应的容错技术，通过系统的整体来提供相应的可靠性，主要有双机热备技术和服务器集群技术。

双机容错技术：是一种软硬件结合的容错应用方案。该方案是由两台服务器和一个外接共享磁盘阵列及相应的双机软件组成。

双机容错系统采用“心跳”方法保证主系统与备用系统的联系。所谓心跳，是指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系统的心跳信号，则系统的高可用性管理软件认为主系统发生故障，立即将系统资源转移到备用系统上，备用系统替代主系统工作，以保证系统正常运行和网络服务不间断。

工作模式：双机热备模式；双机互备模式；双机双工模式。

集群技术就是将多台计算机组织起来进行协同工作，它是提高系统可用性和可靠性的一种技术。在集群系统中，每台计算机均承担部分计算任务和容错任务，当其中一台计算机出现故障时，系统使用集群软件将这台计算机从系统中隔出离去，通过各计算机之间的负载转嫁机制完成新的负载分担，同时向系统管理人员发出警报。集群系统通过功能整合和故障过渡，实现了系统的高可用性和可靠性。

特点：可伸缩性、高可用性、可管理性、高性价比、高透明性。

分类：高性能计算集群、负载均衡集群、高可用性集群。

负载均衡是集群系统中的一项重要技术，可以提高集群系统的整体处理能力，也提高了系统的可靠性，最终目的是加快集群系统的响应速度，提高客户端访问的成功概率。集群的最大特征是多个节点的并行和共同工作，如何让所有节点承受的负荷平均，不出现局部过大负载或过轻负载的情况，是负载均衡的重要目的。

比较常用的负载均衡实现技术主要有以下几种：

(1) 基于特定软件的负载均衡。很多网络协议都支持重定向功能，例如，基于 HTTP 重定向服务，其主要原理是服务器使用 HTTP 重定向指令，将一个客户端重新定位到另一个位置。服务器返回一个重定向响应，而不是返回请求的对象。客户端确认新地址然后重发请求，从而达到负载均衡的目的。

(2) 基于 DNS 的负载均衡属于传输层负载均衡技术，其主要原理是在 DNS 服务器中为同一个主机名配置多个地址，在应答 DNS 查询时，DNS 服务器对每个查询将以 DNS 文件中主机记录的 IP 地址按顺序返回不同的解析结果，将客户端的访问引导到不同的节点上去，使得不同的客户端访问不同的节点，从而达到负载均衡的目的。

(3) 基于 NAT 的负载均衡。将一个外部 IP 地址映射为多个内部 IP 地址，对每次连接需求动态地转换为一个内部节点的地址，将外部连接请求引到转换得到地址的那个节点，上从而达到负载均衡的目的。

(4) 反向代理负载均衡。将来自 Internet 上的连接请求以反向代理的方式动态地转发给内部网络上的多个节点进行处理，从而达到负载均衡的目的。

(5) 混合型负载均衡。

5 软件可靠性测试与评价

软件可靠性测试由可靠性目标的确定、运行剖面的开发、测试用例的设计、测试实施、测试结果的分析等主要活动组成。

测试步骤：定义软件运行剖面（为软件的使用行为建模）——设计可靠性测试用例——实施可靠性测试。

软件可靠性评价 3 个过程：选择可靠性模型、收集可靠性数据、可靠性评估和预测。

选择可靠性模型考虑因素：模型假设的适用性、预测的能力与质量、模型输出值能否满足可靠性评价需求、模型使用的简便性。

可靠性数据的收集：可靠性数据主要是指软件失效数据，是软件可靠性评价的基础，主要是在软件测试、实施阶段收集的。应采用的解决方法：及早确定所采用的的可靠性模型、制订可实施性较强的可靠性数据收集计划、重视软件测试数据的整理和分析、充分利用数据库来完成可靠性数据的存储和统计分析。

可靠性评估和预测：判断是否达到了可靠性目标；如未能达到要再投入多少；在软件系统投入实际运行一年或若干时间后，经过维护、升级和修改，软件能否达到交付或部分交付用户使用的可靠性水平。辅助方法：失效数据的图形分析法、试探性数据分析技术。

十五 软件架构的演化和维护

1 软件架构演化和定义

软件架构的演化和维护就是对架构进行修改和完善的过程，目的就是为了使软件能够适应环境的变化而进行的纠错性修改和完善性修改等，是一个不断迭代的过程，直至满足用户需求。

本质上讲，软件架构的演化就是软件整体结构的演化，演化过程涵盖软件架构的全生命周期，包括软件架构需求的获取、软件架构建模、软件架构文档、软件架构实现以及软件架构维护等阶段。

软件架构演化的重要性体现在：一是架构是整个系统的骨架，是软件系统具备诸多好的特性的保障；二是软件架构作为软件蓝图为人们宏观管控软件系统的整体复杂性和变化性提供了一条有效途径。

软件架构的演化能降低软件演化的成本的原因：

(1) 对系统的软件架构进行的形式化、可视化表示提高了软件的可构造性，便于软件演化。

(2) 软件架构设计方案涵盖的整体结构信息、配置信息、约束信息等有助于开发人员充分考虑未来可能出现的演化问题、演化情况和演化环境。

(3) 架构设计时对系统组件之间的耦合描述有助于软件系统的动态调整。

软件架构的定义包含组件、连接件、约束三大要素，这类软件架构演化主要关注的就是这三者之间的添加、修改和删除等。

2 面向对象软件架构演化

对象演化：在顺序图中，组件的实体是对象，会对架构设计的动态行为产生影响的演化只包括 AddObject (AO) 和 DeleteObject (DO) 两种。

AO 表示在顺序图中添加一个新的对象。这种演化一般是在系统需要添加新的对象来实现某种新的功能，或需要将现有对象的某个功能独立以增加架构灵活性的时候发生。

DO 删除顺序图中现有的一个对象。这种演化一般在系统需要移除某个现有的功能，或需要合并某些对象及其功能来降低架构的复杂度的时候发生。

消息演化：将消息演化分为 AddMessage(AM)、DeleteMessage (DM)、SwapMessageOrder (SMO)、OverturnMessage (OM)、ChangeMessageModule (CMM) 5 种。

AM 增添一条新的消息，产生在对象之间需要增加新的交互行为的时候。DM 删除当前的一条消息，产生在需要移除某个交互行为的时候，是 AM 的逆向演化。SMO 交换两条消息的时间顺序，发生在需要改变两个交互行为之间关系的时候。OM 反转消息的发送对象与接收对象，发生在需要修改某个交互行为本身的时候。CMM 改变消息的发送或接收对象，发生在需要修改某个交互行为本身的时候。

复合片段演化：复合片段是对象交互关系的控制流描述，表示可能发生在不同场合的交互，与消息同属于连接件范畴。复合片段的演化分为 **AddFragment(AF)**、**DeleteFragment(DF)**、**FragmentTypeChange (FTC)**和 **FragmentConditionChange (FCC)**。

FCC 改变复合片段内部执行的条件，发生在改变当前控制流的执行条件时。自动机中与控制流执行条件相对应的转移包括两个，一个是符合条件时的转移，另一个是不符合条件时的转移，因此每次发生 **FCC** 演化时会同时修改这两个转移的触发事件。

AF 在某几条消息上新增复合片段，发生在需要增添新的控制流时。复合片段所产生的分支是不同类型的，例如 **ref** 会关联到另一个顺序图，**par** 会产生并行消息，其余的则为分支过程。

