### 软件生命周期

软件生命周期

软件生命周期是软件开发的全过程，由诸多阶段构成，包括需求分析阶段，软件设计阶段，编码及单元测试阶段，集成及系统测试阶段，安装 ，实施与维护阶段。

各阶段的任务（核心）

1需求分析阶段：

1. 确定软件的综合要求：系统界面，系统功能，系统性能，安全性，保密性，可靠性，系统的运行要求，异常处理，将来的扩充和修改。
2. 分析软件系统的数据要求，基本数据元素，数据元素之间的逻辑关系，数据量，峰值。
3. 导出系统的逻辑模型
4. 修正项目开发计划。

这一阶段最重要的是软件说明书，主要内容是系统的数据描述，数据流图，数据字典描述，系统接口描述，内部接口说明，系统功能描述，处理说明，系统设计性能描述，性能参数，系统测试种类。

---------------------------------------------------------------------------------------------------------------------------------

（一）问题定义

关键任务：“要解决的问题是什么”

问题定义报告（文档）：通过对客户的访问调查，系统分析员扼要地写出关于问题性质、目标工程和工程规模的书面报告，经过讨论和必要修改之后这份报告必须应得到客户的确认。

（二）可行性研究

关键任务：回答关键问题是：“对于上一个阶段所确定的问题有行的通的解决办法吗？”

