**软件工程概述**

软件=程序+文档+数据

软件是一种产品，软件是一种交付产品的工具

软件特性：相对于硬件，软件是自定义构建，复杂的，可修改，升级的，退化的

* **软件是逻辑的、知识性的智力产品，是对物理世界的一种抽象**
  + **Developed/engineered， Custom built，Complex， Deteriorates**🡪
  + **软件具创造性，非标准化，需求可能会不断变化，软件技术日新月异，易修改**🡪
* **软件危机的通用概念：**
  + **软件的可靠性没有保障、维护费用不断上升、进度无法预测、成本增长无法控制、程序员无限度增加等，形成软件开发局面失控的状态**
* **第一次软件危机（ 1945 to 1968）**
  + **危机表现**  
    • 软件生产效率低下**,**软件成本高: 软件开发成本和进度的估计常常  
    很不准确。  
    • 软件产品质量差: 用户对“已完成的”软件系统不满意的现象经  
    常发生。  
    • 软件产品难以**(**不能**)**维护: 软件通常没有适当的文档资料
* **第二次软件危机（ 1968 to 2001 ）**
  + **危机表现**
    - The project will produce the wrong product[有错的产品].
    - The project will produce a product of inferior quality[低质量的产品].
    - The project will be late[延迟].
    - We’ll have to work 80 hour weeks[每周工作80小时].
    - We’ll have to break commitments[违约].
    - We won’t be having fun[没有乐趣]
* **第三次软件危机（2001年至今）**
  + **危机表现**
    - 变更/演化/重用（质量、效率、成本）
    - 知识产权问题、安全问题
  + **危机根源**
    - 大量遗产系统的挖掘和重用
    - 大量开源软件/代码的使用
    - 普遍存在的动态/不确定需求

软件工程 **软件工程是计算机科学、工程和管理三个学科的综合**

软件生命周期

* **Coarse granularity**
  + **1 Analysis**
  + **2 Design**
  + **3 Implementation**
  + **4 Maintenance 维护**
* **Refined granularity**
  + **1 Application Requirements determination**
  + **2 Software Requirements specification**
  + **3 Architectural design**
  + **4 Detailed design**
  + **5 Implementation**
  + **6 Integration**
  + **7 Maintenance**

软件过程 **生产一个最终满足需求且达到工程目标的软件产品所需要的步骤**

软件过程模型

**Umbrella Activities 管理**

* **Software project management**
* **Formal technical reviews**
* **Software quality assurance**
* **Software configuration management**
* **Work product preparation and production**
* **Reusability management**
* **Measurement**
* **Risk management**

**Framework Activities 工程**

* **Communication**
* **Planning**
* **Modeling**
  + **Analysis of requirements**
  + **Design**
* **Construction**
  + **Code generation**
  + **Testing**
  + **Deployment**
  + **瀑布模型(waterfall model)**
    - * 1. **传统的生命周期模型**
        2. **最早由Royce 在1970年提出**
        3. **典型瀑布模型具有顺序性和依赖性**

