# 计算机病毒

根据感染的途径及采用的技术区分，可分为：

文件型：感染可执行文件（包括EXE和COM文件）

引导型：影响软盘或硬盘的引导扇区

宏病毒：感染的对象是使用某些程序创建的文本文档、数据库、电子表格等文件

目录型：修改硬盘上存储的所有文件的地址

# 中断

在中断时，CPU断点信息一般保存到栈中。

# 异步串行通信

在系统间进行异步串行通信时，数据的串/并和并/串转换一般是通过接口中的移位寄存器实现的。

# 流水线执行时间

理论公式：nt（t为流水线的周期，为最长一段的时间，n为指令条数）

实际公式：kt+(n-1)t（k为第一条指令执行时间，称为建立时间）

# 计算机性能指标

时钟频率、数据处理速率、运算精度、内存容量

# 数据加密算法

DES密钥长度为56位，3DES的密钥长度为112位

# 企业应用集成

企业应用集成技术可以消除信息孤岛，他将多个企业信息系统连接起来，实现无缝集成，使他们就像一个整体一样。EAI所连接的应用包括各种电子商务系统、ERP、CRM、SCM、OA、数据库系统和数据仓库等。EAI可以包括表示集成、数据集成、控制集成和业务流程集成等多个层次和方面，也可以在多个企业之间进行应用集成。

数据集成是白盒集成，在集成之前必须先对数据进行标识并编成目录，另外还要确定数据模型，保证数据在数据库系统中分布和共享。

控制集成也称为功能集成或应用集成，属于黑盒集成，它是在业务逻辑层上对应用系统进行集成的。

业务流程集成也称为过程集成，它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，以便改进操作、减少成本、提高响应速度。

企业应用集成通过采用多种集成模式，构建统一标准的基础平台，将具有不同功能的目的而又独立运行的企业信息系统联合起来。目前市场上主流的集成模式有3种：面向信息的集成、面向过程的集成和面向服务的集成。其中面向过程的集成模式强调处理不同应用系统之间的交互逻辑，与核心业务逻辑相分离，并通过不同应用系统之间的协作共同完成某项业务功能。

# 客户关系管理

CRM是一种以客户位中心的商业策略，注重的是与客户的交流，企业的经营是以客户为中心，而不是传统的以产品或以市场为中心。在注重提高客户的满意度的同时，一定要把帮助企业提高获取利润的能力作为重要指标。CRM的实施要求企业对其业务功能进行重新设计，并对工作流程进行重组，将业务的中心转移到客户，同时要针对不同的客户群体有重点的采取不同的策略。

CRM是一套先进的管理思想及技术手段，它通过将人力资源、业务流程与专业技术进行有效的整合，最终为企业涉及到客户或者消费者的各个领域提供了完美的集成，使得企业可以更低成本、更高效率地满足客户的需求，并与客户建立起基于学习性关系基础上的一对一营销模式，从而让企业可以最大程度的提高客户满意度和忠诚度。

# RFID

射频标签也叫电子标签、射频识别。他是一种非接触式的自动识别技术，通过射频信号识别目标对象并获取相关数据。识别工作无须人工干预。作为条形码的无线版本，RFID技术具有条形码所不具备的防水、防磁、耐高温、使用寿命长、读取距离大、标签上数据可以加密、存储数据容量更大、存储信息更改自如、更容易附着在不同产品上等优点。

# 电子商务

电子商务（Electronic Commerce，CE）是指买卖双方利用现代开放的Internet，按照一定的标准所进行的各类商业活动。主要包括网上购物，企业之间的网上交易和在线电子支付等新型的商业运营模式。

# 信息系统架构三要素

构件、模式和规划

# 固态硬盘

Solid State Disk，简称固盘，用固态电子存储芯片阵列而制成的硬盘，由控制单元和存储单元（Flash芯片、DRAM芯片）组成；固态硬盘的功耗要低于传统硬盘。

固态硬盘与机械硬盘相比具有读写速度快、防震抗摔性、低功耗、无噪音、工作温度范围大、轻便等特点。

# 基准测试程序

把应用程序中应用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。

# 数字签名

数字签名首先需要生成消息摘要，然后发送方用自己的私钥对报文摘要进行加密，接收方用发送方公钥验证真伪。生成消息摘要的目的是（防止篡改），对摘要进行加密的目的是（防止抵赖）。

# 区块链

区块链是一种按照时间顺序将数据区块以顺序相连的方式组合成的一种链式数据结构，并以密码学方式保证的不可篡改和不可伪造的分布式账本。主要解决交易的信任和安全问题，最初是作为（比特币）的底层技术出现的。

# 总线

半双工总线可在两个方向上轮流传输信息，全双工总线可在两个方向上同时传输信息

# 磁盘阵列

Redundant Arrays of Independ Drives，RAID，有独立磁盘构成的具有冗余能力的阵列之意。

## **优点**

提高传输速率，RAID通过在多个磁盘上同时存储和读取数据来大幅提高存储系统的数据吞吐量（Throughput）；通过数据校验提供容错功能。

## RAID级别

### RAID0

是最早的RAID模式，及Data Stripping数据分条技术。

RAID0是组件磁盘阵列中最简单的一种形式，只需要2块以上的硬盘即可，成本低，可以提高整个磁盘的性能和吞吐量。RAID0没有提供冗余或错误修复能力，但实现成本是最低的。如使用了三块80GB的硬盘组件RAID0，那么磁盘容量就会是240GB。

### **RAID1**

称为磁盘镜像，原理是把一个磁盘的数据镜像到另一个磁盘上，也就是说数据在写入一块磁盘的同时，会在另一块闲置的磁盘上生成镜像文件，在不影响性能的情况下最大限度的保证系统的可靠性和可修复性。但是成本也会明显的增加，磁盘利用率为50%，以四块80GB的硬盘来讲，可利用的磁盘空间仅为160GB。

# 实时操作系统

RTOS的正确性依赖于运行结果的逻辑正确性和运行结果产生的时间正确性，即实时系统必须在规定的时间范围内正确地响应外部物理过程的变化。实时多任务操作系统是根据操作系统的工作特性而言的。实时是指物理进程的真实时间。实时操作系统是指具有实时性，能支持实时控制系统工作的操作系统。首要任务是调度一切可利用的资源来完成实时控制任务，其次才着眼于提高计算机系统的使用效率，重要特点是要满足对时间的限制和要求。一个实时操作系统可以在不破坏规定的时间限制的情况下完成所有任务的执行。任务执行的时间可以根据系统的软硬件的信息而进行确定性的预测。也就是说，如果硬件可以做这件工作，那么实时操作系统的软件将可以确定性的做这件工作。实时操作系统可根据实际应用环境的要求对内核进行裁剪和重新配置，根据不同的应用，其组成有所不同。

# 嵌入式操作系统

BSP（Board Support package）时板级支持包，是介于主板硬件和操作系统之间的一层，板级支持包BSP作为对硬件的抽象，实现了：硬件有关性，操作系统无关性。

# 微内核操作系统

采用微内核结构的操作系统提高了系统的灵活性和可扩展性，并增强了系统的可靠性和可移植性，可运行于分布式系统中。

# 电子邮件服务

MIME（Multipurpose Internet Mail Extensions）中文名为：多用途互联网邮件扩散类型。Internet电子邮件由一个邮件头部和一个可选的邮件主题组成，其中邮件头部含有邮件的发送方和接收方的有关信息。而MIME是针对邮件主体的一种扩散描述机制。他设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名，以及一些媒体文件打开方式。所以这是与邮件内容直接相关的一个协议。而S/MIME（Secure Multipurpose Internet Mail Extensions）是对MIME在安全方面的扩展。它可以把MIME实体（比如数字签名和加密信息等）封装成安全对象，这样就可以确保接收者不能否认已经收到过的邮件。还可以用于提供数据保密、完整性保护、认证和鉴定服务等功能。S/MIME只保护邮件的邮件主体，对头部信息则不进行加密，以便让邮件成功地在发送者和接收者的网关之间传递。

# API

Application Programming Interface，应用程序编程接口，是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。

所有操作系统（不仅仅是嵌入式操作系统）给应用程序提供的接口，称之为API。

# PCI

Peripheral Component Interconnect（外部部件互联标准），它是目前个人电脑中使用最为广泛的接口，几乎所有的主板产品上都带有这种插槽。

# 嵌入式处理器

嵌入式处理器是嵌入式系统的核心部件，一般可分为嵌入式微处理器（MPU）、微控制器（MCU）、数字信号处理器（DSP）和片上系统（SOC）。

MPU采用增强型通用微处理器。由于嵌入式系统通常应用于环境比较恶劣的环境中，因而MPU在工作温度、电磁兼容性以及可靠性方面的要求较通用的标准微处理器高。但是，MPU在功能方面与标准的微处理器基本上是一样的。

MCU又称单片微型计算机（Single Chip Microcomputer）或者单片机，是指随着大规模集成电路的出现及其发展，将计算机的CPU、RAM、ROM、定时计数器和多种I/O接口集成在一片芯片上，形成芯片级的计算机，为不同的应用场合做不同的控制组合。典型的代表是单片机、体积小从而使功耗和成本下降。

DSP是一种特殊的微处理器，是以数字信号来处理大量信息的器件。其实时运行速度可达每秒数以千万条复杂指令程序，远远超过通用微处理器，他的强大数据处理能力和高运行速度，是最值得称道的两大特色。

SOC称为系统级芯片，也有称片上系统，意指他是一个产品，是一个有专用目标的集成电路，其中包含完整系统并由嵌入软件的全部内容。

# 成本管理

成本管理的过程包括：成本估算、成本预算与成本控制。其中成本预算的含义是将总的成本估算分配到各项活动和工作包上，来建立一个成本的基线。而成本估算是对完成项目活动所需资金进行近似的估算。

# 项目配置管理

配置管理是PMBOK、ISO9000和CMMI中的重要组成元素，他在产品开发的生命周期中，提供了结构化的、有序化的、产品化的管理方法，是项目管理的基础工作。配置管理是通过技术和行政手段对产品及其开发过程和生命周期进行控制、规范的一系列措施和过程。信息系统开发过程中的变更以及相应的返工会对产品的质量有很大的影响。产品配置是指一个产品在其生命周期各个阶段所产生的各种形式（机器可读或人工可读）和各种版本的文档、计算机程序、部件及数据的集合。该集合中的每一个元素称为该产品配置中的一个配置项（Configuration Item，CI），配置项主要有两大类：

* 属于产品组成部分的工作成果，如需求文档、设计文档、源代码、测试用例等。
* 属于项目管理和机构支撑过程域产生的文档，如工作计划、项目质量报告、项目跟踪报告等。这些文档虽然不是产品的组成部分，但是值得保存。