 在较抽象的高层次上进行的分析和设计过程。

目的：是客户作出是否继续进行这项工程的决定的重要依据

可行性研究报告（文档）：经济（成本效益）、技术（技术可行）、社会（操作公共安全社会道德）

（三）需求分析

关键任务：目标系统必须做什么

目的：完整、准确、清晰、具体的建立系统逻辑模型，是以后设计和实现目标系统的基础。

要求：必须准确完整地体现用户的需求

（四）总体设计

关键任务：怎样实现目标系统

    根据需求设计方案，分析推荐最佳方案，设计软件结构等。

    设计程序的体系结构，月就是确定程序有哪些模块组成以及模块之间的关系

（五）详细设计

关键任务：该怎样具体实现系统？

    设计每个模块的算法和数据结构

（六）编码和单元测试

关键任务:选择语言、工具翻译详细设计结果、测试模块

实现阶段文档：

    1、程序清单 2、单元测试报告

（七）综合测试

关键任务：通过各类型的测试（及相应的调试）是软件达到预定要求

（八）软件维护

关键任务：通过各种必要的维护活动使系统持久地满足用户的需求

1、改正性维护：软件运行过程中发现错误进行维护

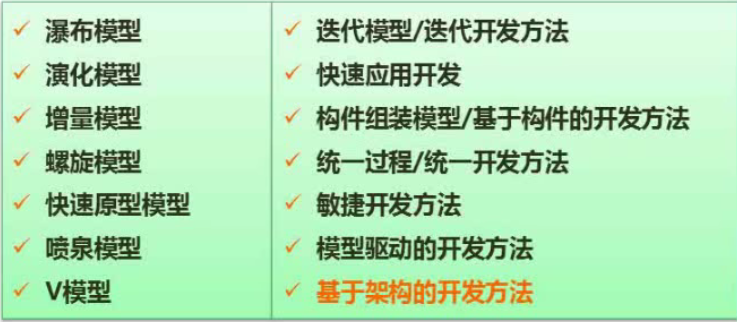
2、适应性维护：软件运行软件硬件环境变化进行的维护

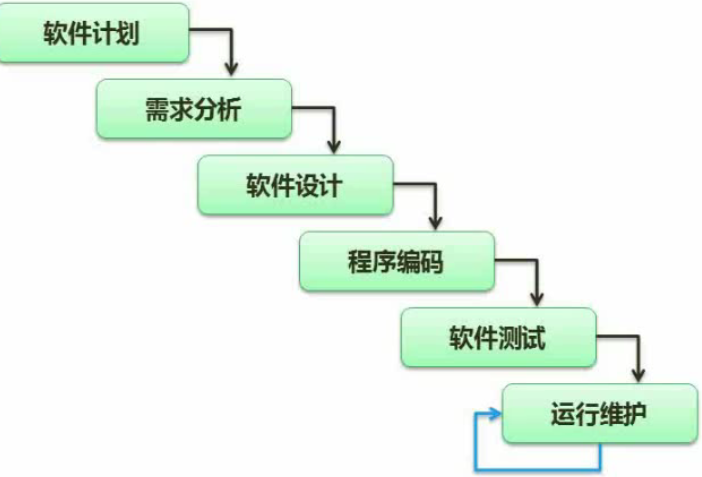
3、完善性维护：用户要求改进或扩充软件进行的维护

4、预防性维护：为将来的维护作准备

### 软件开发模型

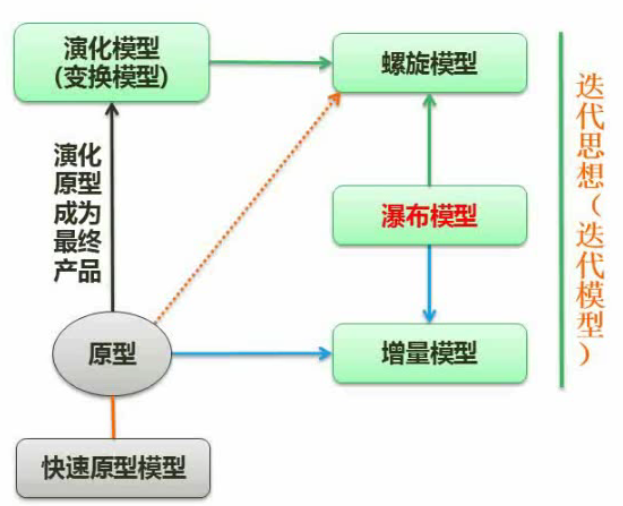
瀑布模型





瀑布模型只适用于需求明确的开发，瀑布模型属于结构化方法模型，严格遵守软件生命周期各阶段固定顺序。（上面6大顺序）

原型模型



原型是在初期（需求不明确）完成初步系统然后与用户交流，若获得用户明确需求则放弃原型。而若在交流过程中演化原型成为最终产品，则为演化模型，若原型和瀑布模型基础上一步一步分段完成为增量模型