**瀑布模型的特征**

* + - * 1. **从上一项活动中接受该项活动的工作成果(工作产品)，作为输入**
        2. **利用这一输入实施该项活动应完成的内容**
        3. **给出该项活动的工作成果，作为输出传给下一项活动**
        4. **对该项活动实施的工作进行评审。若其工作得到确认，则继续下一项活动**
      1. **瀑布模型的优点：**
         1. **强调开发的阶段性；**
         2. **强调早期计划及需求调查；**
         3. **强调产品测试。**
      2. **瀑布模型的缺点：**
         1. **瀑布模型过于依赖早期进行的唯一一次需求调查，不能适应需求的变化**
         2. **瀑布模型是单一流程，开发中的经验教训不能反馈应用于本产品的过程。**
  + **原型模型(prototyping model)**
  1. **在获取一组基本的需求定义后，利用高级软件工具的可开发环境，快速地建立一个目标系统的最初版本-原型，并把它交给用户试用、补充和修改，再进行新的版本开发。反复进行这个过程，直到得出系统的“精确解”，即用户满意为止。**
  2. **经过这样一个反复补充和修改的过程，应用系统的“最初版本”就逐步演变为系统的“最终版本”。**
  3. **原型Prototype：**
     + **一个真实的可执行模型，它实现了系统的若干基本功能。**
     + **注意：原型可用来帮助分析和设计工作，但不是一个软件产品。**
* **原型特征：**
  + **软件原型是软件的最初版本，以最少的费用、最短的时间开发出的、反映最后软件的主要特征的系统。它具有以下特征：**
  + **(1)它是一个可实际运行的系统。**
  + **(2)它没有固定的生存期。它可能被扔掉，或者作为最终产品的一部分。**
  + **(3)从需求分析到最终产品都可作原型，即可为不同目标作原型。**
  + **(4)它必须快速、廉价。**
  + **(5)它是迭代过程的集成部分，即每次经用户评价后修改、运行，不断重复双方认可。**
* **原型法的优点：**
  + **一开始就能弄清楚所有的产品需求，或者至少可以帮助导引出高质量的产品要求。**
  + **可在项目早期就获取项目的相关数据，尽早进行风险管理和配置管理。**
  + **心理上，开发人员早日见到产品的雏型，是一种鼓舞。**
  + **使用户可以在新的一批功能开发测试后，立即参加验证，以便提供非常有价值的反馈。**
  + **可使销售工作有可能提前进行，因为可以在产品开发的中后期取得包含了主要功能的产品原型去向客户作展示和试用。**
* **原型法的缺点：**
  + **如果缺乏严格的过程管理的话，这个生命周期模型很可能退化为一种原始的无计划的“试－错－改”的循环模式。**
  + **心理上，可能产生一种影响尽最大努力的想法，认为虽然不能完成全部功能，但还是造出了一个有部分功能的产品。**
  + **如果不加控制地让用户接触开发中尚未测试稳定的功能，可能对开发人员及用户都产生负面的影响。**
  + **增量模型(incremental model)**
    - **增量模型又称产品改进模型**
    - **从给定需求开始，通过构造一系列中间版本来实施开发活动，依次类推，直到系统完成。**
    - **每一个中间版本都是需求分析、设计、编码和测试的过程。**
    - **某些中间版本的开发可以并行进行。**
  1. **增量模型特征**
     1. **融合了线性顺序模型的基本成分和原型的迭代特征。**
     2. **是随着日程时间的进展而交错的线性序列。**
     3. **与原型不一样的地方是强调每个增量均发布一个可操作产品。**
     4. **最典型的应用是同一个产品的不同项目(合同、用户)版本**
  2. **增量模型特别适用于：**
     1. **需求经常变化的软件开发**
     2. **市场急需而开发人员和资金不能在设定的市场期限之前实现一个完善的产品的软件开发**
  3. **增量模型能有计划地管理技术风险，如早期增量版本中避免采用尚未成熟的技术**
  + **螺旋模型(spiral model)**

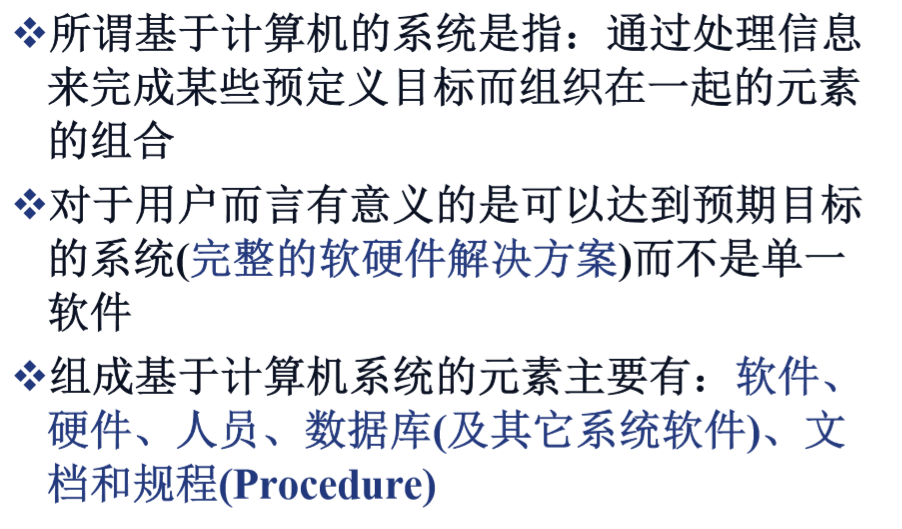
1. **是瀑布模型和演化模型的结合，并增加了风险分析**
2. **螺旋模型沿着螺线旋转，在四个象限上分别表达四个方面的活动，即：**
   1. **制定计划：确定软件目标，选定实施方案，弄清项目开发的限制条件**
   2. **风险分析：评价所选的方案，识别风险，消除风险**
   3. **工程实施：实施软件开发，验证工作产品**
   4. **客户评估：评价开发工作，提出修正建议**

