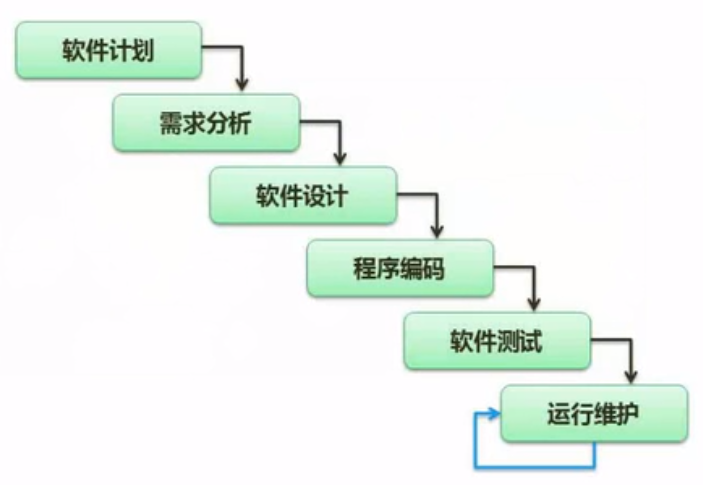
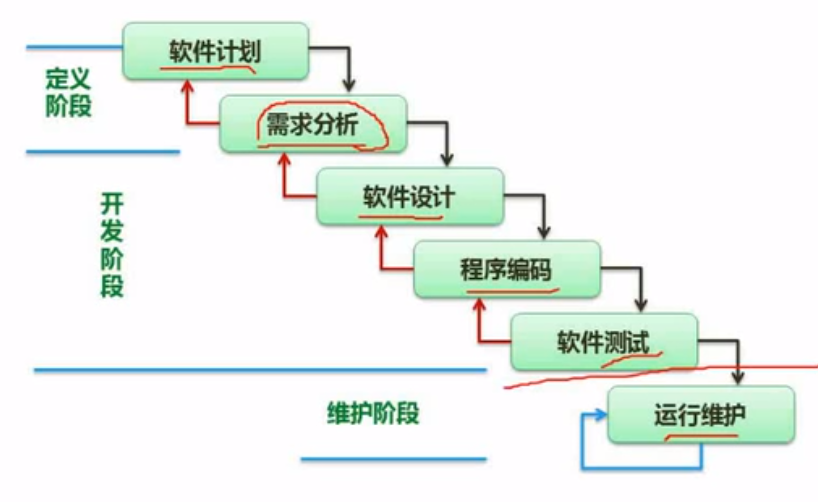
# 软件开发模型

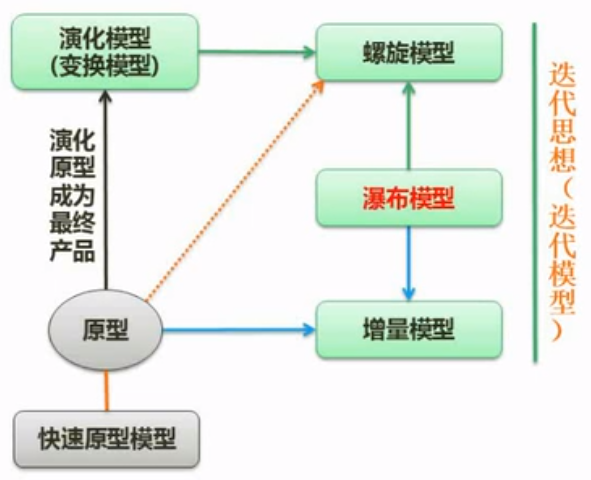
|  |  |
| --- | --- |
| 瀑布模型  演化模型  增量模型  螺旋模型  快速原型模型  喷泉模型  V模型 | 迭代模型/迭代开发方法  快速应用开发  构件组装模型/基于构件的开发方法  统一过程/统一开发方法  敏捷开发方法  模型驱动的开发方法  基于架构的开发方法 |