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，并不关心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。

* 用户文档是用户了解系统的第一步，它可以让用户获得对系统的准确的初步印象。

用户文档至少应该包括下述5方面的内容：

1. 功能描述：说明系统能做什么
2. 安装文档：说明怎样安装这个系统以及怎样使系统适应特定的硬件配置
3. 使用手册：简要说明如何着手使用这个系统（通过丰富的例子说明怎样使用常用的系统功能，并说明用户操作错误时怎样恢复和重新启动）
4. 参考手册：详尽描述用户可以使用的所有系统设施以及他们的使用方法，并解释系统可能产生的各种出错信息的含义（对参考手册最主要的要求是完整，因此通常使用形式化的描述技术）
5. 操作员指南（如果需要有系统操作员的话）：说明操作员应如何处理使用中出现的各种情况。

* 系统文档

所谓系统文档指从问题定义、需求说明到验收测试计划这样一系列和系统实现有关的文档。描述系统设计、实现和测试的文档对于理解程序和维护程序来说是非常重要的。

配置项的状态通常包括：草稿、正式发布和正在修改。

# 项目时间管理

过程包括活动定义、活动排序、活动的资源估算、活动历时估算、制定进度计划以及进度控制。

# 需求的属性

包括：创建需求的时间、需求的版本号、创建需求的作者、负责认可该软件需求的人员、需求状态、需求的原因和根据、需求涉及的子系统、需求涉及的产品版本号、使用的验证方法或者接受的测试标准、产品的优先级或者重要程度、需求的稳定性。

# 需求的变更流程

1. 问题分析和变更描述
2. 变更分析和成本计算
3. 变更实现

自动化工具能够帮助变更控制过程更有效地运作。许多团队使用商业问题跟踪工具来收集、存储和管理需求变更。用这样的工具创建的最近提交的变更建议清单，可以用作CCB会议的议程。问题跟踪工具也可以随时按变更状态分类报告出变更请求的数目。

因为可用的工具、厂商和特性总在频繁的变化，所以这里无法给出有关工具的具体建议。但工具应该具有以下几个特性，以支持需求变更过程：

1. 可以定义变更请求中的数据项
2. 可以定义变更请求生命周期的状态转换模型
3. 可以强制实施状态转换模型，以便只有授权用户可以作出允许的状态变更
4. 可以记录每一个状态变更的日期和作出这一变更的人
5. 可以定义当提议者提交新的请求或请求状态被更新时，哪些人可以自动接受电子邮件通知
6. 可以生成标准的和定制的报告和图表

# 敏捷开发方法

1. XP（Extreme Programming，极限编程）在所有的敏捷型方法中，XP是最引人瞩目的。XP在一些对费用控制严格的公司中使用，已经被证明是非常有效的。是一种轻量级（敏捷）、高效、低风险、柔性、可预测的、科学的软件开发方式。它由价值观、原则、实践和行为四个部分组成，彼此相互依赖、关联、并通过行为贯穿于整个生命周期。其四大价值观包括沟通、简单、反馈和勇气。
2. 水晶系列方法，它与XP方法一样都是以人为中心的理念，但在实践上有所不同。考虑到人们一般很难严格遵循一个纪律约束很强的过程，因此与XP的高度纪律性不同，水晶方法探索用最少纪律约束而仍能成功的方法，从而在产出效率与易于运作上达到一种平衡。也就是说，虽然水晶系列不如XP那样的产出效率，但会有更多的人能够接受并遵循它。
3. 开放式源码，这里提到的开放式源码指的是开放源码界所用的一种运作方式。开放式源码项目有一个特别之处，就是程序开放人员在地域上分布很广，这使得他和其他敏捷方法不同，因为一般的敏捷方法都强调项目组成员在同一地点工作。开放源码的一个突出特点就是差错排障（debug）的高度并行性，任何人发现了错误都可将改正源码的“补丁”文件发给维护者。然后由维护者将这些“补丁”或是新增的代码并入源码库。
4. SCRUM。SCRUM已经出现很久了，像前面所涉及的方法一样，该方法强调这样一个事实，即明确定义了的可重复的方法过程只限于在明确定义了的可重复的环境中，为明确定义了的可重复的人员所用，去解决明确定义了的可重复问题。
5. 功用驱动开放方法（FDD-Feature Driven Development）FDD像其他方法一样，它致力于短时的迭代阶段和可见可用的功能。在FDD中，一个迭代周期一般是两周。在FDD中，编程开放人员分成两类：首席程序员和“类”程序员（class owner）。首席程序员是最富有经验的开发人员，他们是项目的协调者、设计者和指导者，而“类”程序员则主要做源码编写。

# 网络逻辑结构设计

利用需求分析和现有网络体系分析的结果来设计逻辑网络结构，最后得到一份逻辑网络设计文档，输出内容包括以下几点：

1. 逻辑网络设计图
2. IP地址方案
3. 安全方案
4. 招聘和培训网络员工的具体说明
5. 对软硬件、服务、员工和培训的费用初步估计

# 网络物理设计

物理网络设计是对逻辑网络设计的物理实现，通过对设备的具体物理分布、运行环境等确定，确保网络的物理连接符合逻辑连接的要求。输出如下内容：

1. 网络物理结构图和布线方案
2. 设备和部件的详细列表清单
3. 软硬件和安装费用的估算
4. 安装日程表，详细说明服务的时间及期限
5. 安装后的测试计划
6. 用户的培训计划

# 5G

峰值理论可达下行10Gbps。目前试用阶段一般下行1Gbps左右。

# FTP

FTP协议占用两个标准的端口号：20和21，其中20为数据口，21为控制口。

# 网络设备

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

# 网闸

网闸是使用带有多种控制功能的固态开关读写介质连接两个独立主机系统的信息安全设备。由于物理隔离网闸所连接的两个独立主机系统之间，不存在通信的物理连接、逻辑连接、信息传输命令、信息传输协议，不存在依据协议的信息包转发，只由数据文件的无协议“摆渡”，且对固态存储介质只有“读”和“写”两个命令。所以，物理隔离网闸从物理上隔离、阻断了具有潜在攻击可能的一切连接，使“黑客”无法入侵、无法攻击、无法破坏，实现了真正的安全。

使用安全隔离网闸的意义：

1. 当用户的网络需要保证高强度的安全，同时又与其他不信任网络进行信息安全交换的情况下，如果采用物理隔离卡，用户必须使用开关在内外网之间来回切换，不仅管理起来非常麻烦，使用起来也非常不方便，如果采用防火墙，由于防火墙自身的安全很难保证，所以防火墙也无法防止内部信息泄露和外部病毒、黑客程序的渗入，安全性无法保证。在这种情况下，安全隔离网闸能够同时满足这两个要求，弥补了物理隔离卡和防火墙的不足之处，是最好的选择。
2. 对网络隔离是通过网闸隔离硬件实现两个网络在链路层断开，但是为了交换数据，通过设计的隔离硬件在两个网络对应的层上进行切换，通过对硬件上的存储芯片的读写，完成数据的交换。
3. 安装了相应的应用模块之后，安全隔离网闸可以在保证安全的前提下，使用户可以浏览网页、收发电子邮件、在不同网络上的数据库之间交换数据，并可以在网络之间交换定制的文件。

# 软件系统工具

通常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。

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。

# 面向构件的编程

COP关注于如何建立面向构件的解决方案。一个基于一般OOP风格的COP定义如下：面向构件的编程需要下列基本的支持：

1. 多态性（可替代性）
2. 模块封装性（高层次信息的隐藏）
3. 后期的绑定和装载（部署独立性）
4. 安全性（类型和模块安全性）

# 分布式数据库

## 分布透明性

分布透明性包括：分片透明性、位置透明性和局部数据模型透明性。

1. 分片透明性是分布透明性的最高层次、所谓分片透明性是指用户或应用程序只对全局关系进行操作而不必考虑数据的分片。当分片模式改变时，只要改变全局模式到分片模式的映像（映像2），而不影响全局模式和应用程序。全局模式不变，应用程序不必改写，这就是分片透明性。
2. 位置透明是分布透明性的下一层次。所谓位置透明性是指：用户或应用程序应当了解分片情况，但不必了解片段的存储场地。当存储场地改变时，只要改变分片模式到分配模式映像（映像3），而不影响应用程序。同时，若片段的重复副本数目改变了，那么数据的冗余也会改变，但用户不必关心如何保持各副本的一致性，这也提供了重复副本的透明性。
3. 局部数据模型（逻辑透明）透明性是指用户或应用程序应当了解分片及各片段存储的场地，但不必了解局部场地上使用的时何种数据模型。

# 特定领域软件架构

特定领域软件架构（Domain Specific Software Architecture，DSSA）是在一个特定应用领域中为一组应用提供组织结构参考的软件体系结构，参与DSSA的人员可以划分为4中角色，包括：领域专家、领域设计人员、领域实现人员和领域分析师。

DSSA的基本活动包括：领域分析、领域设计、领域实现。

特定领域软件架构以一个特定问题领域为对象，形成由领域参考模型、参考需求、参考架构等组成的开发基础架构，其目标是支持一个特定领域中多个应用的生成。DSSA的基本活动包括领域分析、领域设计和领域实现。其中领域分析的主要目的是获得领域模型，领域模型描述领域中系统之间共同的需求，即领域需求；领域设计的主要目标是获得DSSA，DSSA描述领域模型中表示需求的解决方案；领域实现的主要目标是依据领域模型和DSSA开发和组织可重用信息，并对基础软件架构进行实现。

# 螺旋模型

螺旋模型将瀑布模型和演化模型结合起来，不仅体现了两个模型的优点，而且还强调了

其他模型均忽略了的风险分析。这种模型将整个软件开发流程分成多个阶段，每个阶段都由4部分组成，他们是：

1. 目标设定：为该项目进行需求分析，定义和确定这一个阶段的专门目标，指定对过程和产品的约束，并且制定详细的管理计划。
2. 风险分析：对可选方案机型风险识别和详细分析，指定解决办法，采取有效的措施避免这些风险
3. 开发和有效性验证：风险评估后，可以为系统选择开发模型，并且进行原型开发，即开发软件产品
4. 评审：对项目进行评审，以确定是否需要进入螺旋线的下一次回路，如果决定继续，就要指定下一阶段计划

螺旋模型的额软件开发过程实际上是上述4个部分的迭代过程，每迭代一次，螺旋线就增加一周，软件系统就生成一个新版本，这个新版本实际上是对目标系统的一个逼近。经过若干次的迭代后，系统应该尽快的收敛到用户允许或可以接受的目标范围内，否则也可能中途夭折。