DF 删除某个现有的复合片段，发生在需要移除当前某段控制流时。**DF** 与 **AF** 互为逆向演化过程。

FTC 改变复合片段的类型，发生在需要改变某段控制流时。类型演化意味着交互流程的改变，一般伴随着条件、内部执行序列的同时演化，可以视为复合片段的删除与添加的组合。

约束演化：即直接对约束信息进行添加和删除。

AC (Add Constraint)直接添加新的约束信息，会对架构设计产生直接的影响，需要判断当前设计是否满足新添加的约束要求。

DC (Delete Constraint)直接移除某条约束信息，发生在去除某些不必要条件的时候，一般而言架构设计均会满足演化后的约束。

3 软件架构演化方式的分类

3 种典型的分类方法：

(1) 按照软件架构的实现方式和实施粒度分类：基于过程和函数的演化、面向对象的演化、基于组件的演化和基于架构的演化。

(2) 按照研究方法将软件架构演化方式分为 4 类：第 1 类是对演化的支持，如代码模块化的准则、可维护性的指示（如内聚和耦合）、代码重构等；第 2 类是版本和工程的管理工具；第 3 类是架构变换的形式方法，包括系统结构和行为变换的模型，以及架构演化的重现风格等；第 4 类是架构演化的成本收益分析，决定如何增加系统的弹性。

(3) 针对软件架构的演化过程是否处于系统运行时期，可以将软件架构演化分为静态演化和动态演化。

软件架构的演化时期包括：设计时演化、运行前演化、有限制运行时演化、运行时演化。

软件架构静态演化主要是在设计时演化以及运行前演化。与此相对应的维护方法有 3 类：更正性维护、适应性维护和完善性维护。

软件的静态演化一般包括如下 5 个步骤。

- 软件理解：查阅软件文档，分析软件架构，识别系统组成元素及其之间的相互关系，提取系统的抽象表示形式。

- 需求变更分析：静态演化往往是由于用户需求变化、系统运行出错和运行环境发生改变等原因所引起的，需要找出新的软件需求与原有的差异。

- 演化计划：分析原系统，确定演化范围和成本，选择合适的演化计划。
- 系统重构：根据演化计划对系统进行重构，使之适应当前的需求。
- 系统测试：对演化后的系统进行测试，查找其中的错误和不足之处。

一次完整软件架构演化过程可以看作经过一系列原子演化操作组合而成。所谓原子演化操作是指基于 UML 模型表示的软件架构，在逻辑语义上粒度最小的架构修改操作。每经过一次原子演化操作，架构会形成一个演化中间版本。

架构演化的可维护性度量基于组件图表示的软件架构，在较高层次上评估架构的某个原子修改操作对整个架构所产生的影响。这些原子修改操作包括增加 / 删除模块间的依赖、增加 / 删除模块间的接口、增加 / 删除模块、拆分 / 聚合模块等。

架构演化的可靠性评估基于用例图、部署图和顺序图，分析在架构模块的交互过程中某个原子演化操作对交互场景的可靠程度的影响。这些原子修改操作包括增加 / 删除消息、增加 / 删除交互对象、增加 / 删除 / 修改消息片段、增加 / 删除用例执行、增加 / 删除角色等。

动态演化是在系统运行期间的演化，需要在不停止系统功能的情况下完成演化，较之静态演化更加困难。具体发生在有限制的运行时演化和运行时演化阶段。

架构的动态演化主要来自两类需求：

①软件内部执行所导致的体系结构改变，例如，许多服务器端软件会在客户请求到达时创建新的组件来响应用户需求；

②软件系统外部的请求对软件进行的重配置，例如，操作系统在升级时无须重新启动，在运行过程中就完成对体系结构的修改。

软件的动态性分为 3 个级别：

①交互动态性，要求数据在固定的结构下动态交互；

②结构动态性，允许对结构进行修改，通常的形式是组件和连接件实例的添加和删除，这种动态性是研究和应用的主流；

③架构动态性，允许软件架构的基本构造的变动，即结构可以被重定义，如新的组件类型的定义。

根据所修改的内容不同，软件的动态演化主要包括以下 4 个方面。

• 属性改名：目前所有的 ADL 都支持对非功能属性的分析和规约，而在运行过程中，用户可能会对这些指标进行重新定义（如服务响应时间）。

• 行为变化：在运行过程中，用户需求变化或系统自身服务质量的调节都将引发软件行为的变化。

• 拓扑结构改变：如增删组件，增删连接件，改变组件与连接件之间的关联关系等。

• 风格变化：一般软件演化后其架构风格应当保持不变，如果非要改变软件的架构风格，也只能将架构风格变为其衍生风格，如两层 C/S 到三层 C/S。

目前，实现软件架构动态演化的技术主要有两种：采用动态软件架构（DSA）和进行动态重配置（DR）。DSA 是指在运行时刻会发生变化的系统框架结构，允许在运行过程中通过框架结构的动态演化实现对架构的修改；DR 从组件和连接件的配置入手，允许在运行过程中增删组件，增删连接件，修改连接关系等操作。

实现软件架构动态演化的基本原理是使 DSA 在可运行应用系统中以一类有状态、有行为、可操作的实体显式地表示出来，并且被整个运行环境共享，作为整个系统运行的依据。也就是说，运行时体系结构相关信息的改变可用来触发、驱动系统自身的动态调整。

系统必须提供 SA 动态演化的一些相关功能：保存当前软件架构信息的功能、设置监控机制监视系统有无需求变化、保证演化操作原子性。

DSA 实施动态演化大体遵循以下 4 步：①捕捉并分析需求变化；②获取或生成体系结构演化策略；③根据步骤 2 得到的演化策略，选择适当的演化策略并实施演化；④演化后的评估与检测。

基于软件动态重配置的软件架构动态演化主要是指在软件部署之后对配置信息的修改，常常被用于系统动态升级时需要进行的配置信息修改。一般来说，动态重配置可能涉及的修改有：

- ①简单任务的相关实现修改；②工作流实例任务的添加和删除；③组合任务流程中的个体修改；④任务输入来源的添加和删除；⑤任务输入来源的优先级修改；⑥组合任务输出目标的添加和删除；⑦组合任务输出目标的优先级修改等。

动态重配置模式：主从模式、中央控制模式、客户端/服务器模式、分布式控制模式。

4 软件架构演化原则

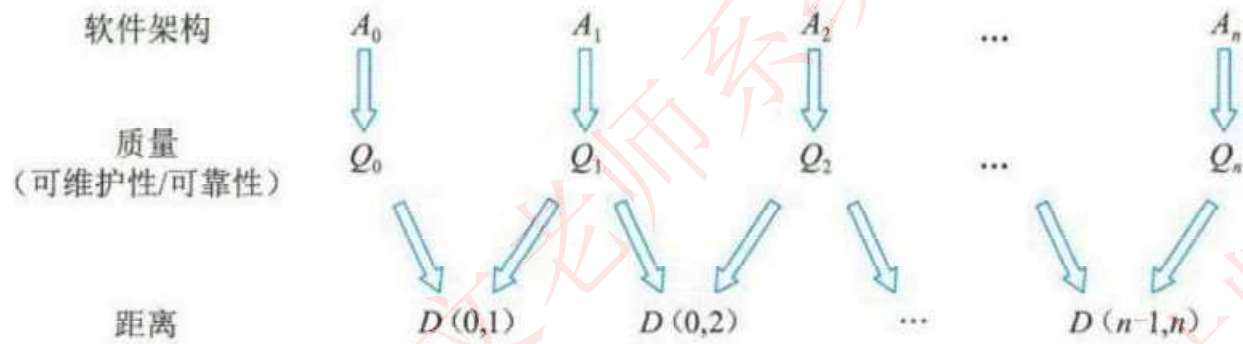
1. 演化成本控制原则
2. 进度可控原则
3. 风险可控原则
4. 主体维持原则：保证软件系统主体行为稳定。
5. 系统总体结构优化原则
6. 平滑演化原则：演化速率趋于稳定。
7. 目标一致原则
8. 模块独立演化原则
9. 影响可控原则
10. 复杂性可控原则
11. 有利于重构原则
12. 有利于重用原则
13. 设计原则遵从性原则：判断架构设计原则是否被破坏。
14. 适应新技术原则
15. 环境适应性原则
16. 标准依从性原则
17. 质量向好原则
18. 适应新需求原则