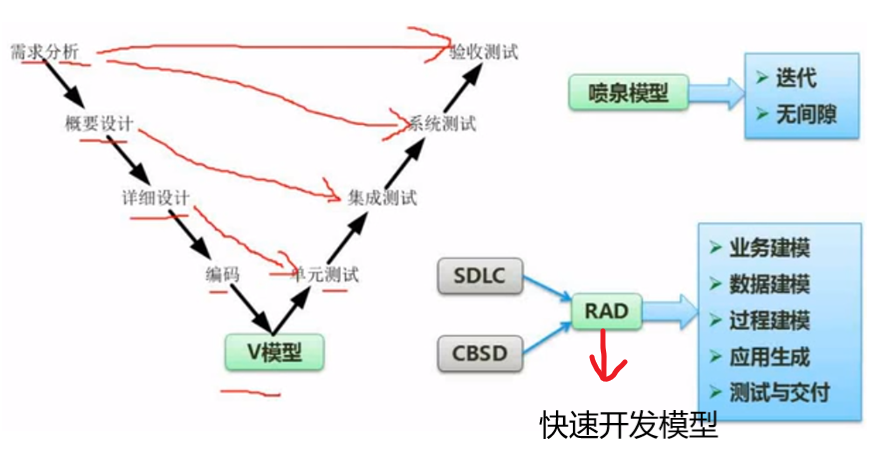
## 瀑布模型（SDLC）



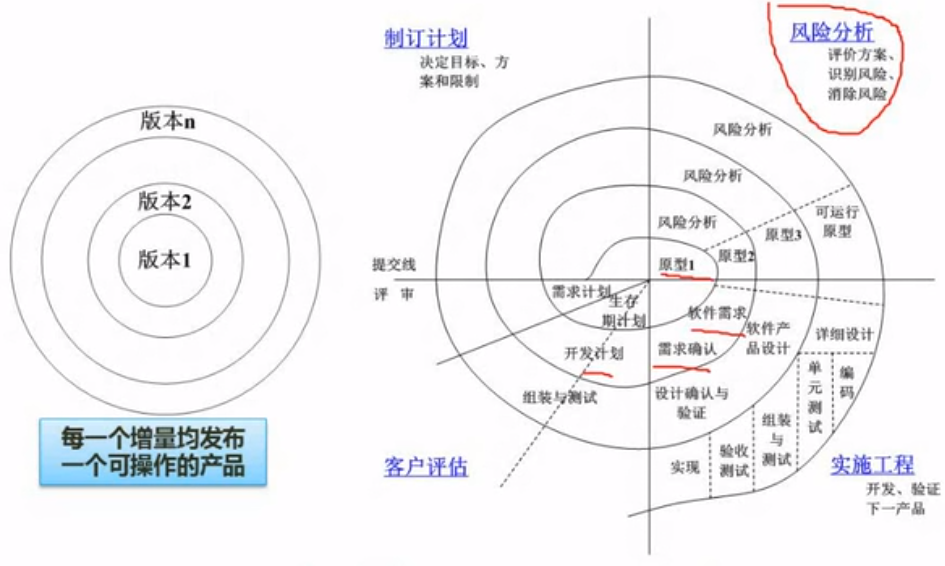


## 其他经典模型

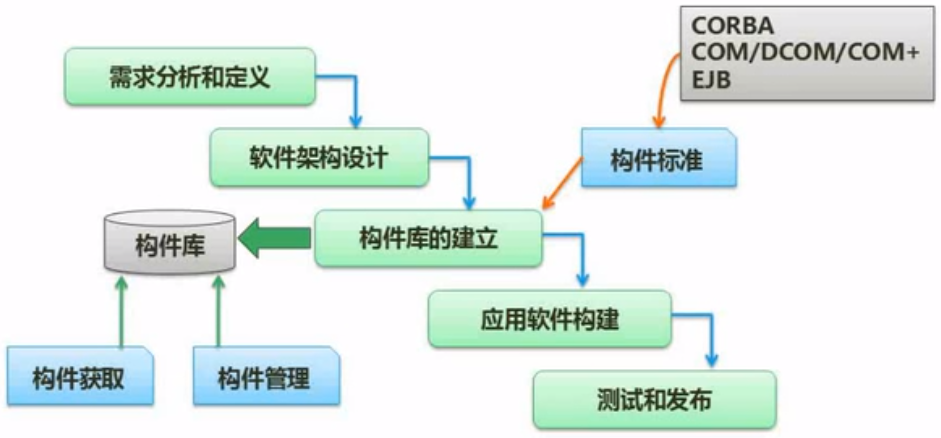




## 增量模型与螺旋模型



## 构建组装模型（CBSD）



### EJB

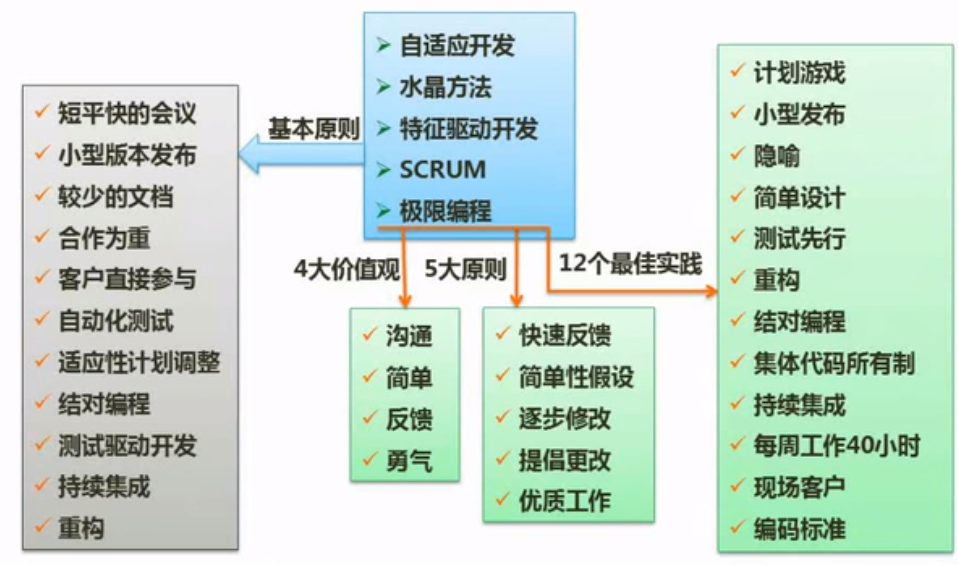
EJB是sun的javaEE服务器端组件模型，设计目标与核心应用是部署分布式应用程序。简单来说就是把已经编写好的程序（即：类）打包放在服务器上执行。凭借java跨平台的优势，用EJB技术部署的分布式系统可以不限于特定的平台。EJB（Enterprise JavaBean）是J2EE的一部分，定义了一个用于开发基于组件的企业多重应用程序的标准。其特点包括网络服务中心支持和核心开发工具（SDK）。在J2EE里，Enterprise Java Beans（EJB）称为Java企业Bean，是Java的核心代码，分别是会话Bean（Session Bean），实体Bean（Entity Bean）和消息驱动Bean（Message Bean）。在EJB3.0推出以后，实体Bean被单独分了出来，形成新的规范JPA。