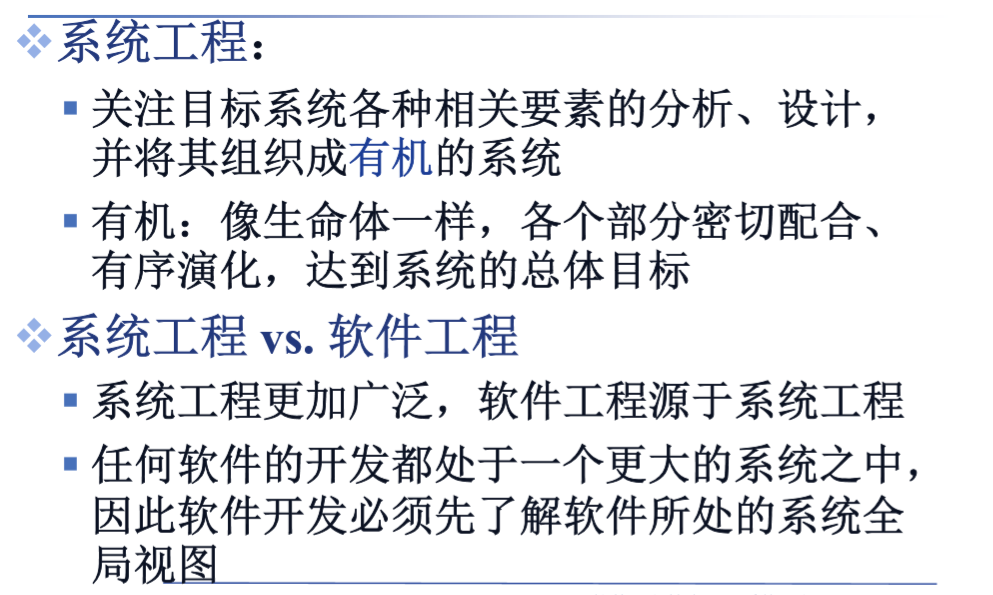
* **优点**： **a.强调严格的全过程风险管理。  
  b.强调各开发阶段的质量。  
  c.提供机会检讨项目是否有价值继续下去。**
* **缺点：  
  必须引入非常严格的风险识别，风险分析和风险控制，这对风险管理的技能水平提出了很高的要求。也需要人员，资金和时间的大量投入。** 
  + **喷泉模型(water fountain model)**
    1. **喷泉模型认为软件生命周期的各个阶段是相互重叠和多次反复的。**
    2. **也是一种线性开发模型, 与瀑布模型类似，只是从串行改并行。**
    3. **主要用于面向对象方法中，面向对象的分析和设计重叠，交叉、并行进行。**
  + **RUP模型(Rational Unified Process)**
    1. **一种利用UML进行面向对象软件工程的框架。**
    2. **特点：**
    3. **用例驱动**
    4. **以架构为核心**
    5. **迭代并且增量**
    6. **RUP的时间轴被分解为四个顺序的阶段：**
       - **初始阶段(Inception)为系统建立业务案例并确定项目的边界。**
       - **细化阶段(Elaboration)分析问题领域，建立健全的体系结构基础，编制项目计划，淘汰项目中最高风险的元素。**
       - **构造阶段(Construction)所有剩余的构件和应用程序功能被开发并集成为产品，所有的功能被详细测试。**
       - **交付阶段(Transition)确保软件对最终用户是可用的。**
* **核心过程工作流**
  + **(1)业务建模工作流为组织开发一个构想，并基于这个构想在业务用例模型和业务对象模型中定义组织的过程，角色和责任。**
  + **(2)需求工作流的目标是描述系统应该做什么，并使开发人员和用户就这一描述达成共识。**
  + **(3)分析和设计工作流将需求转化成未来系统的设计，为系统开发一个健壮的结构并调整设计使其与实现环境相匹配，优化其性能。**
  + **(4)实现工作流的目的包括以层次化的子系统形式定义代码的组织结构；以组件的形式(源文件、二进制文件、可执行文件)实现类和对象；将开发出的组件作为单元进行测试以及集成由单个开发者(或小组)所产生的结果，使其成为可执行的系统。**
  + **(5)测试工作流要验证对象间的交互作用，验证软件中所有组件的正确集成，检验所有的需求已被正确的实现,识别、提出缺陷并确认缺陷在软件部署之前被处理。**
  + **(6)部署工作流的目的是成功的生成版本并将软件分发给最终用户。**
* **核心支持工作流**
  + **(1)配置和变更管理工作流描绘了如何在多个成员组成的项目中控制大量的产物，管理演化系统中的多个变体，跟踪软件创建过程中的版本。**
  + **(2)软件项目管理工作流平衡各种可能产生冲突的目标，管理风险，克服各种约束并成功交付使用户满意的产品。**
  + **(3)环境工作流的目的是向软件开发组织提供软件开发环境，包括过程和工具。**
* **RUP的优点：**
  + **提高了团队生产力，在迭代的开发过程、需求管理、基于组件的体系结构、可视化软件建模、验证软件质量及控制软件变更等方面，针对所有关键的开发活动为每个开发成员提供了必要的准则、模板和工具指导，并确保全体成员共享相同的知识基础。它建立了简洁和清晰的过程结构，为开发过程提供较大的通用性。**
* **RUP的缺点**
  + **RUP在理论上，是比较理想的，但在实际应用上，还需要更多的工具的支持和普及推广工作。**
  + **敏捷开发宣言**
    - * **个体和交互胜过过程和工具**
      * **可工作软件胜过宽泛的文档**
      * **客户合作胜过合同谈判**
      * **响应变化胜过遵循计划**
  + **极限编程(XP，Extremely Programming)**
    1. **The most widely used agile process, originally proposed by Kent Beck，in 1996**
    2. **XP使用面向对象方法作为推荐的开发范型。**
    3. **XP包含了四个框架活动的规则和实践：**
       - **策划**
       - **设计**
       - **编码**
       - **测试**
  + **XP Coding**
    - * **Recommends the construction of a unit test for a story *before* coding commences (测试驱动编程)**
      * **Encourages pair programming(结对编程)**
  + **XP Testing**
    - * **All unit tests are executed daily(强调Daily Building，冒烟测试)**
      * **Acceptance tests are defined by the customer and executed to assess customer visible functionality(强调验收测试，回归测试)**
  + **极限编程适合**
    - * **小团队 (2-10 programmers)**
      * **“高风险”**
      * **快速变化或不稳定的需求**
      * **强调可测试性**