5 软件架构演化评估方法

根据演化过程是否已知可将评估过程分为：演化过程已知的评估和演化过程未知的评估。

演化过程已知的评估其目的在于通过对架构演化过程进行度量，比较架构内部结构上的差异以及由此导致的外部质量属性上的变化，对该演化过程中相关质量属性进行评估。

架构演化评估的执行过程如图所示。图中 A_0 和 A_n 表示一次完整演化前后的相邻版本的软件架构。每经过一次原子演化，即可得到一个架构中间演化版本 A_i 。对每个中间版本架构进行度量，得到架构 A_i 的质量属性度量值 Q_i 。 $D(i-1,i)$ 是版本间的质量属性距离。



基于度量的架构演化评估方法，其基本思路在于通过对演化前后的软件架构进行度量，比较架构内部结构上的差异以及由此导致的外部质量属性上的变化。具体包括：架构修改影响分析、监控演化过程、分析关键演化过程。

当演化过程未知时，我们无法像演化过程已知时那样追踪架构在演化过程中的每一步变化，只能根据架构演化前后的度量结果逆向推测出架构发生了哪些改变，并分析这些改变与架构相关质量属性的关联关系。

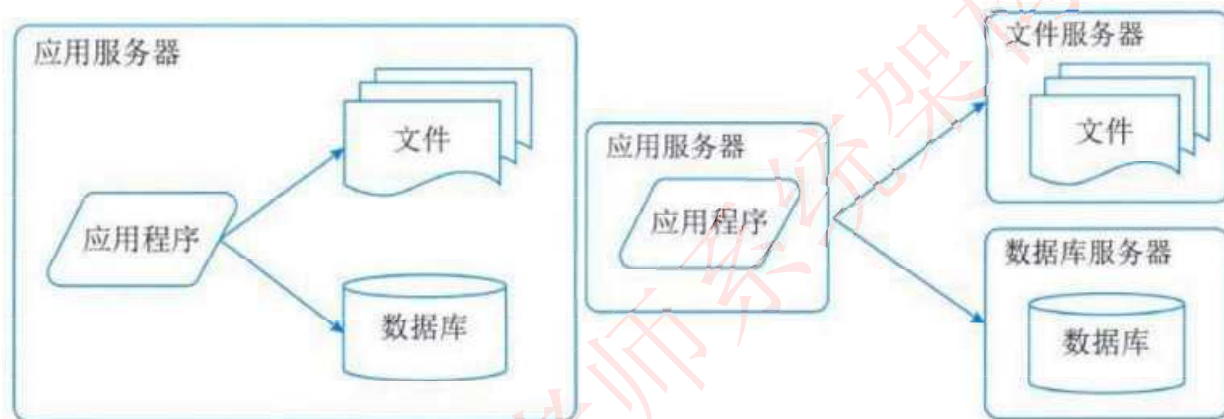


6 大型网站系统架构演化实例

主要解决大数据和高并发的问题。

第一阶段：单体架构。

第二阶段：垂直架构



第三阶段：使用缓存改善网站性能

第四阶段：使用服务集群改善网站并发处理能力

第五阶段：数据库读写分离

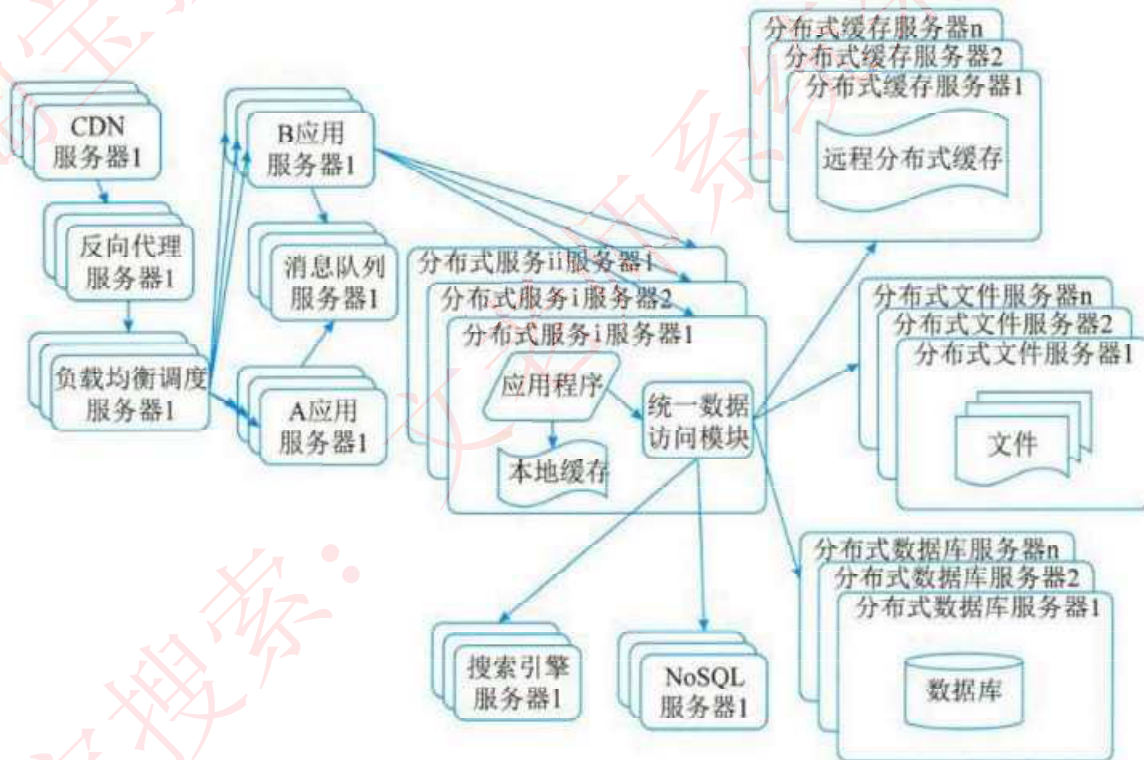
第六阶段：使用反向代理和 CDN 加速网站响应

第七阶段：使用分布式文件系统和分布式数据库系统

第八阶段：使用 NoSQL 和搜索引擎

第九阶段：业务拆分

第十阶段：分布式服务



7 软件架构维护

软件架构维护过程一般涉及架构知识管理、架构修改管理和架构版本管理。

软件架构知识管理是对架构设计中所隐含的决策来源进行文档化表示,进而在架构维护过程中帮助维护人员对架构的修改进行完善的考虑,并能够为其他软件架构的相关活动提供参考。