#### 简介

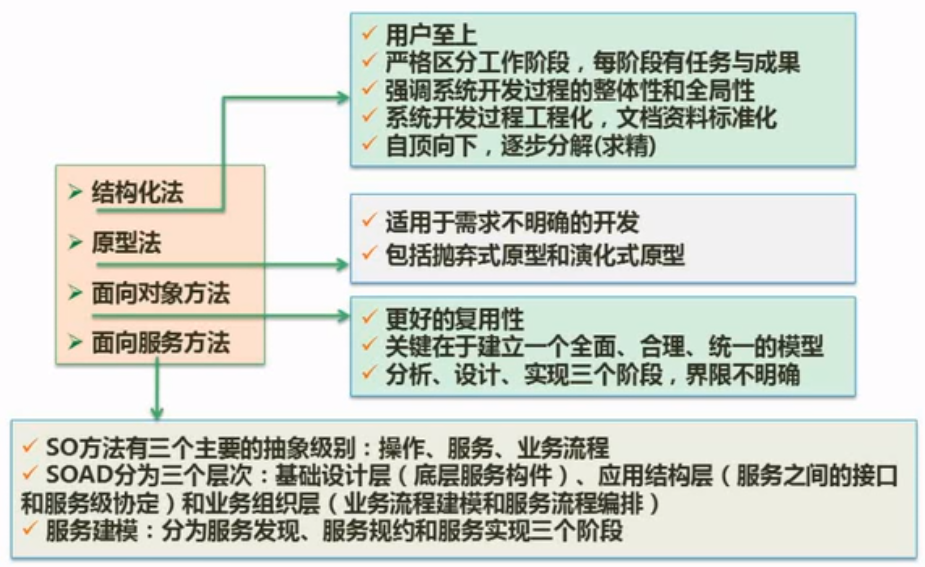
1. SessionBean用于实现业务逻辑，它可以是有状态的，也可以是无状态的。每当客户端请求时，容器就会选择一个SessionBean来为客户端服务。Session Bean可以直接访问数据库，但更多时候，它会通过Entity Bean实现数据访问。
2. EntityBean是域模型对象，用于实现O/R映射，负责将数据库中的表记录映射为内存中的Entity对象，事实上，创建一个EntityBean对象相当于新建一条记录，删除一个EntityBean会同时从数据库中删除对应的记录，修改一个EntityBean时，容器会自动将Entity Bean的状态和数据库同步。
3. MessageDriven Bean是EJB2.0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SessionBean，客户端调用MDB后无需等待，立刻返回，MDB将异步处理客户端请求。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。