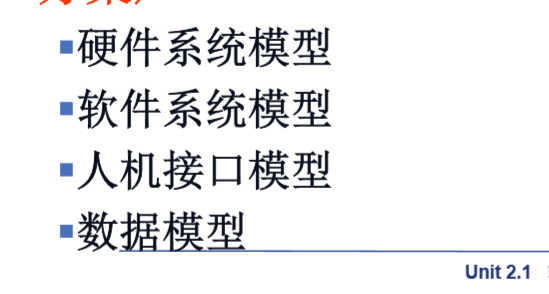
**软件工程技术**

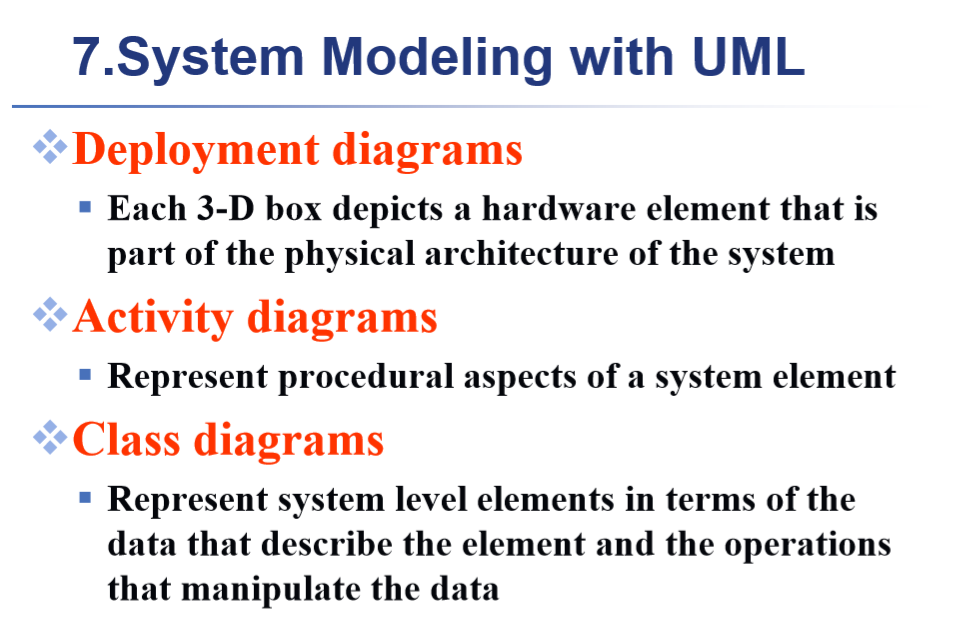
1. **系统工程**

**系统定义(ISO)：  一组或一系列相关的元素[人、产品(硬件和 软件)及过程（设备、装备、材料和规程）]， 其行为满足运转需要并且为产品生存周期的 维持提供支撑。**









1. 需求工程

**什么是需求？**

* 1. **软件开发的需求：需求来源于用户的一些“需要”，这些“需要”被分析、确认后形成完整的文档，该文档详细地说明了产品“必须或应当”做什么。**
  2. **功能性需求**
     1. **系统需要提供的服务或功能：**
        1. **如图书检索**
     2. **系统对特定输入的处理方式：**
        1. **如对非法输入的提示**
     3. **系统在特定环境下的行为：**
        1. **如长时间无操作时自动退出**
  3. **非功能性需求**
     1. **对系统功能或服务附加的质量约束，例如响应时间、容错性、安全性等——客户所关心的(外部质量)**
     2. **从系统开发和维护角度出发的质量属性，例如可理解性、可扩展性、可配置性等——软件开发或维护者所关心的(内部质量、软件所特有)**
  4. **需求分析过程包括：**
     1. **确定对系统的综合要求**
     2. **分析系统的数据要求**
     3. **抽象出并确立目标系统的逻辑模型**
     4. **编写需求规格说明书**
* **需求获取方法**
  + **访谈与调查（会议，问卷，原型）**
  + **需求获取步骤**
    - **1）确定stakeholders**
    - **2）从不同视角，不同角度收集信息**
    - **3）stakeholders协同合作收集需求**
      * **会议，问卷，原型**
    - **4）创建用户场景，构建用况**
* **2.需求分析方法**
  + **数据流图，实体关系图，数据字典，控制流图**
  + **需求分析的步骤**
    - **1）进行系统分析，确定对系统的综合要求**
    - **2）分析系统的功能要求**
    - **3）分析系统的数据要求**
    - **4）抽象出并确立目标系统的逻辑模型**
* **3.需求处理方法**
  + **需求规格说明书模板**
* **4.需求确认方法**
  + **评审**

1. 设计工程
   1. **系统设计阶段，我们将解决“怎么做”**
      1. **设计模型提供了软件的体系结构、数据结构、接口和构件的细节**
      2. **设计模型可以直接转化为系统实现**
   * **软件体系结构设计的任务**
   * **定义系统的主要结构元素及相互的关系**
     1. **结构化方法： 从数据流图出发对数据进行分析，得出软件的层次化的模块结构图**
     2. **面向对象方法：从分析模型出发划分子系统，在考虑通信、并发、 部署、复用等问题的基础上，建立系统层次结构**