# 统一过程

统一过程（UP）定义了初启阶段、精化阶段、构建阶段、移交阶段和产生阶段，每个阶段达到某个里程碑时结束。其中初启阶段的里程碑是生命周期目标，精化阶段的里程碑是生命周期架构，构建阶段的里程碑是初始运作功能，移交阶段的里程碑是产品发布。

# 软件开发模型三种类型

软件开发模型大体上可以分为三种类型。第一种是以软件需求完全确定为前提的瀑布模型；第二种是在软件开发初始阶段只能提供基本需求时采用的迭代式或渐进式模型，例如喷泉模型、螺旋模型、统一开放过程和敏捷方法等；第三种是以形式化为基础的变化模型。

# 增量模型

增量模型又称为渐增模型，也称为有计划的产品改进模型，他从一组给定的需求开始，通过构造一系列可执行中间版本来实施开发活动。第一个版本纳入一部分需求，下一个版本纳入更多的需求，依次类推，直到系统完成。每个中间版本都要执行必须的过程、活动和任务。增量模型是瀑布模型和原型进化模型的综合，他对软件过程的考虑是：在整体上按照瀑布模型的流程实施项目开发，以方便对项目的管理；但在软件的实际创建中，则将软件系统按功能分解为许多增量构件，并以构件为单位逐个的创建与交付，直到全部增量构件创建完毕，并都被集成到系统之中交付用户使用。比较瀑布模型、原型进化模型，增量模型具有非常显著的优越性。但增量模型对软件设计有更高的技术要求，特别是对软件体系结构，要求它具有很好的开放性与稳定性，能够顺利地实现构件的集成。

# 系统规划方法

用于管理信息系统规划的方法很多，主要是关键成功因素法（Critical Success Factors，CSF）、战略目标集转换法（Strategy Set Transformation，SST）和企业系统规划法（Business System Palnning，BSP）。其他还有企业信息分析与集成技术（BIAIT）、产出/方法分析（E/MA）、投资回收法（ROI）、征费法、零线预算法、阶石法等。用的最多的是前面三种。

1. 关键成功因素法（CSF），在现行系统中，总存在着多个变量影响系统目标的实现，其中若干个因素是关键的和主要的（即关键成功因素）。通过对关键成功因素的识别，找出实现目标所需的关键信息集合，从而确定系统开发的优先次序。关键成功因素来自于组织的目标，通过组织的目标分解和关键成功因素的识别、性能指标识别，一直到产生数据字典。识别关键成功因素，就是要识别联系于组织目标的主要数据类型及其关系。不同的组织的关键成功因素不同。当在一个时期内的关键成功因素解决后，新的识别关键成功因素又开始。关键成功因素法能抓住主要矛盾，使目标的识别突出重点。由于经理们比较熟悉这种方法，使用这种方法所确定的目标，因而经理们乐于努力去实现。该方法最有利于确定企业的管理目标。
2. 战略目标集转换法（SST），把整个战略目标看成是一个“信息集合”，由使命、目标、战略等组成，管理信息系统的规划过程即是把组织的战略目标转变成为管理信息系统的战略目标的过程。战略目标集转化法从另一个角度识别管理目标，它反映了各种人的要求，而且给出了按这种要求的分层，然后转化为信息系统目标的结构化的方法。他能保证目标比较全面，疏漏较少，但它在突出重点方面不如关键成功因素法。
3. 企业系统规划法（BSP），信息支持企业运行。通过自上而下地识别系统目标、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，该管理信息系统支持企业目标的实现，表达所有管理层次的要求，向企业提供一致性信息，对组织机构的变动具有适应性。企业系统规划法虽然也首先强调目标，但他没有明显的目标引导过程。它通过识别企业“过程”引出系统目标，企业目标到系统目标的转化是通过企业过程/数据类等矩阵的分析得到的。

# 项目范围管理

在初步项目范围说明书中已文档化的主要的可交付物、假设和约束条件的基础上准备详细的项目范围说明书，是项目成功的关键。范围定义的输入包括以下内容：

1. 项目章程
2. 项目范围管理计划
3. 组织过程资产
4. 批准的变更申请

# 软件过程模型

软件过程模型的基本概念：软件过程是制作软件产品的一组活动以及结果，这些活动主要由软件人员来完成，软件活动主要有：

1. 软件描述
2. 软件开发
3. 软件有效性验证
4. 软件进化

# 需求获取

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。常见的需求获取方式有用户访谈、问卷调查、抽样和联合需求计划等。

例：某大型移动通信运营商欲开发一个新的应用系统以替代原有系统。在需求分析阶段，为尽快从已有系统文档资料和用户处获取整体系统需求，采用（抽样）的方法捕获需求最为合适。

# UML

1、状态图描述了一个对象在其生命周期中可能的状态组合；

2、顺序图用来描述对象按照时间顺序的消息流来建模用例；

3、数据流图是一种描述数据通过系统的流程以及系统实施的工作或处理过程的过程模型；

4、流程图以图形化的方式展示应用程序从数据输入开始到获得输出为止的逻辑过程。

# 面向对象的分析模型

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。

# 面向对象分析模型

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。

# WEB服务器性能指标

最大并发连接数、响应延迟和吞吐量。

# 面向对象类之间的关系

1. 关联关系，关联提供了不同类对象之间的结构关系，他在一段时间内将多个类的实例连接在一起
2. 依赖关系，两个类A和B，如果B的变化可能会引起A的变化，则称A依赖于B
3. 泛化关系，泛化关系描述了一般事物与该事物中的特殊种类之间的关系，也就是父类与子类之间的关系。继承关系是泛化关系的反关系，也就是说，子类继承了父类，而父类则是子类的泛化
4. 共享聚集，共享聚集关系通常简称为聚合关系，它表示类之间的整体与部分的关系。其含义是“部分”可能同时属于多个“整体”，“部分”与“整体”的生命周期可以不相同。例如，汽车和车轮就是聚合关系，车子坏了，车轮还可以用；车轮坏了，还可以在换一个。
5. 组合聚集，通常简称为组合关系，他也是表示类之间的整体与部分的关系。与聚合关系的区别在于，组合关系中的“部分”只能属于一个“整体”，“部分”与“整体”的生命周期相同。例如，一个公司包含多个部门，他们之间的关系就是组合关系，公司一旦倒闭，也就无所谓部门了
6. 实现关系，实现关系将说明和实现联系起来。接口是对行为而非实现的说明，而类中则包含了实现的结构。一个或多个类可以实现一个接口，而每个类分别实现接口中的操作

# 用例之间的关系

1. 包含关系，当可以从两个或两个以上的用例中提取公共行为时，应该使用包含关系来表示他们。
2. 扩展关系，如果一个用例明显地混合了两种或两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样使描述可能更加清晰。
3. 泛化关系，当多个用例共同拥有一种类似的结构和行为的时候，可以将他们的共性抽象成为父用例，其他的用例作为泛化关系中的子用例。

# 结构化分析

结构化分析方法的基本思想使自顶向下，逐层分解，把一个大问题分解成若干个小问题，每个小问题再分解成若干个更小的问题。经过逐层分解，每个最底层的问题都是足够简单、容易解决的。结构化方法分析模型的核心是数据字典，围绕这个核心，有三个层次的模型，分别是数据模型、功能模型和行为模型（也称为状态模型）。在实际工作中，一般使用E-R图表示数据模型，用DFD表示功能模型，用状态转换图表示行为模型。这三个模型有着密切的关系，他们的建立不具有严格的时序性，而是一个迭代的过程。

通过顺序、分支和循环三种基本的控制结构可以构造出任何单入口单出口的程序。

# 集成测试

集成测试可以分为一次性组装和增量式组装，增量式组装测试效果更好。集成测试计划一般在概要设计阶段完成。

# 系统可修改性

可修改性（modifiability）是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可修改性。可修改性包含四个方面：

1. 可维护性（maintainability）。这主要体现在问题的修复上：在错误发生后“修复”软件系统。为可维护性做好准备的软件体系结构往往能做局部性的修改并能使对其他构件的负面影响最小化。
2. 可扩展性 （extendibility）。这一点关注的使使用新特性来扩展软件系统，以及使用改进版本来替换构件并删除不需要或不必要的特性和构件。为了实现可扩展性，软件系统需要松散耦合的构件。其目标使实现一种体系结构，它能使开发人员在不影响构件客户的情况下替换构件。支持把新构件集成到现有的体系结构中也是必要的。
3. 结构重组（reassemble）。这一点处理的是重新组织软件系统的构件及构件间的关系，例如通过将构件移动到一个不同的子系统而改变它的位置。为了支持结构重组，软件系统需要精心设计构件之间的关系，理性情况下，他们允许开发人员在不影响实现的主体部分的情况下灵活的配置构件。
4. 可移植性（portability）。可移植性使软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器。为了实现可移植，需要按照硬件无关的方式组织软件系统，其他软件系统和环境被提取出。可移植性是系统能够在不同计算环境下运行的能力。这些环境可能是硬件、软件，也可能是两者的结合。在关于某个特定计算环境的所有假设都集中在一个构件中时，系统是可移植的。如果移植到新的系统需要做些更改，则可移植性就是一种特殊的可修改性。

# 软件确认测试

包括：内部确认测试、Alpha测试、Beta测试和验收测试。

# 软件测试的分类

软件测试一般分为两个大类：动态测试和静态测试。

动态测试是通过运行程序发现错误，包括黑盒测试（等价类划分、边界值分析法、错误推测法）与白盒测试（各种类型的覆盖测试）。

静态测试是人工测试方法，包括桌前检查（桌面检查）、代码走查、代码审查。

# 集中式负载均衡

缺点：

1. 系统的可扩展性不强，均衡器需要记录所有计算机的负载信息
2. 安全性较差，如果均衡器所在的计算机瘫痪，则会导致整个集群系统的瘫痪
3. 实现不够灵活，负载均衡器很难根据不同的脚手架的特性配置不同的均衡策略

# 软件维护

1. 改正性维护，为了识别和纠正软件错误、改正软件性能上的缺陷、排除实施中的误使用，应当进行的诊断和改正错误的过程称为改正性维护。
2. 适应性维护，在使用过程中，外部环境（新的硬、软件配置）、数据环境（数据库、数据格式、数据输入/输出方法、数据存储介质）可能发生变化。为使软件适应这种变化而修改软件的过程称为适应性维护
3. 完善性维护，在软件的使用过程中，用户往往会对软件提出新的功能与性能要求。为了满足满足这些要求，需要修改或再开发软件，以扩充软件功能、增强软件性能、改进加工效率、提高软件的可维护性。这中情况下进行的维护活动称为完善性活动
4. 预防性维护，指预先提高软件的可维护性、可靠性等，为以后进一步改进软件打下良好的基础。采用先进的软件工程方法对需要维护的软件或软件中某一部分（重新）进行设计、编码和测试。