EJB实际上是SUN的J2EE中的一套规范，并且规定了一系列的API用来实现把EJB概念转换成EJB产品。

## 敏捷开发方法

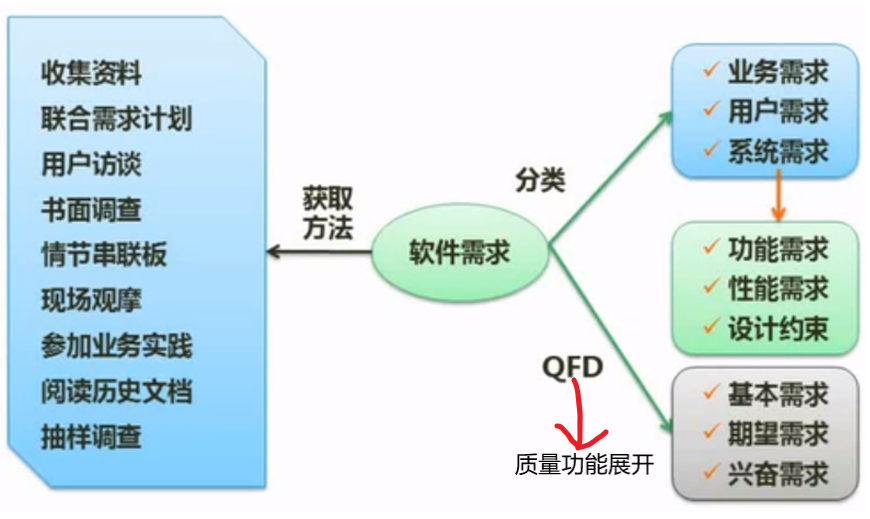


## 信息系统开发方法



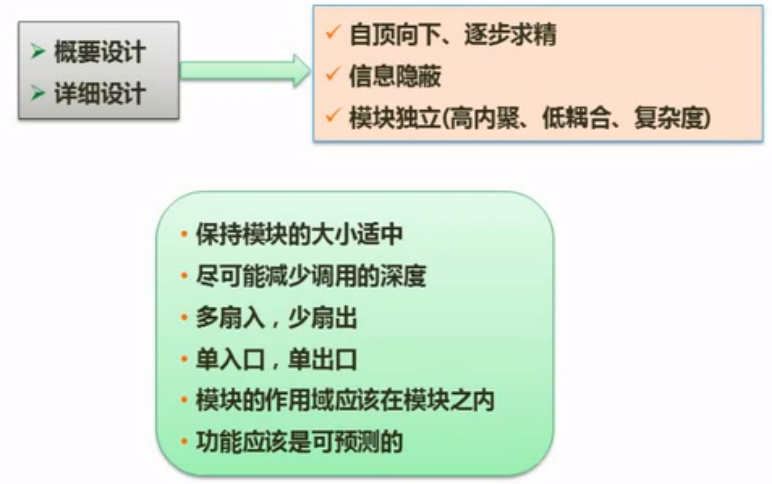
## 需求开发

### 需求分类于需求获取



## 结构化设计

### 基本原则

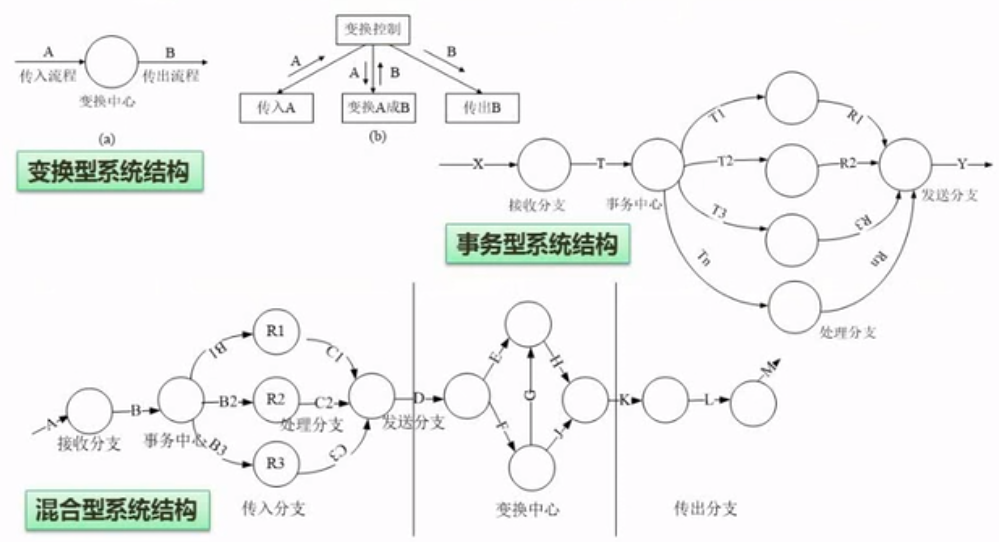


### 内聚与耦合

|  |  |
| --- | --- |
| 内聚类型 | 描述 |
| 功能内聚 | 完成一个单一功能，各个部分协同工作，缺一不可 |
| 顺序内聚 | 处理元素相关，而且必须顺序执行 |
| 通信内聚 | 所有处理元素集中在一个数据结构的区域上 |
| 过程内聚 | 处理元素相关，而且必须按特定的次序执行 |
| 瞬时内聚（时间内聚） | 所包含的任务必须在同一时间间隔内执行 |
| 逻辑内聚 | 完成逻辑上相关的一组任务 |
| 偶然内聚（巧合内聚） | 完成一组没有关系或松散关系的任务 |

|  |  |
| --- | --- |
| 耦合类型 | 描述 |
| 非直接耦合 | 两个模块之间没有直接关系，他们之间的联系完全是通过主模块的控制和调用来实现的 |
| 数据耦合 | 一组模块借助参数表传递简单数据 |
| 标记耦合 | 一组模块通过参数表传递记录信息（数据结构） |
| 控制耦合 | 模块之间传递的信息中包含用于控制模块内部逻辑的信息 |
| 外部耦合 | 一组模块都访问同一全局简单变量，而且不是通过参数表传递该全局变量的信息 |
| 公共耦合 | 多个模块都访问同一个公共数据环境 |
| 内容耦合 | 一个模块直接访问另一个模块的内部数据；一个模块不通过正常入口转到另一个模块的内部；两个模块有一部分程序代码重叠；一个模块有多个入口 |