* **体系结构设计**
  + **面向结构的设计：划分系统层次结构，确定功能模块，确定模块调用关系及接口。**
  + **面向对象的设计：对子系统进行设计，定义出若干个一致的类与对象、关系、行为、功能的集合。**
* **1、抽象（abstraction）**
  + - **抽象：就是抽出事物的本质特性而暂时不考虑它们的细节。**
    - **是控制复杂性的基本策略之一**
    - **抽象过程：从特殊到一般的过程，上层概念是下层概念的抽象，下层概念是上层概念的精化和细化**
* **2、体系结构（architecture ）** 
  + - **软件体系结构关注系统的一个或多个结构，包含软件部件、部件对外可见的属性以及部件间的关系**
    - **体系结构的作用**
      * **方便利益相关人员的交流**
      * **有利于系统设计的前期决策**
      * **可传递、易于理解的系统级抽象**
* **3、模式（patterns）**
* **每个模式都描述了一个在我们所处环境内反复发生的问题，描述了该问题的核心解决方案。**
* **模式的定义便于复用。**
* **模式的类型**
  + **结构模式 （Architecture Pattern）**
  + **设计模式 （Design Pattern）**
  + **编码模式 （Idiom）**
* **4、模块化（modularity）**

**按照设计原则将系统划分为若干个较小的模块（构件）**

**相互独立但又相互关联**

**实际上是系统分解和抽象的过程**

**模块是相对独立的程序体**

**独立命名的，并且可以通过名字来访问**

**例如：过程、函数、子程序、宏等**

* **5、信息隐藏（information hiding）**
* **每个模块都尽量对其他模块隐藏自己的内部实现细节**
  + **模块内部的数据和过程不允许其它不需要这些信息的模块使用**
  + **定义和实施对模块的过程细节和局部数据结构的存取限制**
* **信息隐藏是实现抽象/模块化机制的基本支撑**
* **6、功能独立（functional independence）**
* **功能独立（模块独立）是模块化的根本要求**
  + **模块完成独立的功能：明确可辨识**
  + **高内聚：内部结构紧密**
  + **低耦合：模块间关联和依赖程度尽可能小，与其他模块的接口简单**
  + **符合信息隐蔽和信息局部化原则**
* **7、精化（refinement）**
* **把问题的求解过程分解成若干步骤或阶段，每步都比上步更精化，更接近问题的解法**
* **常与分层抽象的思想相结合**
  + **抽象使得设计者能够描述过程和数据而忽略低层的细节**
  + **求精有助于设计者在设计过程中揭示低层的细节**
  + **高层抽象将在下层不断精化，最终得到软件实现**
* **8、重构（refactoring）**
  + - **"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure.”**
    - **重构是一个改变软件系统的过程，在不改变代码[设计]的外部行为的同时改进其内部结构**