# 单元测试

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、软件构件或面向对象软件中的类（统称为模块），其目的是检查每个模块能否正确地实现设计说明中的功能、性能、接口和其他设计约束等条件，发现模块内可能存在的各种差错。单元测试的技术依据是软件详细设计说明书。测试一个模块时，可能需要为该模块编写一个驱动模块和若干个桩模块。驱动模块用来调用被测模块，他接收测试者提供的测试数据，并把这些数据传送给被测模块，然后从被测模块接收测试结果，并以某种可见的方式将测试结果返回给测试人员；桩模块用来模拟被测模块所调用的子模块，他接受被测模块的调用，检验调用参数，并以尽可能简单的操作模拟被测模块所调用的子程序模块功能，把结果送回被测模块。顶层模块测试时不需要驱动模块，底层模块测试时不需要桩模块。单元测试的策略主要包括自顶向下的单元测、自底向上的单元测试、孤立测试和综合测试策略。

1. 自顶向下的单元测试先测试上层模块，再测试下层模块。测试下层模块时由于它的上层模块已经测试通过，所以不必另外编写驱动模块
2. 自底向上的单元测试先测试下层模块，再测试上层模块。测试上层模块由于他的下层模块已经测试通过，所以不必另外编写桩模块。
3. 孤立测试不需要考虑每个模块与其他模块之间的关系，逐一完成所有模块的测试。由于各模块之间不存在依赖性，单元测试可以并行进行，但因为需要为每个模块单独设计驱动模块和桩模块，增加了额外的测试成本
4. 综合测试，上述三种单元测试策略各有利弊，实际测试时可以根据软件特点和进度安排情况，将几种测试方法混合使用

# 架构复审

在对一个软件系统的架构进行设计与确认之后，需要进行架构复审。架构复审的目的是为了标识潜在的风险，及早发现架构设计中的缺陷和错误。在架构复审过程中，主要由用户代表与领域专家决定架构是否满足需求、质量需求是否在设计中得到体现。

# 敏捷开发

敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构件初期被切分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

# ODP架构过程

系统构想、需求分析、原型分析、架构规划、架构原型、项目规划、并行开发、系统转换、操作维护、系统移植。

# OMG接口定义

OMG接口定义语言IDL是Interface description language的缩写，指接口描述语言，是CORBA规范的一部分，是跨平台开发的基础。IDL通常用于远程调用软件。在这种情况下，一般是由远程客户终端调用不同操作系统上的对象组件，并且这些对象组件可能是由不同计算机语言编写的。IDL建立起了两个不同操作系统间通信的桥梁。

从本质上讲，OMG IDL接口定义语言不是作为程序设计语言体现在CORBA体系结构中的，而是用来描述产生对象调用请求的客户对象和服务对象之间的接口的语言。OMG IDL文件描述数据类型和方法框架，而服务对象则为一个指定的对象实现提供上述数据和方法。

值类型是一个IDL文件核心的内容。

模块定义将映射为java语言中的包（package）或c++语言中的命名空间（namespace）。

# 软件逆向工程

所谓软件逆向工程就是分析已有的程序，寻求比源代码更高级的抽象表现形式。一般认为，凡是在软件生命周期内将软件某种形式的描述转换成更为抽象形式的活动都可以称为逆向工程。

# 引入DTD的原因

1. 提供一种验证的手段
2. 实现了文件格式的统一化，提高了文件的重用性
3. 使用DTD进行验证，增加了操作时间

# 系统工程

系统工程在上个世纪中后期发展起来的一门新兴学科。它最早约产生于20世纪40年代的美国，时至今日，系统工程已经成为现代社会高速发展不可或缺的一部分。系统工程的诞生让自然科学和社会科学中有关的思想、理论和方法根据总体协调的需要联系起来，综合应用，并利用现代电子计算机，对系统的结构、要素、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等目的。

霍尔三维结构是由逻辑维、时间维和知识维组成的立体空间结构

1. 逻辑维，运用系统工程方法解决某一大型工程项目时，一般可分为七个步骤：
2. 明确问题
3. 建立价值体系或评价体系
4. 系统分析
5. 系统综合
6. 系统方案的优化选择
7. 决策“决策就是管理”，“决策就是决定”，人类的决策管理活动面临着被决策系统的日益庞大和日益复杂
8. 制定计划有了决策就要付诸实施，实施就要依靠严格的有效的计划
9. 时间维（工作进程），对于一个具体的工作项目，从制定规划起一直到更新为止，全部过程分为七个阶段：
10. 规划阶段，即调研、程序设计阶段，目的在于谋求活动的规划与战略
11. 拟定方案，提出具体的计划方案
12. 研制计划，作出研制方案及生产计划
13. 生产阶段，生产出系统的零部件及整个系统，并提出安装计划
14. 安装阶段，将系统安装完毕，并完成系统的运行计划
15. 运行阶段，系统按照预期的用途开展服务
16. 更新阶段，即为了提高系统的功能，取消旧系统而代之以新系统，或改进原有系统，使之更加有效的工作
17. 知识维（专业科学知识），系统工程除了要求为完成上述各步骤、各阶段所需的某些共性知识外，还需要其他学科的知识和各种专业技术，霍尔把 这些知识分为工程、医药、建筑、商业、法律、管理、社会科学和艺术等。各类系统工程，如军事系统工程、经济系统工程、信息系统工程等，都需要使用其他相应的专业基础知识。

# 需求管理活动

变更控制、版本控制、需求跟踪、需求状态跟踪。

# 快速应用开发

快速应用开发利用了基本构件开发方法的思想，大量采用现成的构件进行系统的开发，所以速度很快。但这种开发，要求系统模块化程度高，因为只有这样，才能更好利用现有的构件。

# 软件重用

软件重用分垂直式重用与水平式重用，垂直式重用是指局限于某一垂直领域的重用，如只在电力系统中用到的构件；而水平式重用是指通用领域的重用，如标准函数库，任何软件都能用，所以是水平式重用。

# 版本控制软件

版本控制软件提供完备的版本管理功能，用于存储、追踪目录（文件夹）和文件的修改历史，是软件开发者的必备工具，是软件公司的基础设施。版本控制软件的最高目标，是支持软件公司的配置管理活动，追踪多个版本的开发和维护工作，及时发布软件。SCCS是元老级的版本控制软件，也叫配置管理软件。

# 甘特图

甘特图是一种能清晰描述每个任务的开始和截至时间，能有效获得任务并行进行的信息的项目管理工具。

# 知识产权

计算机软件既是作品，又是一种使用工具，还是一种工业产品（商品），具备作品性、工具性、商业性特征。因此对于计算机软件保护来说，仅依靠某项法律或法规不能解决软件的所有知识产权问题，需要利用多层次的法律保护体系对计算机软件实施保护。我国已形成了比较完备的计算机软件知识产权保护大的法律体系，即已形成以著作权法、计算机软件保护条例、计算机软件著作权登记办法保护为主，以专利法、反不正当竞争法、合同法、商标法、刑法等法律法规为辅的多层次保护体系，可对计算机软件实施交叉和重叠保护。在这样的保护体系下，计算机软件能够得到全面的、适度的保护。

例如，计算机软件符合专利法所保护的法定主题，就可以申请专利，利用专利法来保护其中符合发明创造条件的创造性成果。对于那些为极少数专门用户开发的专用软件，可以利用反不正当竞争法中的商业秘密权和合同法来保护其中的技术秘密。

我国没有专门针对知识产权制定统一的法律（知识产权法），而是在民法通则规定的原则下，根据知识产权的不同类型制定了不同的单项法律即法规，如著作权法、商标法、专利法、计算机软件保护条例等，这些法律、法规共同构成了我国保护知识产权的法律体系。

《计算机软件保护条例》第六条 规定：“本条例对软件著作权的保护不延及开发软件所用的思想、处理过程、操作方法或者数学概念等。”

软件商标权的权利人是指软件注册商标所有人。

自然人的软件著作权，保护期为自然人终生及其死亡后50年，截止于自然人死亡后第50年12月31日；软件是合作开发的，截止于最后死亡的自然人死亡后第50年的12月31日。法人或者其他组织的软件著作权，保护期限为50年，截止于软件首次发表后第50年的12月31日，但软件自开发完成之日起50年内未发表的，不再受到法律保护。

# 多态

在收到消息时，对象要予以响应。不同的对象收到统一消息可以产生完全不同的结果，这一现象叫做多态（polymorphism）。在使用多态的时候，用户可以发送一个通用的消息，而实现的细节则由接收对象自行决定。这样同一消息就可以调用不同的方法。

# 绑定

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，叫做静态绑定。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结合直到调用发生时才进行。

动态绑定是和类的继承以及多态相联系的。在继承管系统，子类是父类的一个特例，所以，父类对象可以出现的地方，子类对象也可以出现。因此在运行过程中，当一个对象发送消息请求服务时，要根据接收对象的具体情况将请求的操作与实现的方法进行连接，即动态绑定。

# 流程设计工具

## 程序流程图

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。任何流程图中只能包括5种基本控制结构：顺序型、选择型、WHILE循环型（当型循环）、UNTIL循环型（直到循环型）和 多分支选择型。

## IPO图

IPO图是由IBM公司发起并逐步完善的一种流程描述工具，其主体是处理过程说明，可以采用流程图、判定树、判定表、盒图、问题分析图或过程设计语言来进行描述。IPO图中的输入、输出与功能模块、文件及系统外部项都需要通过数据字典来描述，同时需要为其中的某些元素添加注释。

## N-S图

N-S图与PFD类似，也包括5中控制结构，分别是顺序型、选择型、WHILE循环型、UNTIL循环型和多分支选择型，任何一个N-S图都是这5种基本控制结构相互组合与嵌套的结果。在N-S图中，过程的作用域明确；他没有箭头，不能随意转移控制；而且容易表示嵌套关系和层次关系；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。

## 问题分析图

问题分析图（Problem Analysis Diagram，PAD）是继PFD和N-S图之后，又一种描述详细设计的工具。PAD也包含5种基本控制结构，并允许递归使用。

## 过程设计语言

过程设计语言（Process Design Language，PDL）也称为结构化语言或伪代码（pseudo code），他是一种混合语言，采用自然语言的词汇和结构化程序设计语言的语法，用于描述处理过程怎么做，类似于编程语言。过程设计语言用于描述模块中算法和加工逻辑的具体细节，以便在开发人员之间比较精确地进行交流。