### 系统结构/模块结构



## 软件测试

### 测试原则与类型

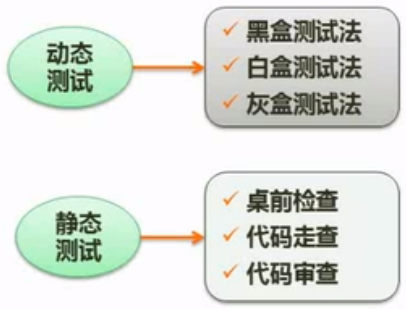
尽早、不断的进行测试

程序员避免测试自己设计的程序

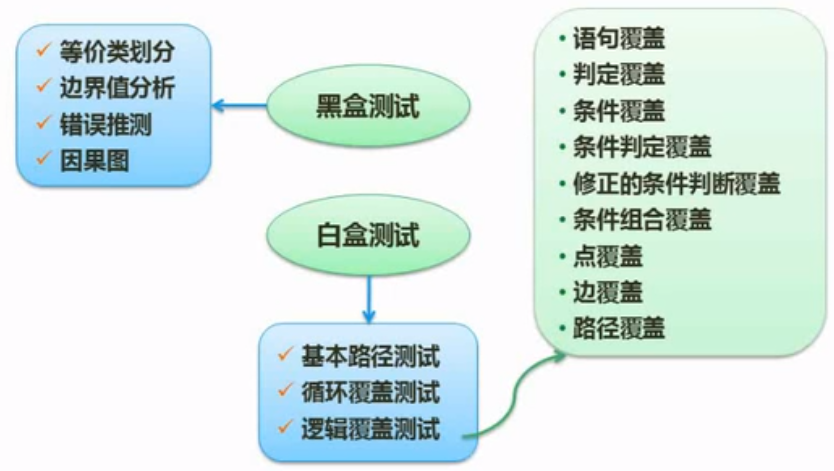
既要选择有效、合理的数据，也要选择无效、不合理的数据

修改后应进行回归测试

尚未发现的错误数量与该程序以发现错误数成正比



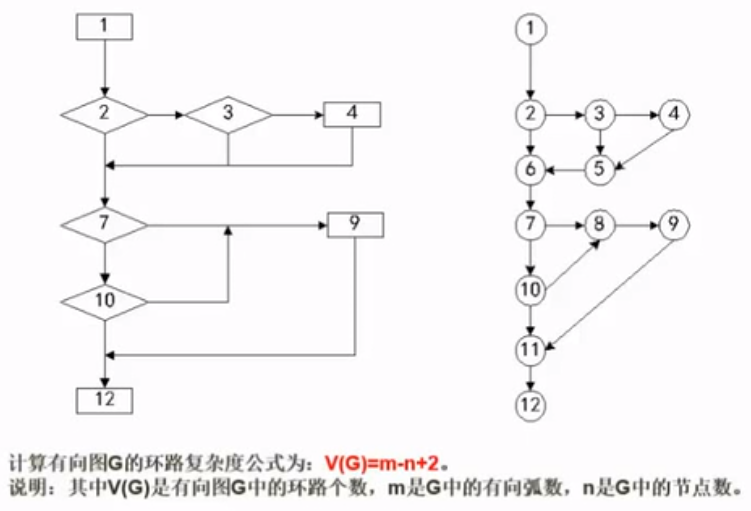
### 测试用例设计



### 测试阶段



### McCab复杂度



另一种算法：流图G的环形复杂度V(G)=P+1，其中，P是流图中判定结点的数目。

## 系统运行与维护

软件维护是生命周期的一个完整部分。可以将软件维护定义为需要提供软件支持的全部活动，这些活动包括在交付前完成的活动，以及交付后完成的活动。交付前完成的活动包括交付后运行的计划和维护计划等；交付后的活动包括软件修改、培训、帮助资料等。



系统交付使用后，改变系统的任何工作，都可以被称为维护。与硬件不同，软件系统构建时就包含了变化，软件并不会老化或需要周期性的维护。软件维护是指在软件交付使用后，直至软件被淘汰的整个时期内，为了改正错误或满足新的需求而修改软件的活动。软件的维护活动基于“软件是可维护的”这一基本前提。

软件可维护性

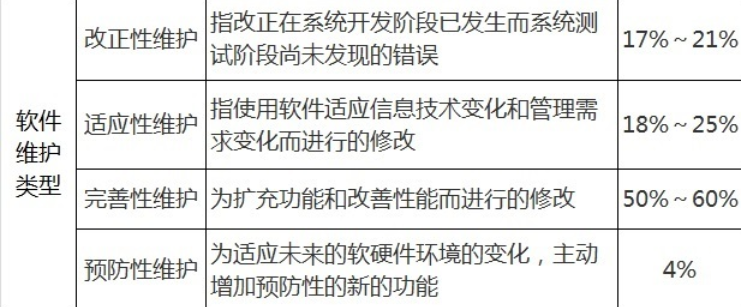
根据国家标准GB/T 16260.1-2006，软件可维护性是指软件产品被修改的能力，修改包括纠正、改进或软件对环境、需求和功能规格说明变化的适应。GB/T16260.1-2006标准还规定了可维护性的五个子特征：

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。如果软件由最终用户修改，那么易改变性可能会影响易操作性。
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。