体系结构设计

* **为什么要进行软件体系结构设计？**
  + **Architecture is a representation of a system that enables the software engineer to:**
    - * **analyze the effectiveness of the design in meeting its stated requirements,**
      * **consider architectural alternatives at a stage when making design changes is still relatively easy, and**
      * **reduce the risks associated with the construction of the software.**
* **3. 体系结构设计的重要性**
  + **软件系统设计是开发者之间的分工和合作和基础，它能够方便利益相关人员的交流**
  + **有利于系统设计的前期决策 ，设计方案是决定系统质量的主要因素**
  + **体系结构构建了一个可传递、易于理解的系统级抽象**
* **. 软件体系结构风格**
* **1）定义：**
  + **模式：每个模式是一个规则，表达了特定环境、问题和解决方式(solution)之间的关系。**
  + **体系结构风格：定义了一系列系统的结构组织的模式，它是对一类具有相似结构的系统体系结构的抽象**
* **1）定义**
  + **同一个问题，可以有不同的解决问题的模式，但我们根据经验，通常会采用特定的模式，这就是风格**
  + **软件风格是对软件构成带有整体性、普遍性、一般性的结构和结构关系的定义方法**
  + **因此，软件风格是一种特定的基本结构。**
* **每种风格描述一种系统类别，其中包括下列元素：**
  + **一些实现系统所需的功能的构件**
  + **连接各个构件，负责构件间通信和协作的连接器**
  + **定义构件之间怎样整合集成的系统约束**
  + **使设计者能够理解整个系统属性并分析已知属性的语义模型**
* **典型的体系结构风格**
  + **以数据为中心的体系结构**
  + **数据流体系结构**
  + **调用返回体系结构**
  + **面向对象体系结构**
  + **层次体系结构**

构建建模

* **什么是构件？**
  + **OO View – A component is a set of collaborating classes.**
* **构件级设计**
  + **是对体系结构设计结果的进一步细化**
  + **是对目标的精确描述**
  + **面向编码实现，可以直接翻译成计算机代码**
* **构件级设计的设计原则** 
  + **开关原则(Open-Closed Principle)**
  + **替换原则(Substitutability)**
  + **依赖倒置原则(Dependency Inversion)**
  + **接口分离原则(Interface Segregation)**
  + **高内聚，低耦合**

1. 测试

**测试策略根据软件的开发阶段来划分**

* 1. **单元测试是针对程序中的模块或构件，主要揭露编码阶段产生的错误。**
  2. **集成测试针对集成的软件系统，主要揭露设计阶段产生的错误。**
  3. **系统测试：对于基于计算机系统中的软件，还需将它集成到基于计算机系统中进行测试，以揭露不符合系统工程中对软件要求的错误。**
  4. **确认测试是根据需求规约对软件进行确认，主要揭露不符合需求规约的错误。**
* **单元测试侧重于对软件设计的最小单元的验证工作。**
* **2)单元测试的目的**
  + **在开发环境中检查单元程序模块内部的逻辑、算法和数据处理结果的正确性等。**
* **3)单元测试的内容主要是：**
  + **算法逻辑、数据定义的理解和使用、接口、各种控制路径、边界条件、错误处理等。**
* **集成测试是构造软件体系结构的系统化技术，同时也是进行一些旨在发现与接口相关的错误的测试。**
* **为什么要进行集成测试？**
  + **数据可能在通过接口时丢失；**
  + **一个模块可能对另一个模块产生非故意的、有害的影响（即副作用）；**
  + **当子功能被组合起来时，可能不能达到期望的主功能；**
  + **单个模块可以接受的不精确性（如误差），连接起来后可能会扩大到无法接受的程度；**
  + **全局数据结构可能会存在问题。**
* **系统测试：**
  + **对完整集成后的产品和解决方案进行测试，用来评价系统对具体需求规格说明的功能和非功能的符合性的测试**
* **系统测试的目的/作用：**
  + **发现可能难以直接与模块或接口关联的缺陷**
  + **发现产品设计、体系和代码的基础问题（产品级缺陷）**
* **确认测试以软件需求规约为依据，以发现软件与需求不一致的错误。**
* **主要检查：**
  + **软件是否实现了规约规定的全部功能要求**
  + **文档资料是否完整、正确、合理**
  + **其他的需求，如可移植性、可维护性、兼容性、错误恢复能力等是否满足**