演化模型

演化模型也属于原型化开发方法。在原型的基础上，若一直到结束未获得明确用户需求，而最终完成项目即为演化模型。

增量模型

增量模型也是在原型的基础上，若在其基础上采用瀑布模型即为增量模型。

喷泉模型

喷泉模型主要用于描述面向对象的开发过程，体现开发过程的迭代和无间隙特征。

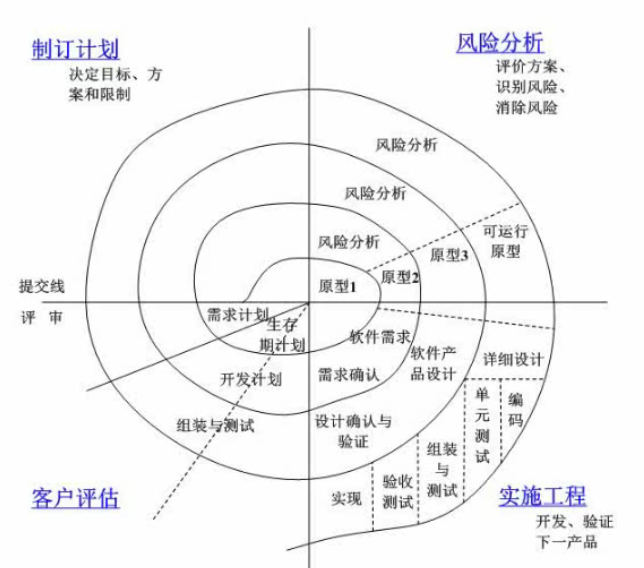
喷泉模型

喷泉模型属于面向对象模型 特点：迭代 无间隙。

RAD 快速开发模型 包括 瀑布模型和构建组装模型（构建化开发）。

螺旋模型

螺旋模型结合瀑布模型和演化模型的特点。



### 软件开发方法论

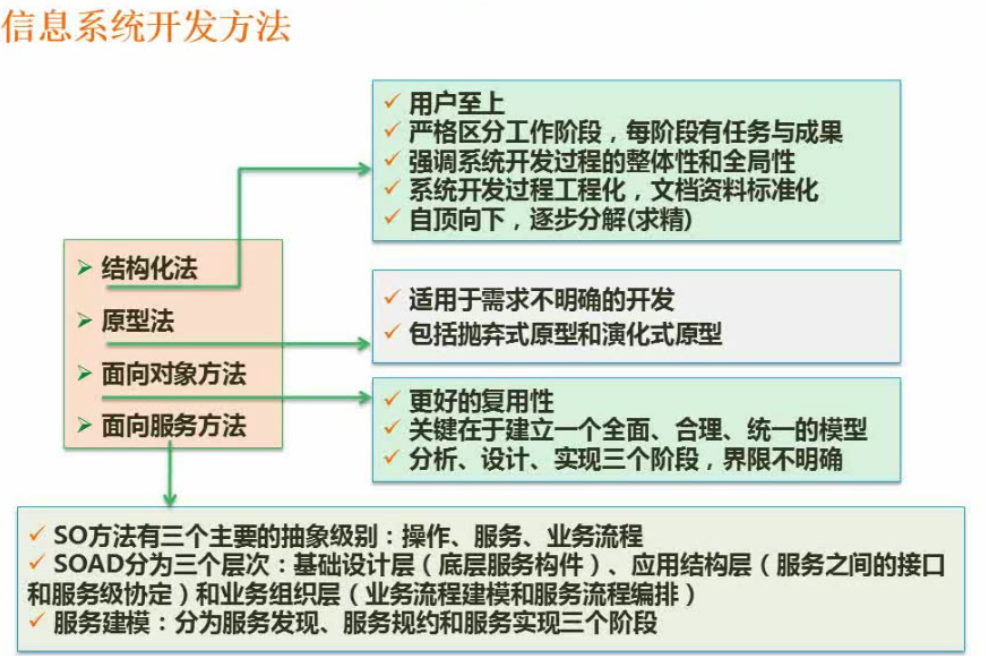
系统开发策略（新系统替换旧系统）

直接转换待新系统开发完毕，直接启用新系统，关闭旧系统。

并行转换让新老系统并行一段时间，安全可靠，但工作量大。用于银行等大型系统。

分段转换渐渐转换，试点过渡，是前两种的结合。一部分一部分的替代老系统。

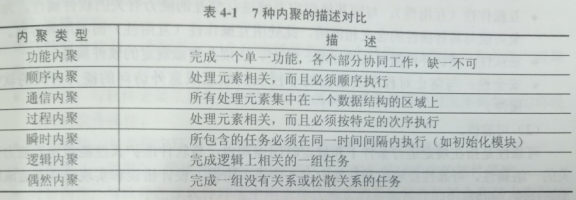
信息系统开发方法



模块设计原则（核心）

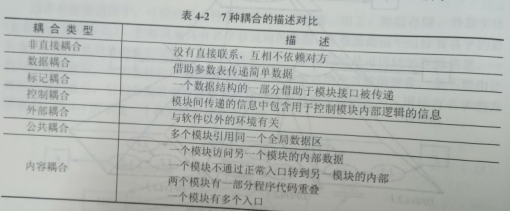
7种内聚理解

聚合度高到低



7种耦合理解

耦合度低到高



软件设计原则：高内聚，低耦合。内聚是指模块内部各元素之间的联系紧密程度，表现为功能的好坏。耦合是指模块之间相互联系的紧密程度。

软件开发工具

数据流图

数据流图是结构化开发方法中需求阶段的开发工具。数据流图中有四种基本符号

1. 数据流：是具有名字和流向的数据，在数据流图中用标有名字的箭头表示。
2. 加工：加工是对数据流的变换，一般用圆圈表示。
3. 数据存储：是可访问的存储信息，一般直线表示
4. 外部实体：位于被建模的系统之外的信息生产者或消费者，不能被计算机识别。