架构知识的定义:架构知识=架构设计+架构设计决策。即需要说明在进行架构设计时采用此种架构的原因。

架构知识管理侧重于软件开发和实现过程所涉及的架构静态演化,从架构文档等信息来源中捕捉架构知识,进而提供架构的质量属性及其设计依据以进行记录和评价。

在**软件架构修改管理**中,一个主要的做法就是建立一个隔离区域保障该区域中任何修改对其他部分的影响比较小,甚至没有影响。为此,需要明确修改规则、修改类型,以及可能的影响范围和副作用等。

软件架构版本管理为软件架构演化的版本演化控制、使用和评价等提供了可靠的依据,并为架构演化量化度量奠定了基础。

十六 未来信息综合技术

1 信息物理系统技术

信息物理系统(CPS)是控制系统、嵌入式系统的扩展与延伸,其涉及的相关底层理论技术源于对嵌入式技术的应用与提升。

CPS 通过集成先进的感知、计算、通信、控制等信息技术和自动控制技术,构建了物理空间与信息空间中人、机、物、环境、信息等要素相互映射、适时交互、高效协同的复杂系统,实现系统内资源配置和运行的按需响应、快速迭代、动态优化。

CPS 的本质就是构建一套信息空间与物理空间之间基于数据自动流动的状态感知、实时分析、科学决策、精准执行的闭环赋能体系,解决生产制造、应用服务过程中的复杂性和不确定性问题,提高资源配置效率,实现资源优化。

1. CPS 的体系架构

(1) 单元级 CPS。是具有不可分割性的 CPS 最小单元,是具备可感知、可计算、可交互、可延展、自决策功能的 CPS 最小单元,一个智能部件、一个工业机器人或一个智能机床都可能是一个 CPS 最小单元。

(2) 系统级 CPS。多个最小单元(单元级)通过工业网络(如工业现场总线、工业以太网等),实现更大范围、更宽领域的信息自动流动,实现了多个单元级 CPS 的互联、互通和互操作,进一步提高制造资源优化配置的广度、深度和精度。包含互联互通、即插即用、边缘网关、数据互操作、协同控制、监视与诊断等功能。其中互连互通、边缘网关和数据互操作主要实现单元级 CPS 的异构集

成；即插即用主要在系统级 CPS 实现组件管理，包括组（单元级 CPS）的识别，配置，更新和删除等功能；协同控制是指对多个单元级 CPS 的联动和协同控制等；监视与诊断主要是对单元级 CPS 的状态实时监控和诊断其是否具备应有的能力。

（3）SoS 级。多个系统级 CPS 的有机组合构成 SoS 级 CPS。例如，多个工序（系统的 CPS）形成一个车间级的 CPS 或者形成整个工厂的 CPS。主要实现数据的汇聚，从而对内进行资产的优化和对外形成运营优化服务。其主要功能包括：数据存储、数据融合、分布式计算、大数据分析、数据服务，并在数据服务的基础上形成了资产性能管理和运营优化服务。

2. CPS 的技术体系

CPS 技术体系主要分为 CPS 总体技术、CPS 支撑技术、CPS 核心技术。CPS 总体技术主要包括系统架构、异构系统集成、安全技术、试验验证技术等，是 CPS 的顶层设计技术；CPS 支撑技术主要包括智能感知、嵌入式软件、数据库、人机交互、中间件、SDN（软件定义网络）、物联网、大数据等，是基于 CPS 应用的支撑；CPS 核心技术主要包括虚实融合控制、智能装备、MBD、数字孪生技术、现场总线、工业以太网、CAX\MES\ERP\PLM\CRM\SCM 等，是 CPS 的基础技术。

上述技术体系可以分为四大核心技术要素即“一硬”（感知和自动控制）、“一软”（工业软件）、“一网”（工业网络）、“一平台”（工业云和智能服务平台）。其中感知和自动控制是 CPS 实现的硬件支撑；工业软件固化了 CPS 计算和数据流程的规则，是 CPS 的核心；工业网络是互联互通和数据传输的网络载体；工业云和智能服务平台是 CPS 数据汇聚和支撑上层解决方案的基础，对外提供资源管控和能力服务。

CPS 的典型应用场景：

（1）智能设计。在产品及工艺设计、工厂设计过程中的大部分工作都可以在虚拟空间中进行仿真，并实现迭代和改进。包括产品及工艺设计、生产线/工厂设计。

（2）智能生产。CPS 可以打破生产过程的信息孤岛现象，实现设备的互联互通，实现生产过程监控，合理管理和调度各种生产资源，优化生产计划，达到资源和制造协同，实现“制造”到“智造”的升级。包括设备管理、生产管理、柔性制造。

（3）智能服务。通过 CPS 按照需要形成本地与远程云服务相互协作、个体与群体、群体与系统的相互协同一体化工业云服务体系，能够更好地服务于生产，解决装备运行日益复杂、使用难度日益增大的困扰，实现智能装备的协同优化，支持企业用户经济性、安全性和高效性经营目标落地。包括健康管理、智能维护、远程征兆性诊断、协同优化、共享服务。

（4）智能应用。将设计者、生产者和使用者的单调角色转变为新价值创造的参与者，并通过新型价值链的创建反馈到产业链的转型，从根本上调动各个参与者的积极性，实现制造业转型。包括无人装备、产业链互动、价值链共赢。

CPS 建设路径：CPS 体系设计、单元级 CPS 建设、系统级 CPS 建设和 SoS 级 CPS 建设阶段。

2 人工智能技术

人工智能（AI）是利用数字计算机或者数字计算机控制的机器模拟、延伸和扩展人的智能，感知环境、获取知识并使用知识获得最佳结果的理论、方法、技术及应用系统。

人工智能的目标是了解智能的实质，并生产出一种新的能以人类智能相似的方式做出反应的智能机器。该领域的研究包括机器人、自然语言处理、计算机视觉和专家系统等。

根据人工智能是否能真正实现推理、思考和解决问题，可以将人工智能分为弱人工智能和强人工智能。

人工智能关键技术

1.自然语言处理（NLP）。研究实现人与计算机之间用自然语言进行有效通信的各种理论和方法。自然语言处理涉及的领域主要包括机器翻译（利用计算机实现从一种自然语言到另外一种自然语言的翻译）、语义理解（利用计算机理解文本篇章内容，并回答相关问题）和问答系统（让计算机像人类一样用自然语言与人交流）等。

2.计算机视觉。是使用计算机模仿人类视觉系统的科学，让计算机拥有类似人类提取、处理、理解和分析图像以及图像序列的能力，将图像分析任务分解为便于管理的小块任务。

3.知识图谱。就是把所有不同种类的信息连接在一起而得到的一个关系网络，提供了从“关系”的角度去分析问题的能力。一般用于反欺诈、不一致性验证等问题。

4. 人机交互（HCI）。主要研究人和计算机之间的信息交换，包括人到计算机和计算机到人的两部分信息交换

5.虚拟现实或增强现实（VR/AR）。以计算机为核心的新型视听技术。结合相关科学技术，在一定范围内生成与真实环境在视觉、听觉等方面高度近似的数字化环境

6.机器学习（ML）。是以数据为基础，通过研究样本数据寻找规律，并根据所得规律对未来数据进行预测。目前，机器学习广泛应用于数据挖掘、计算机视觉、自然语言处理、生物特征识别等领域。