## 总结

对于具有多个相互联系的条件和可能产生多种结果的问题，用结构化语言描述则显得不够直观和紧凑，这时可以用以清楚、简明为特征的判定表（Decision Table）来描述。判定表采用表格形式来表达逻辑判断问题，表格分成4个部分，左上部分为条件说明，左下部分为行动说明，右上部分为各种条件的组合说明，右下部分为各条件组合下相应的行动。

判定树（Decision Tree）也是用来表示逻辑判断问题的一种常用的图形工具，他用树来表达不同条件下的不同处理流程，比语言、表格的方式更为直观。判定树的左侧（称为树根）为加工名，中间是各种条件，所有的行动都列于最右侧。

# 设计原则

1、单一职责原则：设计目的单一的类

2、开放-封闭原则：对扩展开放，对修改封闭

3、李（里）氏（Liskov）替换原则：子类可以替代父类

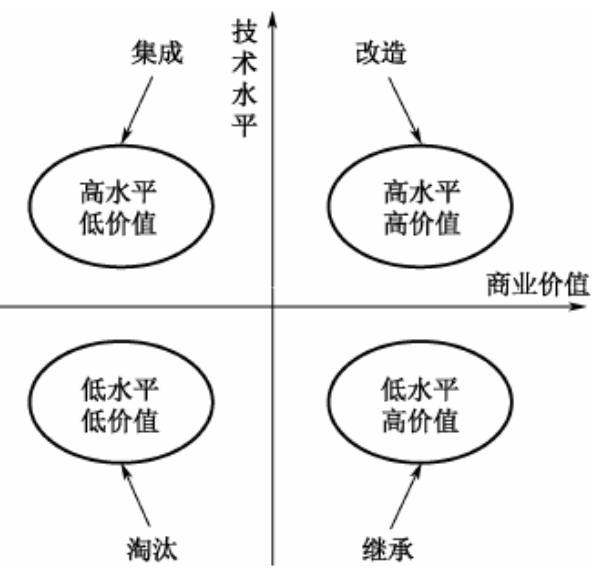
4、依赖倒置原则：要依赖于抽象，而不是具体实现；针对接口编程，不要针对实现编程

5、接口隔离原则：使用多个专门的接口比使用单一的总接口要好

6、组合重用原则：要尽量使用组合，而不是继承关系达到重用目的

7、迪米特（Demeter）原则（最少知识法则）：一个对象应当对其他对象有尽可能少的了解

# 遗留系统演化策略



1. 淘汰策略

第3象限为低水平、低价值区，即遗留系统的技术含量较低，且具有较低商业价值。对这种遗留系统的演化策略为淘汰，即全面开发新的系统以代替遗留系统。

1. 继承策略

第4象限为低水平、高价值区，即遗留系统的技术含量低，可满足企业运作的功能或性能要求，但具有较高的商业价值，目前企业业务对该系统仍有很多的依赖性。对这种遗留系统的演化策略为继承。

1. 改造策略

第1象限为高水平、高价值区，即遗留系统的技术含量较高，本身还有较低的生命力，且具有较高的商业价值，基本上能够满足企业业务运作和决策支持的要求。这种系统可能建成的时间还很短，对这种遗留系统的演化策略为改造。

1. 继承策略

第2象限为高水平、低价值区，即遗留系统的技术含量较高，但其商业价值较低，可能只能完成某个部门（或子公司）的业务管理。这种系统在各自的局部领域能工作 良好，但从企业全局来看，多个这样的系统，他们各自基于不同的平台，不同的数据模型，无法互联互通，数据还不一致，这就是很严重的问题了。对这种系统的演化策略为继承。

# 存储器

SRAM静态的随机存储器：特点是工作速度快，只要电源不撤除，写入SRAM的信息就不会消失，不需要刷新电路，同时在读出时不破坏原来存放的信息，一经写入可多次读出，但集成度较低，功耗较大。SRAM一般用来作为计算机中的高速缓冲存储器（Cache）。

DRAM只能将数据保持很短的时间。为了保持数据，DRAM使用电容存储，所以必须隔一段时间刷新（refresh）一次，如果存储单元没有被刷新，存储的信息就会丢失。关机就会丢失数据。

# DSP芯片

编程DSP芯片是一种具有特殊结构的微处理器，为了达到快速进行数字信号处理的目的，DSP芯片一般采用特殊的软硬件结构：

1. 哈佛结构，DSP采用了哈佛结构，将存储器空间划分成两个，分别存储程序和数据。他们有两组总线连接到处理器核，允许同时对他们进行访问，每个存储器独立编址，独立访问。这种安排将处理器的数据吞吐率加倍，更重要的是同时为处理器核提供数据与指令。在这种布局下，DSP得以实现单周期的MAC指令。在哈佛结构种，由于程序和数据存储器在两个分开的空间中，因此取指和执行能完全重叠运行。
2. 流水线，与哈佛结构相关，DSP芯片广泛采用2-6级流水线以减少指令执行时间，从而增强了处理器的处理能力。这可使指令执行能完全重叠，每个指令周期内，不同的指令都处于激活状态。
3. 独立的硬件乘法器，在实现多媒体功能及数字信号处理的系统中，算法的实现和数字滤波都是计算密集型的应用。在这些场合，乘法运算是数字处理的重要组成部分，是各种算法实现的基本元素之一。乘法的执行速度越快，DSP处理器的性能越高，相比与一般的处理器需要30-40个指令周期，DSP芯片的特征就是有一个专用的硬件乘法器，乘法可以在一个周期内完成。
4. 特殊的DSP指令，DSP的另一特征是采用特殊的指令，专为数字信号处理中的一些常用算法优化。这些特殊指令为一些典型的数字处理提供加速，可以大幅提高处理器的执行效率。使一些高速系统的实时数据处理成为可能。
5. 独立的DMA总线和控制器，有一组或多组独立的DMA总线，与CPU的程序、数据总线并行工作。在不影响CPU工作的条件下，DMA的速度已经达到800MB/S。这在需要大数据量进行交换的场合可以减小CPU的开销，提高数据的吞吐率，提高系统的并行执行能力。
6. 多处理器接口，使多个处理器可以很方便的并行或串行工作以提高处理速度。
7. JTAG（Join Test Action Group）标准测试接口（IEEE1149标准接口），便于对DSP芯片上的在线仿真和多DSP条件下的测试
8. 快速的指令周期，哈佛结构，流水线操作，专用的硬件乘法器，特殊的DSP指令再加上集成电路的优化设计，可使DSP芯片的指令周期在10ns以下。快速的指令周期可以使DSP芯片能够实时实现许多DSP应用。

# 原型系统和XP小规模发布的系统主要差别

原型系统和XP小型发布的系统的主要差别是功能。采用原型系统主要是让用户确认需求，或者用来测试关键的技术，但是它展示的功能并不是实际系统的功能，不能用来评价实际的系统；XP小型发布的系统不包括足够的功能，但是每个功能和可发布的产品的定义是一样的。在完整性上，它配备了一系列实用的功能集；在质量上，它可以健壮地运行。

# VRAPS组织管理原则

Vision：构想，Rhythm：节奏，Anticipation：预见，Partnering：协作，Simplification：简化。

VARPS组织管理原则：包括构想、节奏、预见、协作、简化5个相关原则。

1. 构想原则：说明了如何向架构的受益人描述一副一致的，有约束力和灵活的未来图景
2. 节奏原则：刻画了一种在整个组织范围内的协调程度
3. 预见原则：要在预测未来与检查并适应现状之间作出平衡
4. 协作原则：解决了如何识别对架构成功关键的团体，以及如何确保这些合作伙伴的有效支持
5. 简化原则：要求理解组织的结构，了解结构最小的基本特征并最小化架构

准则：是否和如何执行原则的细节

模式：描述了开发或者使用软件架构时可能遇到的常见问题的解决方法。模式注重于解决特定情况下的问题。传达了在给定背景和多方竞争因素下针对常见问题的解决方案。

反模式：描述了组织在实践中可能遇到的陷阱，描述了不该做的事情，或者用在错误背景下的解决方案。

构想必须是明晰的、有约束力的、一致的和灵活的。

用例建模时把架构的预期使用与能够被满足的切实的用户目标连接起来的一种方法。

# 管道和过滤器软件体系结构

在管道和过滤器软件体系机构中，每个模块都有一组输入和一组输出。每个模块从他的输入端接收输入数据流，在其内部经过处理后，按照标准的顺序，将结果数据流送到输出端，以达到传递一组完整的计算结果实例的目的。它最典型的应用是在编译系统。一个普通的编译系统包括词法分析器，语法分析器，语义分析与中间代码生成器，优化器，目标代码生成器等一系列对源程序进行处理的过程。

# 企业门户

按照实际应用领域，企业门户可以划分为以下四类：

1. 企业网站—功能简单，注重信息的单向传送，忽视用户与企业间、用户相互之间的信息互动。面向特定的使用人群，为企业服务，可以被看做是EP发展的雏形。
2. 企业信息门户—在internet环境下，把各种应用系统、数据资源和互联网资源统一集成到EP之下，根据每个用户使用特点和角色的不同，形成个性化的应用界面，并通过对事件和消息的处理传输把用户有机地联系在一起
3. 企业知识门户—是企业员工日常工作所涉及相关主题内容的“总店”。企业员工可以通过EKP方便地了解当天的最新消息、工作 内容、完成这些工作所需的知识等。可以实时地与工作团队中的其他成员取得联系，寻找能够提供帮助的专家或者快速地连接到相关部门
4. 企业应用门户—是对企业业务流程的集成。他以业务流程和企业应用为核心，把业务流程中功能不同的应用模块通过门户技术集成在一起。可以把EAP看成是企业信息系统的集成界面。企业员工和合作伙伴可以通过EAP访问相应的应用系统，实现移动办公、进行网上交易等。

# 软件开发环境

软件开发环境（Software Development Environment，SDE）是指支持软件的工程化开发和维护而使用的一组软件，由软件工具集和环境集成机制构成。

软件开发环境应支持多种集成机制，例如，平台集成、数据集成、界面集成、控制集成和过程集成等。软件开发环境应支持小组工作方式，并为其提供配置管理，环境的服务可用于支持各种软件开发活动，包括分析、设计、编程、调试和文档等。较完善的软件开发环境通常具有多种功能，例如，软件开发的一致性与完整性维护，配置管理及版本控制，数据的多种表示形式及其在不同形式之间的自动转换，信息的自动检索与更新，项目控制和管理，以及对开发方法学的支持。软件开发环境具有集成性、开放性、可裁剪性、数据格式一致性、风格统一的用户界面等特性，因而能大幅提高软件生产率。