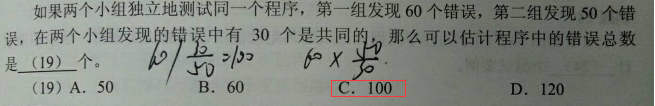
分别表明数据处理过程的数据来源和数据去向。用标有名字的方框表示

数据流图的基本步骤及原则：自顶向下逐层分解

1. 数据流图上的所有图形符号只限于4种基本图形元素。
2. 顶层数据流图必须包括4种基本图形元素，缺一不可。
3. 每个加工最少有一个数据输入流和一个输出数据流。
4. 顶层数据流上的数据流必须封闭在外部实体之间。
5. 在数据流图中，需按层给框加编号。注意父图与子图的对应关系。
6. 规定任何一个数据流子图必须与上层加工对应，输入输出一致。
7. 可以在数据流图中加入物质流，帮助客户理解数据流图。
8. 图上每个元素都有名字。
9. 数据流图中不可夹带控制流。

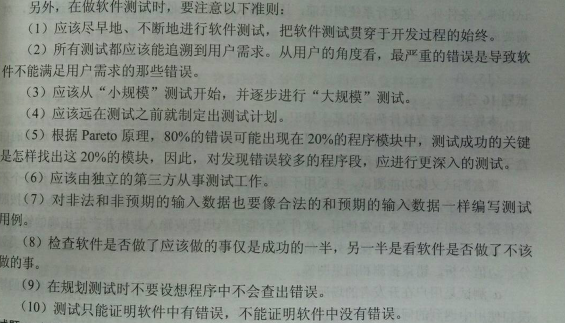
### 软件测试

软件测试错误的计算



软件测试的目的与准则

软件测试追求的目标是以尽可能少的时间和人力发现软件产品中尽可能多的错误。



软件测试的分类

白盒测试：（结构测试）用于单元测试阶段，把程序封装于透明的盒子里，测试者完全知道

程序的结构及处理算法。

黑盒测试：（功能测试）用于集成测试和确认测试阶段，封装于不透明盒子，完全不了解

部程序的结构及处理算法。只检查软件功能是否按照需求说明书的要求正常使用

α测试：是用户在开发者的场所由开发者指导完成的测试。开发者记录错误和问题。

β测试：是一个或多个用户的现场由该软件的最终用户实施的，开发者不在现场，拥护者记录错误和问题，并反馈给开发者。

回归测试：是测试软件变更之后，变更部分的正确性和需求的符合性。只要软件变更，就要进行相应的回归测试。

白盒测试23