广义上来说，机器学习指专门研究计算机怎么模拟或实现人类的学习行为以获取新的知识或技能的学科，使计算机重新组织已有的组织结构并不断改善自身的性能。

按照学习模式的不同，机器学习可分为监督学习、无监督学习、半监督学习、强化学习。其中，监督学习需要提供标注的样本集，无监督学习不需要提供标注的样本集，半监督学习需要提供少量标注的样本，而强化学习需要反馈机制。

监督学习是利用已标记的有限训练数据集，通过某种学习策略 / 方法建立一个模型，从而实现对新数据 / 实例的标记（分类） / 映射。在自然语言处理、信息检索、文本挖掘、手写体识别、垃圾邮件侦测等领域获得了广泛应用。

无监督学习是利用无标记的有限数据描述隐藏在未标记数据中的结构 / 规律。无监督学习不需要以人工标注数据作为训练样本，这样不仅便于压缩数据存储、减少计算量、提升算法速度，还可以避免正负样本偏移引起的分类错误问题。无监督学习主要用于经济预测、异常检测、数据挖掘、图像处理、模式识别等领域。

半监督学习可以利用少量的标注样本和大量的未标识样本进行训练和分类，从而达到减少标注代价、提高学习能力的目的。半监督学习的算法首先试图对未标识数据进行建模，在此基础上再对标识的数据进行预测。例如，图论推理算法或者拉普拉斯支持向量机等。

强化学习可以学习从环境状态到行为的映射，使得智能体选择的行为能够获得环境的最大奖赏，最终目标是使外部环境对学习系统在某种意义下的评价最佳。在机器人控制、无人驾驶、工业控制等领域获得成功应用。

按照学习方法的不同，机器学习可分为传统机器学习和深度学习。区别在于，传统机器学习的领域特征需要手动完成，且需要大量领域专业知识；深度学习不需要人工特征提取，但需要大量的训练数据集以及强大的 GPU 服务器来提供算力。

传统机器学习从一些观测（训练）样本出发，试图发现不能通过原理分析获得的规律，实现对未来数据行为或趋势的准确预测。在自然语言处理、语音识别、图像识别、信息检索等许多计算机领域获得了广泛应用。

深度学习是一种基于多层神经网络并以海量数据作为输入规则的自学习方法，依靠提供给它的大量实际行为数据（训练数据集），进行参数和规则调整。深度学习更注重特征学习的重要性。

机器学习的常见算法还包括迁移学习、主动学习和演化学习。

迁移学习是指当在某些领域无法取得足够多的数据进行模型训练时，利用另一领域数据获得的关系进行的学习。主要在变量有限的小规模应用中使用，如基于传感器网络的定位、文字分类和图像分类等。未来迁移学习将被广泛应用于解决更有挑战性的问题，如视频分类、社交网络分析、逻辑推理等。

主动学习通过一定的算法查询最有用的未标记样本，并交由专家进行标记，然后用查询到的样本训练分类模型来提高模型的精度。

演化学习基于演化算法提供的优化工具设计机器学习算法，针对机器学习任务中存在大量的复杂优化问题，应用于分类、聚类、规则发现、特征选择等机器学习与数据挖掘问题中。算法通常维护一个解的集合，并通过启发式算子来从现有的解产生新解，并通过挑选更好的解进入下一次循环，不断提高解的质量。

人工智能目前典型应用：chatgpt

3 机器人技术

机器人技术已经准备进入 4.0 时代。所谓机器人 4.0 时代，就是把云端大脑分布在各个地方，充分利用边缘计算的优势，提供高性价比的服务，把要完成任务的记忆场景的知识和常识很好地组合起来，实现规模化部署。特别强调机器人除了具有感知能力实现智能协作，还应该具有一定的理解和决策能力，进行更加自主的服务。

我们目前的服务机器人大多可以做到物体识别和人脸识别。在机器人 4.0 时代，我们需要加上更强的自适应能力。

机器人 4.0 的核心技术

1.云-边-端的无缝协同计算。云-边-端一体的机器人系统是面向大规模机器人的服务平台，信息处理和生成主要在云-边-端上分布处理完成。通常情况下，云侧可以提供高性能的计算和知识存储，边缘侧用来进一步处理数据并实现协同和共享。机器人端只用完成实时操作的功能。

2.持续学习与协同学习。希望机器人可以通过少量数据来建立基本的识别能力，然后可以自主地去找更多相关数据并进行自动标注。然后用这些自主得到的数据来对自己已有的模型进行重新训练来提高性能。

3.知识图谱。机器人应用的知识图谱应该具有的不同的需求：需要更加动态和个性化的知识；需要和机器人的感知与决策能力相结合。

4.场景自适应。通过对当前场景进行三维语义理解的基础上，主动观察场景内人和物之间的变化，预测可能发生的事件，从而影响之后的行动模式。这个技术的关键问题在于场景预测能力。场景预测就是机器人通过对场景内的各种人和物进行细致的观察，结合相关的知识和模型进行分析，并预测之后事件即将发生的时间，改变自己的行为模式。

5.数据安全。既要保证端到端的安全传输，也要保障服务器端的安全存储。

机器人的分类

如果按照要求的控制方式分类，机器人可分为操作机器人、程序机器人、示教再现机器人、智能机器人和综合机器人。

1. 操作机器人。典型代表是在核电站处理放射性物质时远距离进行操作的机器人。

2. 程序机器人。可以按预先给定的程序、条件、位置进行作业。

3.示教再现机器人。机器人可以将所教的操作过程自动地记录在磁盘、磁带等存储器中，当需要再现操作时，可重复所教过的动作过程。示教方法有直接示教与遥控示教两种。

4. 智能机器人。既可以进行预先设定的动作，还可以按照工作环境的改变而变换动作。

5. 综合机器人。由操纵机器人、示教再现机器人、智能机器人组合而成的机器人，如火星机器人。整个系统可以看作是由地面指令操纵的操作机器人。

如果按照应用行业来分，机器人可分为工业机器人、服务机器人和特殊领域机器人。

4 边缘计算

边缘计算将数据的处理、应用程序的运行甚至一些功能服务的实现，由网络中心下放到网络边缘的节点上。在网络边缘侧的智能网关上就近采集并且处理数据，不需要将大量未处理的原生数据上传到远处的大数据平台。

采用边缘计算的方式，海量数据能够就近处理，大量的设备也能实现高效协同的工作，诸多问题迎刃而解。因此，边缘计算理论上可满足许多行业在敏捷性、实时性、数据优化、应用智能，以及安全与隐私保护等方面的关键需求。

边缘计算的本质是云计算在数据中心之外汇聚节点的延伸和演进，主要包括云边缘、边缘云和云化网关三类落地形态；以“边云协同”和“边缘智能”为核心能力发展方向；软件平台需要考虑

导入云理念、云架构、云技术，提供端到端实时、协同式智能、可信赖、可动态重置等能力；硬件平台需要考虑异构计算能力。

(1) 云边缘：是云服务在边缘侧的延伸，逻辑上仍是云服务，主要的能力提供依赖于云服务或需要与云服务紧密协同。

(2) 边缘云：是在边缘侧构建中小规模云服务能力，边缘服务能力主要由边缘云提供；集中式DC侧的云服务主要提供边缘云的管理调度能力。

(3) 云化网关：以云化技术与能力重构原有嵌入式网关系统，云化网关在边缘侧提供协议 / 接口转换、边缘计算等能力，部署在云侧的控制器提供边缘节点的资源调度、应用管理与业务编排等能力。