测试技术

* **黑盒方法和白盒方法** 
  + **黑盒测试方法（Blake-box Testing）**
    1. **也称为行为测试**
    2. **把程序看作一个不能打开的黑盒子，不考虑程序内部结构和内部特性**
    3. **考察数据的输入、条件限制和数据输出，以完成测试 。**
  + **白盒测试方法（White-box Testing）**
    1. **也称结构测试或逻辑驱动测试**
    2. **基于内部逻辑结构，针对程序语句、路径、变量状态等来进行测试**
    3. **检验程序中的各个分支条件是否得到满足、每条执行路径是否按预定要求正确的工作。**
* **手工测试 和 自动化测试**
  + **静态分析通常只能进行手工测试，虽然也有辅助工具帮助，但仍以人为主体。**
  + **动态测试往往可以部分地或者全部自动化进行。**
    1. **提供有效的测试用例集，通过测试平台对被测程序进行测试，跟踪并分析测试结果。**
  + **自动化测试的特点**
    - * **自动运行的速度快**
      * **测试结果准确**
      * **例如搜索用时是0.33秒或0.24秒，系统都会发现问题，不会忽视任何差异**
      * **高复用性**
      * **一旦完成所用的测试脚本，可以一劳永逸运行很多遍**
      * **永不疲劳**
      * **可靠**
      * **独特的能力**
  + **两者互相补充**
    1. **在系统功能逻辑测试、验收测试、适用性测试、涉及交互性测试时，多采用手工测试方法；**
    2. **单元测试、集成测试、系统负载或性能、可靠性测试等比较适合采用自动化；**
    3. **由于工具本身缺乏想象力和创造性，自动测试只能发现15%的缺陷，而手工测试可以发现85%的缺陷**

**软件工程管理**

**4P**

* + **人员（People）**
    - 1. **软件开发是人的智力密集的劳动**
      2. **人员是软件工程项目的基本要素和关键因素**
      3. **项目管理必须把人员组织起来以有效地完成软件工作**
  + **产品（Product）**
    - 1. **制定项目计划之前，首先确定产品的范围**
      2. **范围包括：目标和范围、可选的解决方案、技术或管理的约束等**
      3. **其目标和范围必须获得软件开发者和客户的一致认可**
  + **过程（Process）**

1. **在软件过程框架下制定软件开发的综合计划**
2. **根据项目的特征和项目团队的需求选择框架活动**
   * **项目（Project）**
     + 1. **采用科学的方法及工具对项目基本内容进行管理**
       2. **实施有计划，可控制的软件项目**

**W5HH**

* + **Why is the system being developed?**
  + **What will be done?**
  + **When will it be accomplished?**
  + **Who is responsible?**
  + **Where are they organizationally located?**
  + **How will the job be done technically and managerially?**
  + **How much of each resource (e.g., people, software, tools, database) will be needed?**

**软件度量**

* + **度量的作用是为了有效地采用定量的方式来进行管理**
  + **管理人员利用度量来了解软件工程过程的执行情况和产品质量**
  + **需要考虑：**
    - 1. **合适的度量是什么**
      2. **所收集的数据如何使用**
      3. **用于比较个人、过程或产品的度量是否合理**
  + **面向规模的度量（Size Oriented Metrics）**

**软件规模通常是指软件的大小(size)，一般用代码行度量**

* + - * **优点：方便、直观**
      * **缺点：很大程度上取决于程序设计语言以及软件设计的质量**

**WBS**

* **A work breakdown structure （WBS） is a deliverable-oriented grouping of the work involved in a project that defines the total scope of the project**
* **WBS的分解：**
  + **WBS is usually organized around**
    - * **project products (tools, tangible outputs, .)**
      * **project phases (concepts, steps,..)**
      * **project management process groups (initializing, developing, executing, closing)**
  + **因此，其分解方式为**
    - * **产品分解：将一个复杂问题分解成若干个小问题**
      * **过程分解：根据软件开发过程中的活动(分析、设计、编码、测试等)进行分解**
  + **工作包（Work Packages）**
    1. **WBS结构的叶子节点，即任务分解的最小单位，不可再划分的子任务。**
  + **WBS的作用**
    1. **It provides the basis for**
       - **Planning and managing project schedules, costs, and changes**
       - **Risk analysis**
       - **Organizational structure**
       - **Measurement**
* **成本估算的常用方法：**
  + **1）基于已经完成的类似项目进行估算，这是一种常用的也是有效的估算方法**
  + **2）基于规模的估算**
    - * **基于LOC的估算**
      * **基于FP的估算**
  + **3）基于经验估算模型的估算**
    - * **COCOMO模型，等**