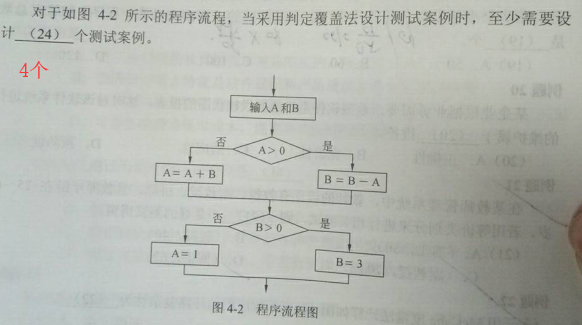
1. 语句覆盖



1. 判定覆盖



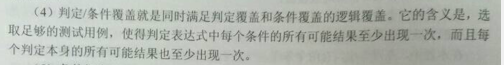
每个判定可能的结果都要测试。



1. 条件覆盖



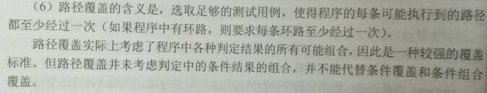
1. 判定/条件覆盖



1. 组合条件覆盖



1. 路径覆盖



### 软件维护

软件维护四种分类

1. 改正性维护

是指改正在系统开发阶段已发生而系统测试阶段尚未发现的错误。

1. 完善性维护

是指使用软件适应信息技术变化和管理需求变化而进行的修改。

1. 适用性维护

这是为扩充功能和改善性能而进行的修改，主要是指对已有的软件系统增加一些在系统分析和设计阶段中没有规定的功能与性能特征。这些功能对完善系统功能足非常必要的。

1. 预防性维护

为了改进应用软件的可靠性和可维护性，为了适应未来的软硬件环境的变化，应主动增加预防性的新的功能，以使应用系统适应各类变化而不被淘汰。

模块的划分

模块的作用范围在其控制范围之内

极限编程

四大价值观：简单 沟通 反馈 勇气

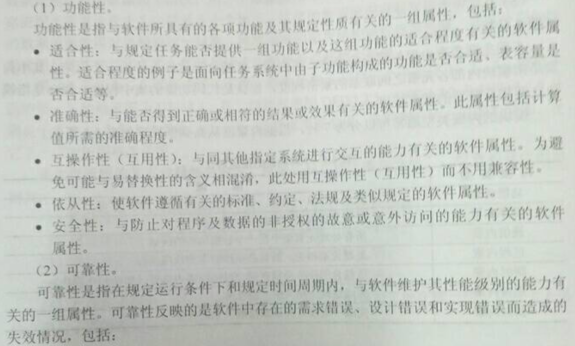
五大原则 快速反馈 简单性假设 逐步修改 提倡更改 优质工作

十二最佳实践 计划游戏 小型发布 隐喻 简单设计 测试先行 重构 集体代码所有制

结对编程 每周工作40小时 持续集成 编码标准 现场客户

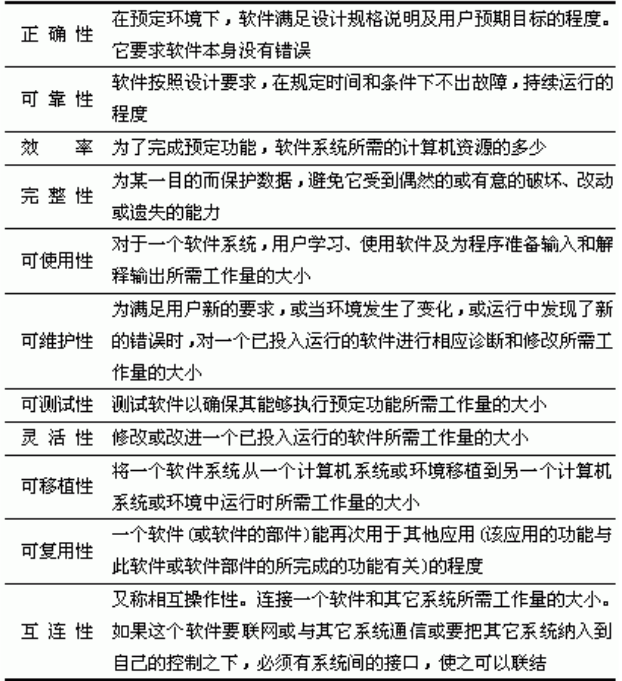
### 软件管理质量

ISO/IEC 9126软件质量模型6个质量特性，21个子特性。