集成机制根据功能的不同，可划分为环境信息库、过程控制与消息服务器、环境用户界面三个部分：

1. 环境信息库。环境信息库是软件开放环境的核心，用以存储与系统开放有关的信息，并支持信息的交流与共享。环境信息库中主要存储两类信息，一类是开放过程中产生的有关被开放系统有关的信息，例如，分析文档、设计文档和测试报告等；另一类是环境提供的支持信息，例如，文档模板、系统配置、过程模型和可复用构件等
2. 过程控制与消息服务器。过程控制与消息服务器是实现过程集成和控制集成的基础。过程集成是按照具体软件开放过程的要求进行工具的选择与组合，控制集成使各工具之间进行并行通信和协同工作。
3. 环境用户界面。环境用户界面包括环境总界面和由他实行统一控制的各环境部件及工具的界面。统一的、具有一致性的用户界面是软件开发环境的重要特征，是充分发挥环境的优越性、高效地 使用工具并减轻用户的学习负担的保证。

# 事务处理（交易）中间件

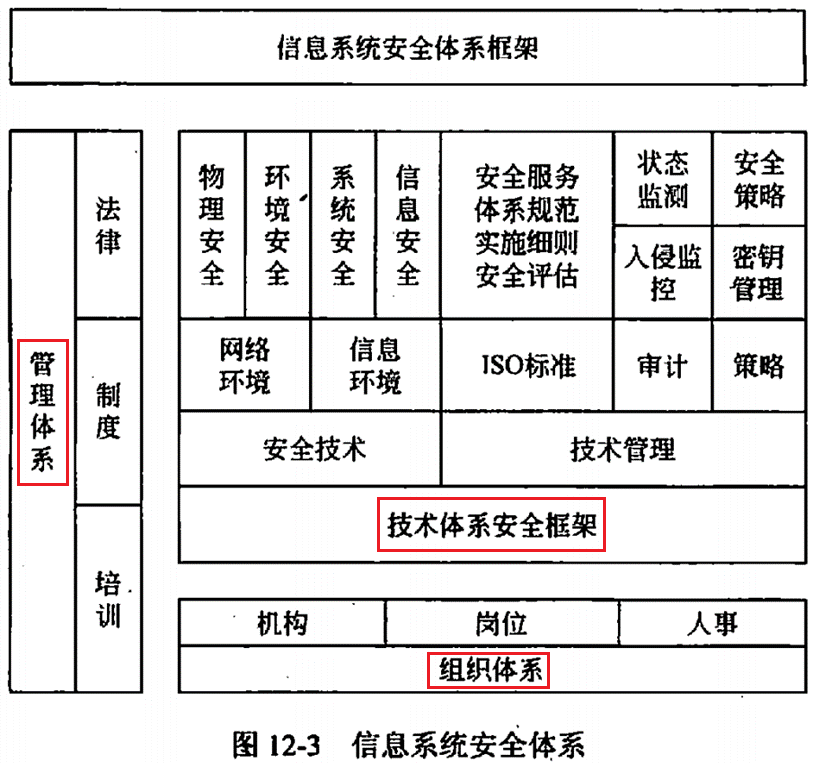
事务处理（交易）中间件，正如城市交通中要运行各种运载汽车，以此来完成日常的运载，同时随时监视汽车运行，在出现故障时及时排堵保畅。在分布式事务处理系统中，经常要处理大量事务，特别是OLTP中，每项事务常常要多台服务器上的程序按顺序协调完成，一旦中间发生某种故障，不但要完成恢复工作，而且要自动切换系统，达到系统永不停机，实现高可靠性运行。要使大量事务在多台应用服务器上能定时并发运行，并进行负载平衡的调度，实现与昂贵的可靠性和大型计算机系统同等的功能，为了实现这个目标，要求中间件系统具有监视和调度整个系统的功能。BEA的Texedo由此而著名，它成为增长率最高的厂商。

# 网络系统生命周期

需求规范、通信规范、逻辑网络设计、物理网络设计、实施

# 网络与信息安全风险

# 信息系统安全体系



# ISO/IEC软件质量模型

由三个层次组成：

第一层是质量特性；

第二层是质量子特性；

第三层是度量指标。

功能性是与一组功能及其指定的性质的存在有关的一组属性，其子特性包括适应性、准确定、互用性、依从性和安全性。

软件质量是软件特性的总和，是软件产品满足规定或潜在用户需求的能力。2001年，软件产品质量国际标准ISO/IEC9126定义的软件质量包括“内部质量”、“外部质量”和“使用质量”3部分。也就是说，“软件满足规定或潜在用户需求的能力”要从软件在内部、外部和使用中的表现来衡量。

|  |  |
| --- | --- |
| 质量特性 | 质量子特性 |
| 功能性 | 适合性 |
| 准确性 |
| 互用性 |
| 依从性 |
| 安全性 |
| 可靠性 | 成熟性 |
| 容错性 |
| 易恢复性 |
| 易使用性 | 易理解性 |
| 易学性 |
| 易操作性 |
| 效率 | 时间特性 |
| 资源特性 |
| 可维护性 | 易分析性 |
| 易改变性 |
| 稳定性 |
| 易测试性 |
| 可移植性 | 适应性 |
| 易安装性 |
| 一致性 |
| 易替换性 |

# 架构描述语言

架构描述语言（Architecture Description Language，ADL）是一种为明确说明软件系统的概念架构和这些概念架构建模提供功能的语言。ADL主要包括以下组成部分：组件、组件接口、连接件和架构配置。ADL对连接件的重视成为区分ADL和其他建模语言的重要特征之一。

# 架构权衡分析法

正确识别风险点、非风险点、敏感点和权衡点是进行软件架构评价的关键步骤。其中敏感点是实现一个特定质量属性的关键特征，该特征为一个或多个软件构件所共有。系统权衡点会影响一个或多个属性，并对于多个属性来说都是敏感点。基于该定义，可以看出“改变加密的级别可能会对安全性和性能都产生显著的影响”正是一个对系统权衡点的描述。

# 构件的特性

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

# 负载均衡

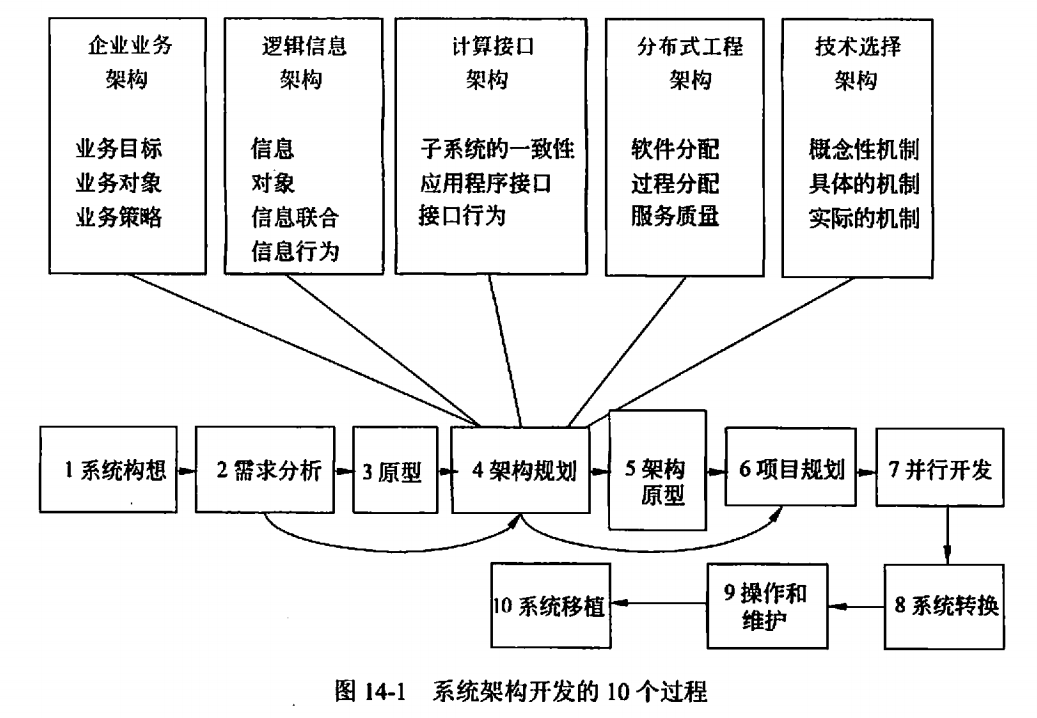
负载均衡（LoadBalance）建立在现有的网络结构之上，它提供了一种廉价、有效、透明的方法，来扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力、提高网络的灵活性和可用性。

负载均衡有两方面的含义：首先，大量的并发访问或数据流量分担到多台节点设备上分别处理，减少用户等待响应的时间；其次，单个重负载的运算分别分担到多台节点设备上做并行处理，每个节点设备处理结束后，将结果汇总，返回给用户，系统处理能力得到大幅度提高。

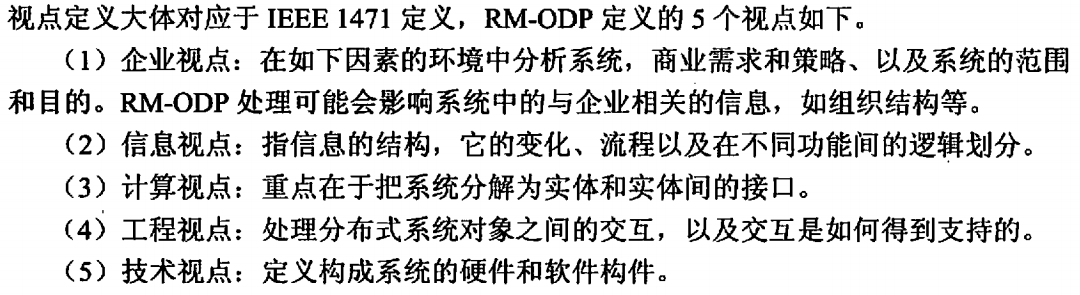
# 入侵检测系统

负责监视网络上的通信数据流和网络服务器系统中的审核信息，捕获可疑的网络和服务器系统活动，发现其中存在的安全问题，当网络和主机被非法使用或破坏时，进行实时响应和报警，产生通告信息和日志。系统不仅仅检测来自外部的额入侵行为，还可以检测内部用户的未授权活动。