边缘计算具有以下特点：

(1) 联接性：联接性是边缘计算的基础。所联接物理对象的多样性及应用场景的多样性，需要边缘计算具备丰富的联接功能。

(2) 数据第一入口：边缘计算作为物理世界到数字世界的桥梁，是数据的第一入口，拥有大量、实时、完整的数据，可基于数据全生命周期进行管理与价值创造，将更好的支撑预测性维护、资产效率与管理等创新应用。

(3) 约束性：边缘计算产品需适配工业现场相对恶劣的工作条件与运行环境，如防电磁、防尘、防爆、抗振动、抗电流 / 电压波动等。在工业互联网场景下，对边缘计算设备的功耗、成本、空间也有较高的要求。

(4) 分布性：边缘计算实际部署天然具备分布式特征。这要求边缘计算支持分布式计算与存储、实现分布式资源的动态调度与统一管理、支撑分布式智能、具备分布式安全等能力。

边云协同：边缘计算与云计算各有所长，云计算擅长全局性、非实时、长周期的大数据处理与分析，能够在长周期维护、业务决策支撑等领域发挥优势；边缘计算更适用局部性、实时、短周期数据的处理与分析，能更好地支撑本地业务的实时智能化决策与执行。

边缘计算既靠近执行单元，更是云端所需高价值数据的采集和初步处理单元，可以更好地支撑云端应用；反之，云计算通过大数据分析优化输出的业务规则或模型可以下发到边缘侧，边缘计算基于新的业务规则或模型运行。

主要包括六种协同：

(1) 资源协同：边缘节点提供计算、存储、网络、虚拟化等基础设施资源、具有本地资源调度管理能力，同时可与云端协同，接受并执行云端资源调度管理策略，包括边缘节点的设备管理、资源管理以及网络连接管理。

(2) 数据协同：边缘节点主要负责现场 / 终端数据的采集，按照规则或数据模型对数据进行初步处理与分析，并将处理结果以及相关数据上传给云端；云端提供海量数据的存储、分析与价值挖掘。边缘与云的数据协同，支持数据在边缘与云之间可控有序流动，形成完整的数据流转路径，高效低成本对数据进行生命周期管理与价值挖掘。

(3) 智能协同：边缘节点按照 AI 模型执行推理，实现分布式智能；云端开展 AI 的集中式模型训练，并将模型下发边缘节点。

(4) 应用管理协同：边缘节点提供应用部署与运行环境，并对本节点多个应用的生命周期进行管理调度；云端主要提供应用开发、测试环境，以及应用的生命周期管理能力。

(5) 业务管理协同：边缘节点提供模块化、微服务化的应用 / 数字孪生 / 网络等应用实例；云端主要提供按照客户需求实现应用 / 数字孪生 / 网络等的业务编排能力。

(6) 服务协同：边缘节点按照云端策略实现部分 ECSaaS 服务，通过 ECSaaS 与云端 SaaS 的协同实现面向客户的按需 SaaS 服务；云端主要提供 SaaS 服务在云端和边缘节点的服务分布策略，以及云端承担的 SaaS 服务能力。

边缘计算的应用场合：智慧园区、安卓云与云游戏、视频监控、工业互联网、Cloud VR。

5 数字孪生体技术

数字孪生体技术是跨层级、跨尺度的现实世界和虚拟世界建立沟通的桥梁。

数字孪生体是现有或将有的物理实体对象的数字模型，通过实测、仿真和数据分析来实时感知、诊断、预物理实体对象的状态，通过优化和指令来调控物理实体对象的行为，通过相关数字模型间的相互学习来进化自身，同时改进利益相关方在物理实体对象生命周期内的决策。

关键技术

1.建模。建模的目的是将我们对物理世界的理解进行简化和模型化。而数字孪生体的目的或本质是通过数字化和模型化，用信息换能量，以使少的能量消除各种物理实体、特别是复杂系统的不确定性。需求指标、生存期阶段和空间尺度构成了数字孪生体建模技术体系的三维空间。

2.仿真。从技术角度看，建模和仿真是一对伴生体。如果说建模是模型化我们对物理世界或问题的理解，那么仿真就是验证和确认这种理解的正确性和有效性。所以，数字化模型的仿真技术是创建和运行数字孪生体、保证数字孪生体与对应物理实体实现有效闭环的核心技术。

仿真是将包含了确定性规律和完整机理的模型转化成软件的方式来模拟物理世界的一种技术。只要模型正确，并拥有了完整的输入信息和环境数据，就可以基本准确地反映物理世界的特性和参数。

3.其他技术。VR、AR 以及 MR 等增强现实技术、数字线程、系统工程和 MB SE、物联网、云计算、雾计算、边缘计算、大数据技术、机器学习和区块链技术。

数字孪生体主要应用于制造、产业、城市和战场。

6 云计算和大数据技术

云计算概念的内涵包含两个方面：平台和应用。平台即基础设施，其地位相当于 PC 上的操作系统，云计算应用程序需要构建在平台之上；云计算应用所需的计算与存储通常在“云端”完成，客户端需要通过互联网访问计算与存储能力。

云计算的服务方式

1) 软件即服务 (SaaS)。在 SaaS 的服务模式下, 服务提供商将应用软件统一部署在云计算平台上, 客户根据需要通过互联网向服务提供商订购应用服务, 服务提供商根据客户所订购软件的数量、时间的长短等因素收费, 并且通过标准浏览器向客户提供应用服务。

2) 平台即服务 (PaaS)。在 PaaS 模式下, 服务提供商将分布式开发环境与平台作为一种服务来提供。这是一种分布式平台服务, 厂商提供开发环境、服务器平台、硬件资源等服务给客户, 客户在服务提供商平台的基础上定制开发自己的应用程序, 并通过其服务器和互联网传递给其他客户。

3) 基础设施即服务 (IaaS)。在 IaaS 模式下, 服务提供商将多台服务器组成的“云端”基础设施作为计量服务提供给客户。具体来说, 服务提供商将内存、I/O 设备、存储和计算能力等整合为一个虚拟的资源池, 为客户提供所需要的存储资源、虚拟化服务器等服务。

在灵活性方面, SaaS → PaaS → IaaS 灵活性依次增强。

在方便性方面, IaaS → PaaS → SaaS 方便性依次增强。

云计算的部署模式

1) 公有云。在公有云模式下, 云基础设施是公开的, 可以自由地分配给公众。企业、学术界与政府机构都可以拥有和管理公用云, 并实现对公有云的操作。公有云能够以低廉的价格为最终用户提供有吸引力的服务, 创造新的业务价值。

2) 社区云。在社区云模式下, 云基础设施分配给一些社区组织所专有, 这些组织共同关注任务、安全需求、政策等信息。云基础设施被社区内的一个或多个组织所拥有、管理及操作。“社区云”是“公有云”范畴内的一个组成部分。

3) 私有云。在私有云模式下, 云基础服务设施分配给由多种用户组成的单个组织。它可以被这个组织或其他第三方组织所拥有、管理及操作。

4) 混合云。混合云是公有云、私有云和社区云的组合。由于安全和控制原因, 并非所有的企业信息都能放置在公有云上, 因此企业将会使用混合云模式。

大数据是指其大小或复杂性无法通过现有常用的软件工具, 以合理的成本并在可接受的时限内对其进行捕获、管理和处理的数据集。这些困难包括数据的收入、存储、搜索、共享、分析和可视化。