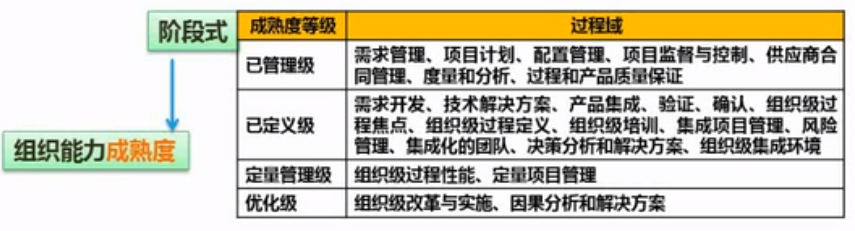
软件维护的分类

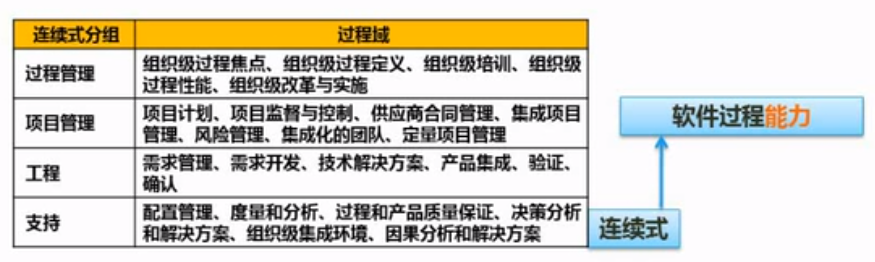
在系统运行过程中，软件需要维护的原因是多样的，根据维护的原因不同，可以将软件维护分为以下四种：

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就称为改正性维护。
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/输出方式、数据存储介质）可能发生变化。为使软件适应这种变化，而去修改软件的过程就称为适应性维护。
3. 完善性维护。在软件使用过程中，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。这种情况下进行的维护活动称为完善性维护。
4. 预防性维护。这是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基础。通常、预防性维护可定义为“把今天的方法用于昨天的系统以满足明天的需要”。也就是说，采用先进的软件工程方法对需要维护的软件或软件中的某一部分（重新）进行设计、编码和测试。



## 软件过程改进-CMMI





CMMI全称是Capability Maturity Model Integration，即能力成熟度模型集成（也有称为：软件能力成熟度集成模型），是美国国防部的一个设想，1994年由美国国防部（United States Department of Defense）与卡内基-梅隆大学（Carnegie-Mellon University）下的软件工程研究中心（Software Engineering Insititute ，SEISM）以及美国国防工业协会（National Defense Industrial Association）共同开发和研制的，他们计划把现在所有现存实施的与即将被发展出来的各种能力成熟度模型集成到一个框架中去，申请此认证的前提条件是该企业具有有效的软件企业认定证书。

其目的是帮助软件企业对软件工程进行管理和改进，增强开发与改进能力，从而能按时地，不超预算地开发出高质量的软件。其所依据的想法是：只要集中精力持续努力去建立有效的软件工程过程的基础结构，不断进行管理的实践和过程的改进，就可以克服软件开发中的困难。CMMI为改进一个组织的各种过程提供了一个单一的集成化框架，新的集成模型框架消除了各个模型的不一致性，减少了模型间的重复，增加透明度和理解，建立了一个自动的、可扩展的框架。因而能够从总体上改进组织的质量和效率。CMMI主要关注点就是成本效益、明确重点、过程集中和灵活性四个方面。