# 系统架构开发10个过程



# RM-ODP定义的五个视点



# MTBF

称为平均故障间隔时间，用于衡量产品可靠性的一个指标。

可靠性：在规定的时间内，产品保持正常功能的一种能力。

# 双机热备份技术

双机热备份方案中，根据两台服务器的工作方式可以有三种不同的工作模式：双机热备模式、双机互备模式和双机双工模式。

双机热备模式即目前通常所说的active/standby方式，active服务器处于工作状态；而standby服务器处于监控准备状态，服务器数据包括数据库数据同时往两台或多台服务器写入（通常各服务器采用RAID磁盘阵列卡），保证数据的即时同步。当active服务器出现故障时，通过软件诊断或手工方式将standby机器激活，保证应用在短时间内完全恢复正常使用。但由于另外一台服务器长期处于后备的状态，所以浪费了一部分计算资源。

双机互备模式，是两个相对独立的应用在两台机器同时运行，但彼此均为备机，当某台服务器出现故障时，另一台服务器可以在短时间内将故障服务器的应用接管过来，从而保证了应用的持续性，但对服务器的性能要求比较高。

双机双工模式是集群的一种形式，两台服务器均处于活动状态，同时运行相同的应用，以保证整体系统的性能，也实现了负载均衡和互为备份，通常使用磁盘柜存储技术。Web服务器或FTP服务器等用此种方式比较多。

# 集成测试

集成测试可以分为一次性组装和增量式组装，增量式组装测试效果更好。集成测试计划一般在概要设计阶段完成。

# 可修改性

可修改性（modifiablity）是指能够快速地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可修改性。可修改性包含四个方面：

1. 可维护性（maintainability）。这主要体现在问题的修复上，在错误发生后“修复“软件系统。为可维护性做好准备的软件体系结构往往能做局部性的修改并能使对其他构件的负面影响最小化
2. 可扩展性（extendibility）。这一点关注的是使用新特性来扩展软件系统，以及使用改进版本来替换构件并删除不需要或不必要的特性和构件。为了实现可扩展性，软件系统需要松散耦合的构件。其目标是实现一种体系结构，它能使开发人员在不影响构件客户的情况下替换构件。支持把新构件集成到现有的体系结构中也是必要的。
3. 结构重组（reassemble）。这一点处理的是重新组织软件系统的构件及构件间的关系，例如通过将构件移动到一个不同的子系统而改变它的位置。为了支持结构重组，软件系统需要精心设计构件及构件间的关系。理想情况下，他们允许开发人员在不影响实现的主体部分的情况下灵活的配置构件。
4. 可移植性（portability）。可移植性使软件系统适用于多种硬件平台、用户界面、操作系统、编程语言或编译器。为了实现可移植，需要按照硬件无关的方式组织软件系统，其他软件系统和环境被提取出。可移植性是系统能够在不同计算环境下运行的能力。这些环境可能是硬件、软件，也可能是两者的结合。在关于某个特定计算环境的所有假设都集中在一个构件中时，系统时可移植的。如果移植到新的系统需要做些更改，则可移植性就是一种特殊的可修改性。

# 数据仓库

数据仓库四大特点：

1. 面向主题：数据按主题组织
2. 集成的：消除了源数据中的不一致性，提供整个企业的一致性全局信息
3. 相对稳定的（非易失的）：主要进行查询操作，只有少量的修改和删除操作（或是不删除的）
4. 反映历史变化（随着时间变化）：记录了企业从过去某一时刻到当前各个阶段的信息，可对发展历程和未来趋势做定量分析和预测

# 串行总线

特点总结如下：

1. 串行总线有半双工、全双工之分，全双工是一条线发一条线收。
2. 串行总线适宜长距离传输数据
3. 串行总线按位（bit）发送和接收。尽管比按字节（byte）的并行通信慢，但是串口可以在使用一根线发送数据的同时用另一根线接收数据。他很简单并且能够实现远距离通信。比如IEEE488定义并行通信状态时，规定设备线总长不得超过20米，并且任意两个设备间的长度不得超过两米；而对于串口而言，长度可达1200米
4. 串口通信最重要的参数是波特率、数据位、停止位和奇偶校验。对于两个进行通信的端口，这些参数必须匹配。
5. 串型总线的数据发送和接收可以使用多种方式，中断方式与DMA都较为常见

# 嵌入式系统软件低功耗设计

软件设计层面的功耗控制主要可以从以下方面展开：

1. 软硬件协同设计，即软件的设计要与硬件匹配，考虑硬件因素
2. 编译优化，采用低功耗优化的编译技术
3. 减少系统的持续运行时间，可以从算法角度进行优化
4. 用“中断”代替“查询”
5. 进行电源的有效管理

# CPU工作频率

CPU的工作频率（主频）包括两个部分：外频与倍频，两者的乘积就是主频。

所谓外频，就是外部频率，指的是系统总线频率。

倍频的全称是倍频系数，倍频系数是指CPU主频与外频之间的相对比例关系。最初CPU主频和系统总线速度是一样的，但CPU的速度越来越快，倍频技术也就相应产生。它的作用是使系统总线工作在相对较低的频率上，而CPU速度可以通过倍频来提升。

# PTR

反向记录，通过IP查域名

# 系统性能优化

为了优化系统性能，有时需要对系统进行调整。对于数据库系统，性能调整主要包括CPU/内存使用状况、优化数据库设计、优化数据库管理以及进程/线程状态、硬盘剩余空间、日志文件大小等；对于应用系统，性能调整主要包括应用系统的可用性、响应时间、并发用户数以及特定应用的系统资源占用等。

# 图与开发阶段

1. 需求分析阶段：数据流图
2. 概要设计阶段：模块结构图、层次图和HIPO图
3. 详细设计阶段：程序流程图、伪代码、盒图

# 软件设计

软件设计包括体系结构设计、接口设计、数据设计和过程设计。

1. 结构设计：定义软件系统各主要部件之间的关系。
2. 数据设计：将模型转换成数据结构的定义。好的数据设计将改善程序结构和模块划分，降低过程复杂性。
3. 接口设计（人机界面设计）：软件内部，软件和操作系统间以及软件和人之间 如何通信
4. 过程设计：系统结构部件转换成软件的过程描述。

# EJB

EJB分为会话bean、实体bean和消息驱动bean。

1. 会话bean：用于实现业务逻辑，它可以是有状态的，也可以是无状态的。每当客户端请求时，容器就会选择一个会话bean来为客户端服务。会话bean可以直接访问数据库，但更多时候，它会通过实体bean实现数据访问。
2. 实体bean：用于实现O/R映射，负责将数据库中的表记录映射为内存中的实体对象，事实上，创建一个实体bean对象相当于新建一条记录，删除一个实体bean会同时从数据库中删除对应记录，修改一个实体bean时，容器会自动将实体bean的状态和数据库同步
3. 消息驱动bean：是EJB3.0中引入的新的企业bean，它基于JMS消息，只能接收客户端发送的JMS消息然后处理。MDB实际上是一个异步的无状态会话bean，客户端调用MDB后无需等待，立刻返回，MDB将异步处理客户请求。这适合于需要异步处理请求的场合，比如订单处理，这样就能避免客户端长时间的等待一个方法调用直到返回结果。

# 系统构件组装

分为三个不同的层次：定制（customization）、集成（integration）、扩展（extension）。这三个层次对应于构件组装过程中的不同任务。

# CORBA服务端构件

1. 伺服对象（servant）：CORBA对象的真正实现，负责完成客户端请求
2. 对象适配器（Object adapter）：用于屏蔽ORB内核的实现细节，为服务器对象的实现者提供抽象接口，以便他们使用ORB内部的某些功能
3. 对象请求代理（Object request broker）：解释调用并负责查找实现该请求的对象，将参数传给找到的对象，并调用方法返回结果。客户方不需要了解服务对象的位置、通信方式、实现、激活或存储机制。

# J2EE核心组成

1. 容器：applet container、application container、web container、EJB container
2. 组件：applet、application、JSP/servlet、EJB
3. 服务：HTTP、RMI-IIOP（remote method invocation over the internet inter-ORB protocol）,融合了java RMI和CORBA，在使用application或web端访问EJB端组件时使用
4. Java IDL主要用于访问外部的CORBA服务
5. JTA（java transaction api）用于进行事务处理操作的api
6. Jdbc（java database connectivity）为数据库操作提供的一组api
7. JMS（Java message service）用于发送点对点消息的服务
8. Java Mail：用于发送邮件
9. JAF（java activation framework）用于封装传递的邮件数据
10. JNDI（java naming and directory interface）
11. JAXP（java api for xml parsing）专门用于xml解析操作的api
12. JCA（J2EE connector architecture）：java连接器架构
13. JAAS（java authentication and authorization service）
14. JSF（java service faces）
15. JSTL（JSP standard tag library）
16. SAAJ（SOAP with attachments api for java）
17. JAXR（java api for xml registries）

# 4+1模型

包括：逻辑视图、开发视图、物理视图（部署视图）、进程视图、场景。

# 体系结构权衡分析法（Architecture Tradeoff Analysis Method，ATAM）

ATAM被分为四个主要活动领域（或阶段），分别是

1、场景和需求收集、

2、体系结构视图和场景实现、

3、属性模型构造和分析、

4、折中。

# 基于场景的架构分析方法（Scenarios-based Architecture Analysis Method，SAAM）

SAAM分析评估体系结构的过程包括五个步骤，即

1、场景开发、

2、体系结构描述、

3、单个场景评估、

4、场景交互

5、总体评估。

SAAM的主要输入问题是

1、问题描述、

2、需求声明

3、体系结构描述。

# 仓库风格

在仓库风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据存储上执行。

# 非功能需求

性能需求：指响应时间、吞吐量、准确性、有效性、资源利用率等与系统完成任务效率相关的指标。可靠性、可用性等指标可归为此类。

安全性需求：系统向合法用户提供服务并阻止非授权用户使用服务方面的系统需求

操作性需求：与用户操作使用系统相关的一些需求

文化需求：带有文化背景因素的系统需求

# 实体与类之间有哪些不同之处

信息系统工程方法中实体与面向对象中的类：

1. 实体用于数据建模，而类用于面向对象建模。实体只有属性，而类有属性和操作
2. Essential Use Case可翻译为抽象用例，Real Use Case可翻译为基础用例。他们区别在于：基础用例是实实在在与用户需求有对应关系的用例，是从用户需求获取的渠道得到的，而抽象用例是从基础用例抽取的用例的公共部分，是为了避免重复工作，优化结构而提出的用例

# SOA概念

SOA是一个组件模型，他将应用程序的不同功能单元（称为服务）通过这些组件之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，他应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种这样的系统中的服务可以通过一种统一和通用的方式进行交互。