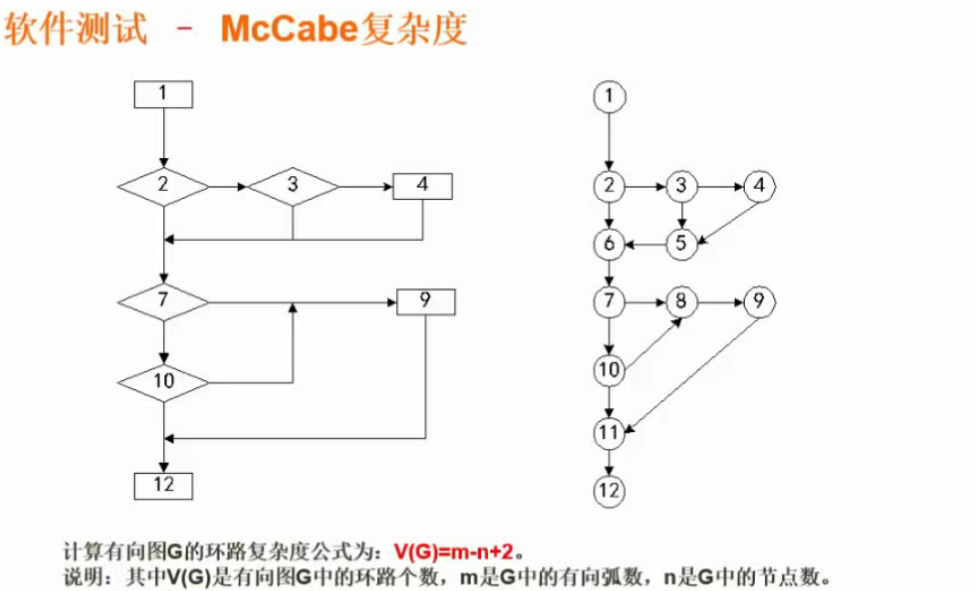


McCall质量模型



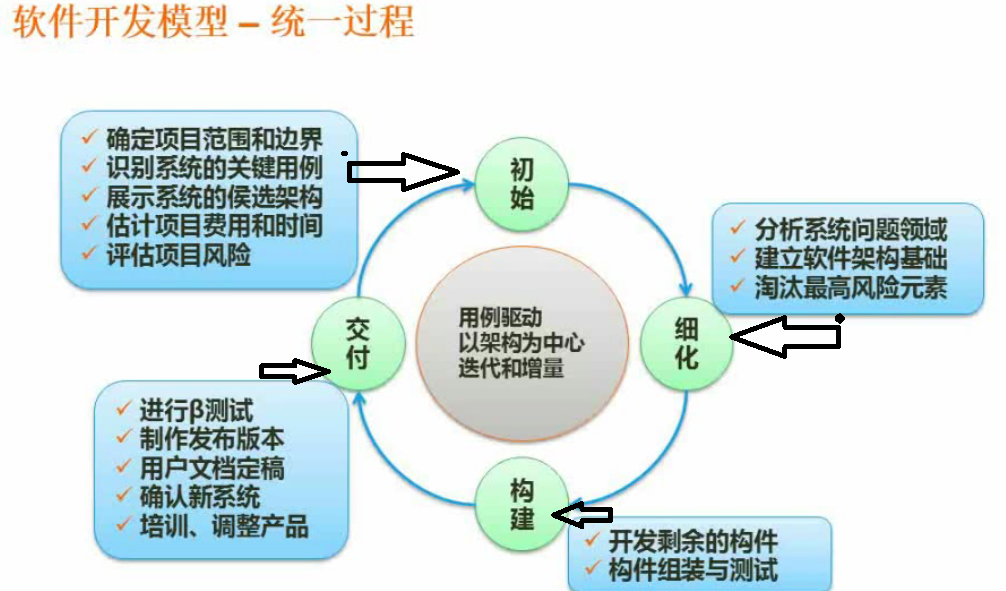
软件复杂性概念及技术（核心）

McCabe复杂度



### 软件改进过程

统一过程（up）



软件能力成熟度模型

CMM/CMMI将软件过程的成熟度分为5个等级,以下是5个等级的基本特征：

　　(1)初始级(initial)。工作无序，项目进行过程中常放弃当初的计划。管理无章法，缺乏健全的管理制度。开发项目成效不稳定，项目成功主要依靠项目负责人的经验和能力，他一但离去，工作秩序面目全非。

　　(2)可重复级(Repeatable)。管理制度化，建立了基本的管理制度和规程，管理工作有章可循。 初步实现标准化，开发工作比较好地按标准实施。 变更依法进行，做到基线化，稳定可跟踪，新项目的计划和管理基于过去的实践经验，具有重复以前成功项目的环境和条件。

　　(3)已定义级(Defined)。开发过程，包括技术工作和管理工作，均已实现标准化、文档化。建立了完善的培训制度和专家评审制度，全部技术活动和管理活动均可控制，对项目进行中的过程、岗位和职责均有共同的理解 。

　　(4)已管理级(Managed)。产品和过程已建立了定量的质量目标。开发活动中的生产率和质量是可量度的。已建立过程数据库。已实现项目产品和过程的控制。可预测过程和产品质量趋势，如预测偏差，实现及时纠正。

　　(5)优化级(Optimizing)。可集中精力改进过程，采用新技术、新方法。拥有防止出现缺陷、识别薄弱环节以及加以改进的手段。可取得过程有效性的统计数据，并可据进行分析，从而得出最佳方法。