## 项目管理

### 九大知识领域

范围管理

时间管理

成本管理

质量管理

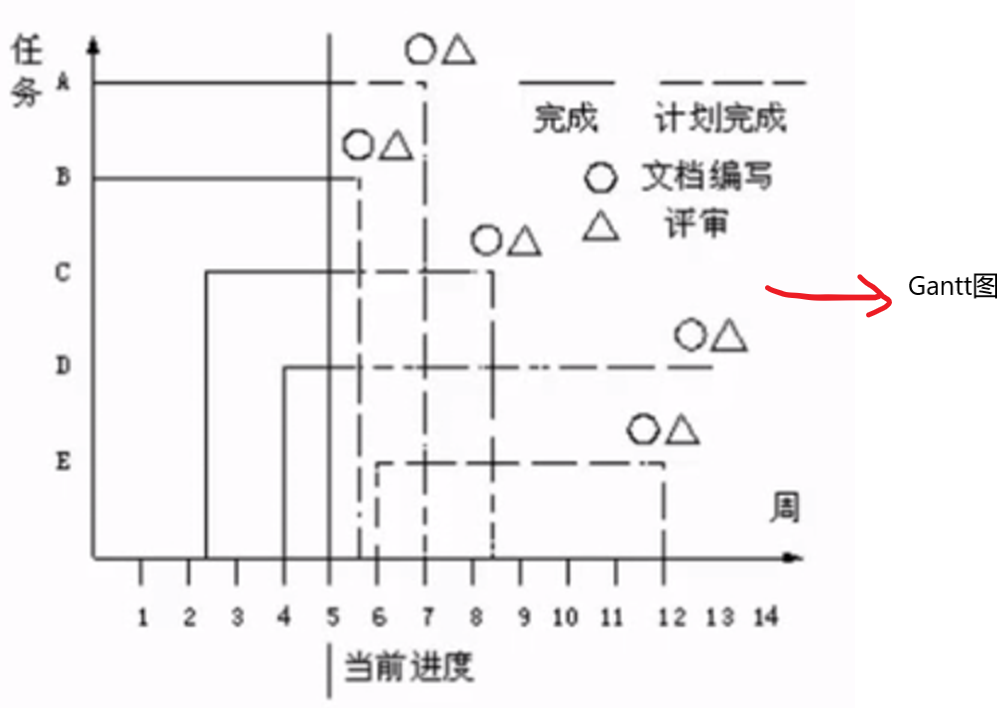
人力资源管理

沟通管理

风险管理

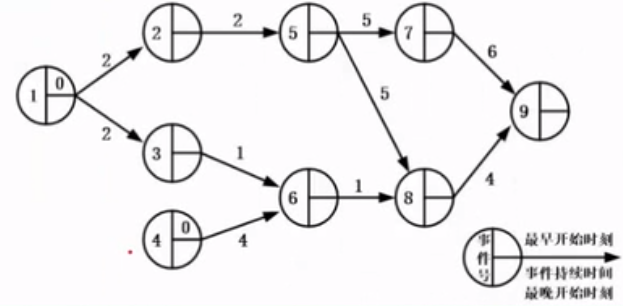
采购管理

整体管理



### 案例

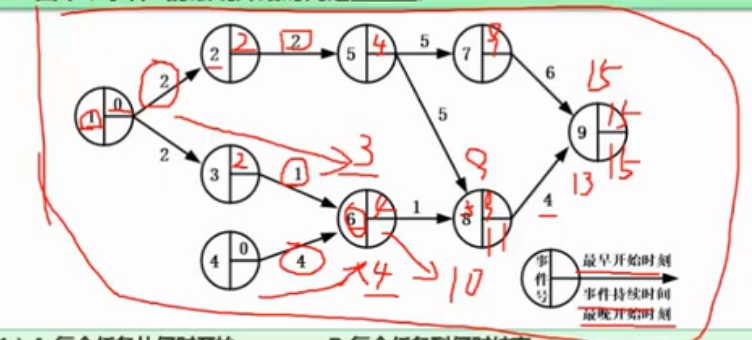
进度安排的常用图形描述方法有Gantt图和PERT图。Gantt图不能清晰的描述（D）；PERT图可以给出哪些任务完成后才能开始另一些任务。下图所示PERT图中，事件6的最晚开始时间是（C）。



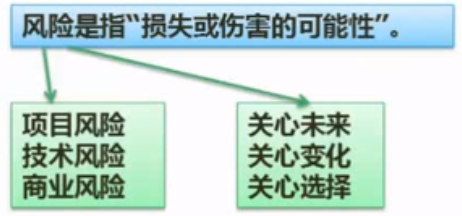
（1）A、每个任务从何时开始 B、每个任务到何时结束

C、每个任务的进展情况 D、各个任务之间的依赖关系

（2）A、0 B、3 C、10 D、11



### 风险



风险曝光度（Risk Exposure）：计算方法是风险出现的概率乘以风险可能造成的损失。

假设正在开发的软件项目可能存在一个未被发现的错误，而这个错误出现的概率是0.5%，给公司造成的损失将是1000000元，那么这个错误的风险曝光度就应为：1000000\*0.5% = 5000元。