大数据的特点: 大规模、高速度、多样化、可变性、复杂性等。

大数据分析的分析步骤, 大致分为数据获取 / 记录、信息抽取 / 清洗 / 注记、数据集成 / 聚集 / 表现、数据分析 / 建模和数据解释 5 个主要阶段。

大数据的应用领域: 制造业、服务业、交通行业、医疗行业。

十七 补充：知识产权和标准化

1 知识产权基础知识

知识产权是指公民、法人、非法人单位对自己的创造性智力成果和其他科技成果依法享有的民事权利。是智力成果的创造人依法享有的权利和在生产经营活动中标记所有人依法所享有的权利的总称。包含著作权、专利权、商标权、商业秘密权、植物新品种权、集成电路布图设计权和地理标志权等。

无体性：知识产权的对象是没有具体形体，是智力创造成果，是一种抽象的财富。

专有性：指除权利人同意或法律规定外，权利人以外的任何人不得享有或使用该项权利。

地域性：指知识产权只在授予其权利的国家或确认其权利的国家产生，并且只能在该国范围内受法律保护，而其他国家则不受保护。

时间性：仅在法律规定的期限内受到保护，一旦超过期限，权利自行消灭，相关知识产品即成为整个社会的共同财富，为全人类所共同使用。

知识产权具有地域限制，保护期限各种情况如下表所示：

客体类型	权力类型	保护期限
公民作品	署名权、修改权、保护作品完整权	没有限制
	发表权、使用权和获得报酬权	作者终生及其死亡后的50年（第50年的12月31日）
单位作品	发表权、使用权和获得报酬权	50年（首次发表后的第50年的12月31日），若其间未发表，不保护。
公民软件产品	署名权、修改权	没有限制
	发表权、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	作者终生及死后50年（第50年12月31日）。合作开发，以最后死亡作者为准。
单位软件产品	发表权、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	50年（首次发表后的第50年的12月31日），若其间未发表，不保护
注册商标		有效期10年（若注册人死亡或倒闭1年后，未转移则可注销，期满后6个月内必须续注）
发明专利权		保护期为20年（从申请日开始）
实用新型和外观设计专利权		保护期为10年（从申请日开始）
商业秘密		不确定，公开后公众可用

知识产权人的确定

（1）职务作品

情况说明		判断说明	归属
作品	职务作品	利用单位的物质技术条件进行创作，并由单位承担责任的	除署名权外其他著作权归单位
		有合同约定，其著作权属于单位	除署名权外其他著作权归单位
		其他	作者拥有著作权，单位有权在业务范围内优先使用
软件	职务作品	属于本职工作中明确规定的开发目标	单位享有著作权
		属于从事本职工作活动的结果	单位享有著作权
		使用了单位资金、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	单位享有著作权
专利权	职务作品	本职工作中作出的发明创造	单位享有专利
		履行本单位交付的本职工作之外的任务所作出的发明创造	单位享有专利
		离职、退休或调动工作后1年内，与原单位工作相关	单位享有专利

(2) 委托作品

单位和委托的区别在于，当合同中未规定著作权的归属时，著作权默认归于单位，而委托创作中，著作权默认归属于创作方个人，具体如下：

情况说明		判断说明	归属
作品 软件	委托 创作	有合同约定，著作权归委托方	委托方
		合同中未约定著作权归属	创作方
	合作 开发	只进行组织、提供咨询意见、物质条件 或者进行其他辅助工作	不享有著作权
		共同创作的	共同享有，按人头比例。 成果可分割的，可分开申请。
商标		谁先申请谁拥有（除知名商标的非法抢注） 同时申请，则根据谁先使用（需提供证据） 无法提供证据，协商归属，无效时使用抽签（但不可不确定）	
专利		谁先申请谁拥有 同时申请则协商归属，协商未果则驳回所有申请	

侵权判定

相关概念如下：即一般通用化的东西不算侵权，个人未发表的东西被抢先发表是侵权。

中国公民、法人或者其他组织的作品，不论是否发表，都享有著作权。

开发软件所用的思想、处理过程、操作方法或者数学概念不受保护。

著作权法不适用于下列情形：

法律、法规、国家机关的决议、决定、命令和其他具有立法、行政、司法性质的文件，及其官方正式译文；时事新闻；历法、通用数表、通用表格和公式。

只要不进行传播、公开发表、盈利都不算侵权，具体如下：

不侵权	侵权
✓ 个人学习、研究或者欣赏； ✓ 适当引用； ✓ 公开演讲内容 ✓ 用于教学或科学研究 ✓ 复制馆藏作品； ✓ 免费表演他人作品； ✓ 室外公共场所艺术品临摹、绘画、摄影、录像； ✓ 将汉语作品译成少数民族语言作品或盲文出版。	✓ 未经许可，发表他人作品； ✓ 未经合作作者许可，将与他人合作创作的作品当作自己单独创作的作品发表的； ✓ 未参加创作，在他人作品署名； ✓ 歪曲、篡改他人作品的； ✓ 剽窃他人作品的； ✓ 使用他人作品，未付报酬； ✓ 未经出版者许可，使用其出版的图书、期刊的版式设计的。

2 标准化基础知识

根据标准制定机构和适用范围的不同，可分为国际标准、国家标准、行业标准、区域/地方标准和企业标准；根据类型划分，又可以分为强制性标准和推荐性标准。

(1) 国际标准：是指国际标准化组织（ISO）、国际电工委员会（IEC）和国际电信联盟（ITU）制定的标准，以及国际标准化组织确认并公布的其他国际组织制定的标准。国际标准在世界范围内统一使用，提供各国参考。

(2) 国家标准：是指由国家标准化管理机构制定或批准发布，在全国范围内统一适用的标准。比如：GB——中华人民共和国国家标准；强制性国家标准代号为 GB，推荐性国家标准代号为 GB/T，国家标准指导性文件代号为 GB/Z，国军标代号为 GJB。

ANSI (American National Standards Institute) ——美国国家标准协会标准；

(3) 行业标准：是由某个行业机构、团体等制定的，适用于某个特定行业业务领域的标准。比如：IEEE——美国电气电子工程师学会标准；GA——公共安全标准；YD——通信行业标准；

(4) 区域/地方标准：是由某一区域/地方内的标准化主管机构制定、批准发布的，适用于某个特定区域/地方的标准。比如：EN——欧洲标准；

(5) 企业标准：是企业范围内根据需要协调、统一的技术要求、管理要求和工作要求所制定的标准，适用于本企业内部的标准。一般以 Q 字开头，比如 Q/320101 RER 007—2012，其中 320101 代表地区，RER 代表企业名称代号，001 代表该企业该标准的序号，2012 代表年号。

《中华人民共和国标准化法》规定：企业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。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。

十八 补充：数学与经济管理

本章节内容涉及到大量数学逻辑和计算问题，详细真题讲解见文老师系统架构设计师视频课程。

1 图论应用

1.1 最小生成树

有两种方法：普里姆算法和克鲁斯卡尔算法，实际计算建议采用克鲁斯卡尔算法。

克鲁斯卡尔算法：将图中所有的边按权值从小到大排序，从**权值最小的边**开始选取，判断**是否为安全边（即不构成环）**，直至选取了 $n-1$ 条边，构成了最小生成树。