### 软件项目管理

软件文档（文件）

软件文档：开发文档 管理文档 用户文档

开发文档《功能要求》《投标方案》《需求分析》《技术分析》《系统分析》《数据库分析》《功能函数文档》《界面文档》《编译手册》《QA文档》《项目总结》

管理文档《产品简介》《产品演示》《疑问解答》《功能介绍》《技术白皮书》《评测报告》

用户文档《安装手册》《使用手册》《维护手册》《用户报告》《销售培训》

甘特图

甘特图不能清晰的描述各任务的依赖关系。

PERT图求关键路径（核心）

1.关键路径：从开始到结束得所有路径中，所话时间最长的一条为关键路径。

注意：在关键路径上的任务的松弛时间为0

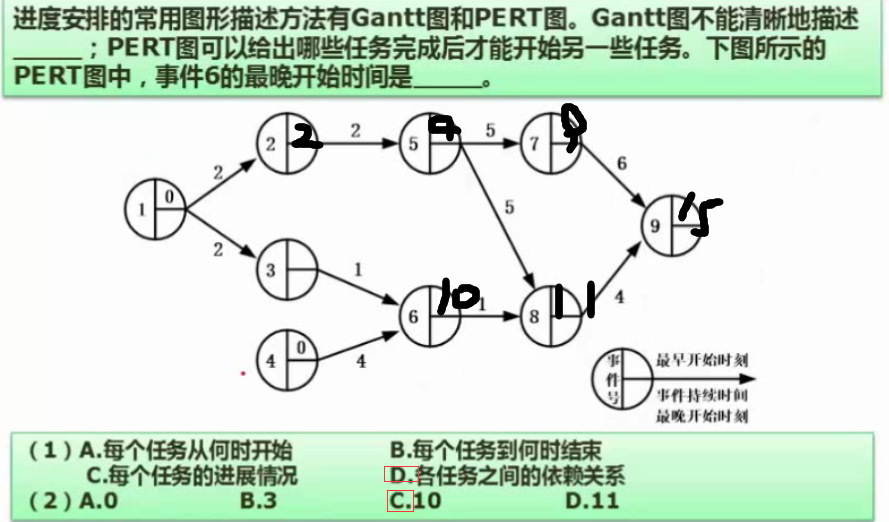
 最早开始时间：在关键路径上，从开始到该任务的最早执行的时间

 最晚开始时间：关键路径的总时间-反向得出该任务的时间

2.松弛时间（最多延迟执行的时间）

 第一种求法：最晚开始时间-最早开始时间

 第二种求法：关键路径的总时间-包含该任务的最长路径花的时间



风险管理（核心）

风险管理：风险识别，风险预测，风险评估，风险控制。

风险识别通过建立风险条目检查表，系统化的确定对项目计划的威胁。该检查表集中一些常见的风险来供识别风险。

风险预测（风险估算），风险发生的可能性及概率，以及发生风险后的后果。

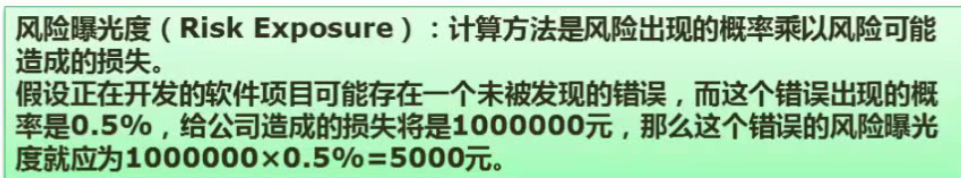
风险评估是定义风险参考水平。

风险控制风险避免，风险监控和风险管理及意外时间计划。

风险预测估算

1. 建立一个尺度，以反映风险发生的可能性。
2. 描述风险后果。
3. 估算风险对项目及产品的影响。
4. 标注风险预测得到整体精确度，以免误解。

风险预测估算相关计算



风险曝光度大的应该更加重视，例如10万 80%则损失8万 100万 0.5% 则损失5000

所以这里更应该注意10万的风险

软件危机

（1）软件本身的复杂性 软件本身的特点 质量不佳，成本高 维护性差等

（2）软件开发方法和技术 缺乏好的开发技术和手段；开发效率低。软件需求不准，软件生产进度无法控制。