# ESB作用与特点

1. SOA的一种实现方式，ESB在面向服务的架构中起到的是总线作用，将各种服务进行连接与整合
2. 描述服务的元数据和服务注册管理
3. 在服务请求者和提供者之间传递数据，以及对这些数据进行转化的能力，并支持由实践中总结出来的一些模式如同步模式、异步模式等
4. 发现、路由、匹配和选择的能力，以支持服务之间的动态交互，解耦服务请求者和服务提供者。高级一些的能力，包括对安全的支持、服务质量保证、可管理和负载平衡等。

# 软件测试

软件测试可分为单元测试、集成测试、配置项测试、系统测试、验收测试和回归测试等类别。

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、软件构件或面向对象软件中的类（统称为模块），其目的是检查每个模块能否正确地实现设计说明中的功能、性能、接口和其他设计约束等条件，发现模块内可能存在的各种差错。单元测试的技术依据是软件详细设计说明书。

集成测试的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。集成测试的技术依据是软件概要设计文档。

系统测试的对象是完整的、集成的计算机系统，系统测试的目的是在真实系统工作环境下，验证完整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。系统测试的技术依据是用户需求或开发合同。

配置项测试的对象是软件配置项，配置项测试的目的检验软件配置项与软件需求规格说明的一致性。

确认测试主要验证软件的功能、性能和其他特性是否与用户需求一致。

验收测试是指针对软件需求规格说明，在交付前以用户为主进行的测试。

回归测试的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复合性，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。

# 企业信息化

企业信息化程度是国家信息化建设的基础和关键，企业信息化就是企业利用现代化信息技术，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。企业信息化方法主要包括业务流程重构、核心业务应用、信息系统建设、主题数据库、资源管理、人力资本投资方法。企业战略规划是指依据企业外部环境和自身条件的状况及其变化来制定和实施战略，并根据对实施过程与结果的评价和反馈来调整，制定新战略的过程。

# 分布式数据库两阶段提交协议

二阶段提交（two-phase commit）是指，在计算机网络以及数据库领域内，为了使基于分布式系统架构下的所有节点在进行事务提交时保持一致性而设计的一种算法。通常，二阶段提交也被称为是一种协议。在分布式系统中，每个节点虽然可以知晓自己的操作成功或者失败，却无法知道其他节点的操作成功或失败。当一个事务跨越多个节点时，为了保持事务的ACID特性，需要引入一个作为协调者的组件来统一掌控所有节点的操作结果并最终指示这些节点是否要把操作结果进行真正的提交。因此，二阶段提交的算法思路可以概括为：参与者将操作成败通知协调者，再由协调者根据所有参与者的反馈情报决定各参与者是否要提交操作还是中止操作。

所谓的两个阶段是指：

第一阶段：准备阶段（表决阶段），事务协调者（事务管理器）给每个参与者（资源管理器）发送prepare消息，每个参与者要么直接返回失败（如权限验证失败），要么再本地执行事务，写本地的redo和undo日志，但不提交，到达一种万事俱备，只欠东风的状态。

第二阶段：提交阶段（执行阶段），如果协调者收到了参与者的失败消息或者超时，直接给每个参与者发送回滚消息，否则发送提交消息，参与者根据协调者的指令执行提交或者回滚操作，释放所有事务处理过程中使用的锁资源（注意：必须再最后阶段释放锁资源）。

# 元组演算式解释

找出这样的元组t（t是R中的元组），t要满足这样的条件：存在u（u是S关系中的元组），u第二列值大于t的第三列的值。

# 层次化路由

层次化路由的含义是指对网络拓扑结构和配置的了解是局部的，一台路由器不需要知道所有的路由信息，只需要了解其管辖的路由信息，层次化路由选择需要配合层次化的地址编码。而子网或超网就属于层次化地址编码行为。

# 信息化需求

一般来说，信息化需求包含三个层次，即战略需求、运作需求和技术需求。

一是战略需求。组织信息化的目标是提升组织的竞争能力，为组织的可持续发展提供一个支持环境。从某种意义上来说，信息化对组织不仅仅是服务的手段和实现现有战略的辅助工具；信息化可以把组织战略提升到一个新的水平，为组织带来新的发展契机。特别是对于企业，信息化战略是企业竞争的基础。

二是运作需求。组织信息化的运作需求是组织信息化需求非常重要且关键的一环，它包含三方面的内容：一是实现信息化战略目标的需要；二是运作策略的需要；三是人才培养的需要。

三是技术需求。由于系统开发时间过长等问题在信息技术层面上对系统的完善、升级、集成和整合提出了需求。也有的组织，原来基本上没有大型的信息化系统项目，有的也只是一些单机应用，这样的组织的信息化需求，一般是从头开发新的系统。

# 系统移植

移植工作大体上分为计划阶段、准备阶段、转换阶段、测试阶段、验证阶段。

1. 计划阶段，在计划阶段，要进行现有系统的调查整理，从 移植技术、系统内容（是否进行系统提炼等）、系统运行三个方面，探讨如何转换成新系统，决定移植方法，确立移植工作体制以及移植日程。
2. 准备阶段，在准备阶段要进行移植方面的研究，准备转换所需的资料。该阶段的作业质量将对以后的生产效率产生很大的影响。
3. 转换阶段，这一阶段是将程序设计和数据转换成新机器能根据需要工作的阶段。提高转换工作的精度，减轻下一阶段的测试负担是提高移植工作效率的基本内容。
4. 测试阶段，这一阶段是进行程序单元、工作单元测试的阶段。在本阶段要核实程序能否在新系统中准确的工作。所以，当有不能准确工作的程序时，就要回到转换阶段重新工作。
5. 验证阶段，这是测试完的程序使新系统工作，最后核实系统，准备正式运行的阶段。

# 基于软件架构的设计

根据基于软件架构的设计的定义，基于软件架构的设计（Architecture Based Software Development，ABSD）强调由商业、质量和功能需求的组合驱动软件架构设计。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）。

# 体系结构文档化（架构文档化）

体系结构文档化过程的主要输出结果是体系机构规格说明和测试体系结构需求的质量设计说明书这两个文档。软件体系结构的文档要求与软件开发项目中的其他文档是类似的。文档的完整性和质量是软件体系结构成功的关键因素。文档要从使用者的角度进行编写，必须分发给所有与系统有关的开发人员，且必须保证开发者手上的文档是最新的

# 体系结构风格

体系结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。对软件体系结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。例如，如果某人把系统描述为客户/服务器模式，则不必给出设计细节，我们立刻就会明白系统是如何组织和工作的。

语音识别是黑板风格的经典应用场景。

输入某个构 件，经过内部处理，产生数据输出的系统，正是管道-过滤器中的过滤器的职能，把多个多滤器使用管道相联的风格为管道-过滤器风格。

# C2体系结构风格

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他构件和连接件连接。
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

# 事务管理

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

（1）原子性（Atomicity）：数据库的逻辑工作单位。

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

（3）隔离性（Isolation）：不能被其他事务干扰。

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

# 质量属性效用树（utility tree）

对系统质量属性进行识别和优先级排序的重要工具。效用树主要关注性能、可修改性、可用性和安全四个方面。

# 系统架构风险

系统架构风险是指架构设计中潜在的、存在问题的架构决策所带来的隐患。敏感点是为了实现某种特定质量属性，一个或多个系统组件所具有的特性。权衡点是影响多个质量属性，并对多个质量属性来说都是敏感点的系统属性。

# 面向对象方法

在面向对象方法中，信息流是通过向参与者或内部对象发送信息形成的。状态图描述了一个对象在其生命周期中可能的状态组合；顺序图用来描述对象按照时间顺序的消息流来建模用例；数据流图是一种描述数据通过系统的流程以及系统实施的工作或处理过程的过程模型；流程图以图形化的方式展示应用程序从数据输入开始到获得输出为止的逻辑过程。

# 软件架构风格

数据流风格：批处理序列和管道-过滤器

调用返回风格：主程序/子程序、面向对象风格和层次结构

独立构件风格：进程通讯和事件系统子风格

虚拟机风格：解释器和规则为中心

仓库风格：数据库系统、超文本系统和黑板风格

# 层次系统架构风格

二层及三层C/S架构风格

B/S架构风格

MVC架构风格

MVP架构风格

# 实时系统

实时系统是指向系统发出一个指令后，在一个极短的时间内，系统回复结果。

实时系统的特性：

1. 时间约束性（及时性）
2. 可预测性
3. 高可靠性
4. 与外部环境的交互作用性
5. 多任务类型
6. 约束的复杂性
7. 具有短暂超载的特点

# 软件错误、缺陷、故障、失效

软件错误：是指在软件生存期内的不希望或不可接受的人为错误，其结果是导致软件缺陷的产生

软件缺陷：是存在于软件（文档、数据、程序）之中的那些不希望或不可接受的偏差

软件故障：软件故障是指软件运行过程中出现的一种不希望或不可接受的内部状态

软件失效：是指软件运行时产生的一种不希望或不可接受的外部行为结果

# PHP缺点

PHP是面向过程的语言，只要业务流程发送变化，修改工作量很大，所以可修改性差，同时可复用性也差。

PHP语言在可靠性方面比J2EE平台差，J2EE平台有大量增强可靠性的成熟解决方案，而PHP只是一种简单的脚本语言，在可靠性方面缺乏成熟的解决方案。

PHP对于不同的数据库采用不同的数据库访问接口，而JAVA通过JDBC来访问数据库，通过不同的数据库厂商提供的数据库驱动方便地访问数据库，访问数据库的接口比较统一。

# 应用服务器

应用服务器是指通过各种协议把商业逻辑暴露给客户端的程序。

1. 若系统负荷很大，可以部署多台应用服务器，多台应用服务器分担任务，以达到性能要求。
2. 应用服务器可以通过灵活的增加服务器完成扩展，所以可扩展性很好。
3. 应用服务器可长时间稳定运行。因为当一台应用服务器出现故障时，可以将当前运行的事务转移至正常应用服务器上完成执行，不影响业务正常执行，从而保障高可靠性。

# 软件架构风格

架构风格定义一个系统家族，即一个体系结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型和子系统有效的组织成一个完整的系统。

# 软件质量属性

软件质量属性描述了软件的非功能性属性。可用性质量属性描述了可用性是系统能够正常运行的时间比例，实现可用性策略的主要方法有错误检测、错误恢复和错误防御。主动冗余就是一种错误恢复的策略。性能是指系统的响应能力，即要经过多长时间才能对某个事件作出响应，或者在某段时间内系统所能处理事件的个数。队列调度是一种提升系统性能的常用方法。