最小生成树并不唯一，但权值之和都相等且最小，只要求出一个就可以。

1.2 最短路径

计算从起点到终点的最短路径，注意与关键路径截然相反，不要混淆。

方法：从起点开始，依次向终点推导，每个经过的中间节点都直接计算出到该中间节点最短的路径，这样递归推导到终点，就是最短路径。

1.3 网络与最大流量

计算从一个节点到另一个节点的最大运输能力，取决于节点之间运输能力的短板：首先看有多少条路径可走，最大流量等于**所有路径最大流量之和**，而每条路径的最大流量是节点之间**运输能力最小的流量决定的**（短板决定）。

解题技巧：

- 1、取每条路径上的最小权值即为此条路径的最大流量，每次走完一条路径后，需要**实时修改此条路径还剩下的运输流量值**，若为 0，则删掉此连线。
- 2、重复第 1 步，直至从起点到终点无路径连通，而后将每条路径上的流量相加得到整体最大流量。

2 运筹方法

2.1 线性规划

在一组约束条件下来寻找目标函数的极值（极大值和极小值）问题。

线性规划问题的数学模型通常由线性目标函数、线性约束条件、变量非负条件组成（实际问题中的变量一般都是非负的）。

线性规划问题就是面向实际应用，求解一组非负变量，使其满足给定的一组线性约束条件，并使某个线性目标函数达到极值。满足这些约束条件的非负变量组的集合称为可行解域。可行解域中使目标函数达到极值的解称为最优解。

线性规划问题的最优解要么是 0 个（没有），要么是唯一的（1 个），要么有无穷个（只要有 2 个，就会有无穷个）。

在实际应用中，可以**直接求约束条件方程组的解**，即是交叉点，将这些解代入到目标函数中判断是否极值即可。

2.2 动态规划

求解给定多种方案求出最佳方案的问题。

直接运用动态规划法的原理来解题非常复杂，因此一般采用穷举法解题：

依据真题规则，将题目给出的所有投资方案全部穷举出来，所有方案获得的收益一目了然，然后取最佳投资方案，简单明了，不容易错。

2.3 伏格尔法

针对多种解决方法问题，如多个煤场供给多个工厂的运输成本。

从正常思维来思考，在没有任何约束条件的情况下，我们会优先考虑运输成本最低的方案，这就是最小元素法，但是当有多个制约因素时，最小元素法的缺点是，为了节约一处的费用，有时造成在其他处要多花几倍的运费。

因此，多个因素互相制约时，普遍采用伏格尔法又称差值法，该方法考虑到，某产地的产品如不能按最小运费就近供应，就考虑次小运费，这就有一个差额。差额越大，说明不能按最小运费调运时，运费增加越多。因而对差额最大处，就应当采用最小运费调运。

由上述原理，可得出其解题步骤为：

- 1、计算出每行每列的最小运费和次小运费的差值（绝对值）。
- 2、从这些差值里选出最大的的行（列），定位到该行（列），从该行（列）中找出最小的那一个，就是优先供应的方案。
- 3、供应后，更新供应量和需求量，如果某行（列）的供应量和需求量为 0，则删除该行列。
- 4、形成一个人新的表格，重复上述步骤。

详细例题解析见文老师视频课程讲解。

2.4 博弈论

研究具有竞争和对抗性质的行为的数学理论和方法。参加竞争的各方都有各自的目标和利益，各方为了达到自己的目标，必须要研究对手的资料，博弈论就是研究如何在多方中寻找出一个最合理的方案。

参与博弈的双方处于竞争状态，双方都**不能信任对方**，因此都**只会采用对自己有用的方案**，而这种方法对双方来说**并不是最优的**，但因为双方都不信任，要只能这样。如下所示：

		乙网站	
		高价	低价
甲网站	高价	1000, 1000	200, 1500
	低价	1500, 200	700, 700

2.5 状态转移矩阵

两个产品之间的状态会互相转变，给出一个状态转换矩阵，表明了两个产品随着时间的变化该如何转变，求解一定时间后两个产品的状态。

解题技巧：当前两个产品的状态依赖于之前一个阶段的状态以及状态转移矩阵系数。

2.6 排队论

求解排队问题，即检票和排队速度不对等时，如何达到一个平衡的问题。

解题技巧：要学会假设，一般对检票速度、排队人数都要做出未知数假设，然后列出多个方程求解出未知数之间的关系，再进行推导。

2.7 决策论

按决策环境分类

确定型决策：决策环境是确定的，结果也是确定的。

风险决策：决策环境是不确定的，但是结果发生的**概率**是一致的。

不确定型决策：决策环境不确定，且结果也不确定，完全凭主观意识来决定。

决策的六个要素

决策者、可供选择的方案（包括行动、策略）、衡量选择方案的准则（目的、目标、正确性等）、事件（被决策的对象）、每一事件的发生将会产生的某种结果、决策者的价值观。

不确定型决策

决策者对环境一无所知，任意猜测，完全凭借与决策者自身价值观，有五种准则（**建立环境与方案的表格，每个环境每种方案对应一个收益**）：

悲观主义准则（小中取大 $\max(\min)$ ，**先取每个方案最小的收益**，再取所有最小收益中最大的那个）；

乐观主义准则（大中取大 $\max(\max)$ ，先取每个方案最大的收益，再取所有最大收益中最大的那个）；

折中主义准则（设定折中系数 a ，用每个方案的最大收益 $*a$ +最小收益 $*(1-a)$ ，选择每个方案中计算**结果最大的那个**，可知， $a=1$ 时为乐观主义， $a=0$ 时为悲观主义）；

等可能性准则（设定每个可能的结果的发生都是等可能的，这样就知道每个结果发生的概率，即将不确定型的问题**转换为了风险决策问题**）；

后悔值准则（最小最大后悔值 $\min(\max)$ ，在**不同的环境中（之前都是方案）**，投资方案获得的最大收益-当前选择的收益=后悔值，将**所有后悔值中**每个方案的最大后悔值选出，再从这些最大的后悔值中选择最小的即可）。

决策矩阵：

预计收益(单位万元人民币)		经济趋势预测		
		不景气	不变	景气
投资策略	积极	50	150	500
	稳健	100	200	300
	保守	300	250	200

后悔值矩阵：

预计收益(单位万元人民币)		经济趋势预测		
		不景气	不变	景气
投资策略	积极	250	100	0
	稳健	200	50	200
	保守	0	0	300

风险决策（决策树和决策表）

对环境不了解，但是对即将发生的结果的概率了解，一般题目会给出每个结果对应的收益及风险概率，然后将**每种结果产生的收益*此种结果发生的概率**，取收益最大的即可，与风险曝光度类似。

3 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。

数学建模过程

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。

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。

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。

模型求解：利用获取的数据资料，对模型的所有参数做出计算（估计）。

模型分析：对所得的结果进行数学上的分析。

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。

模型应用：应用方式因问题的性质和建模的目的而异。

数学建模方法

直接分析法：根据对问题直接的内在的认识，直接构造出模型。

类比法：根据之前类似的模型构造出一个新的模型。

数据分析法：通过实验获得与问题相关的大量数据，用统计分析的方法来进行建模。

构想法：对将来可能发生的情况给出逻辑上合理的方法和描述，而后用现有的方法来建模，然后不断的完善。