项目进度管理

**Gantt charts (甘特图)**

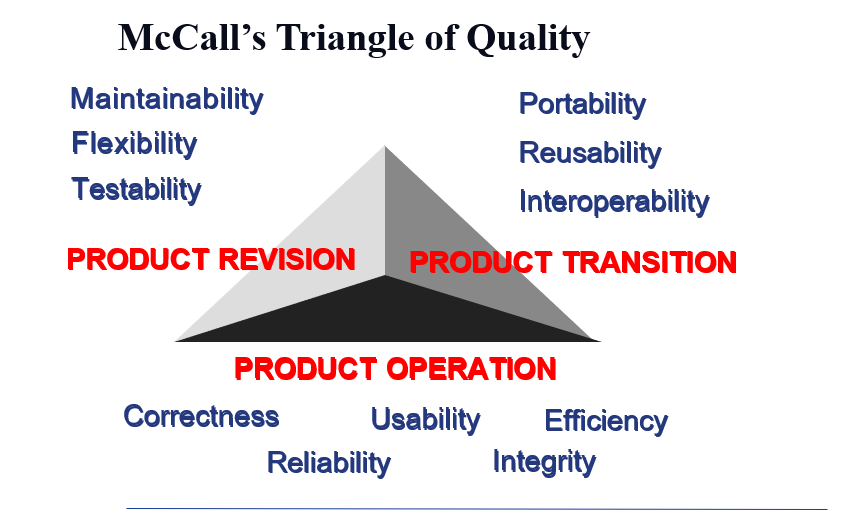
* + **为监控软件项目的进度计划和工作的实际进展情况，表现各项任务之间进度的相互依赖关系**
  + **需要采用图示的方法明确标识：**
    - 1. **各个任务的计划开始时间和完成时间**
      2. **各个任务的完成标志**
      3. **各个任务与参与工作的人数，各个任务与工作量之间的衔接情况**
      4. **完成各个任务所需的物理资源和数据资源**
  + **甘特图(Gantt Chart)也称时间表(Timeline chart)，用来建立项目进度表**
  + **在甘特图中，每项任务的完成以必须交付的文档和通过评审为标准，因此它们往往作为项目里程碑**

风险管理

* **Reactive Risk Management**
  + **project team reacts to risks when they occur**
  + **mitigation—plan for additional resources in anticipation of fire fighting**
  + **fix on failure—resource are found and applied when the risk strikes**
  + **crisis management—failure does not respond to applied resources and project is in jeopardy（危险）**
* **Proactive Risk Management**
  + **formal risk analysis is performed**
  + **organization corrects the root causes of risk**
    - **TQM concepts and statistical SQA**
    - **examining risk sources that lie beyond the bounds of the software**
    - **developing the skill to manage change**
* **主动风险管理的过程**
  + **风险识别**
    - **主要识别下列类型中的已知的和可预测的风险**
      * **Product size：总体规模**
      * **Business impact：业务约束，市场约束**
      * **Customer characteristics：客户素质**
      * **Process definition：过程相关**
      * **Development environment：开发工具**
      * **Technology to be built：复杂性，创新性**
      * **Staff size and experience：人员**
    - **风险因素**
      * **性能风险：产品能满足需求且符合其使用目的的不确定的程度**
      * **成本风险：项目能否被维持的不确定的程度**
      * **支持风险：软件易于维护的不确定的程度**
      * **进度风险：项目进度能被维持且产品能按时交付的不确定的程度**
  + **风险分析**
  + **A.风险预测：**
    - **评价每种风险发生的可能性或概率以及当该风险发生时所导致的后果**
    - **风险预测任务：**
      * **建立一个尺度，以反映风险发生的可能性**
      * **描述风险的后果**
      * **估算风险对项目及产品的影响**
      * **标注风险预测的整体精确度以免产生误解**
  + **B.建立风险表**
    - **第1列列出所有的风险**
    - **第2至4列列出每个风险的种类(项目风险、技术风险、商业风险等)、发生的概率以及所产生的影响**
    - **综合考虑风险发生的概率和风险所产生的影响，对风险表排序**
  + **C.风险评估**
    - **进一步审查风险预测阶段对各种风险预测的精确度，并定义一条中截线**
      * **中截线：风险表中某处的一条水平线**
      * **中截线以上是会进一步关注的高风险**
      * **中截线以下的风险则需要重新评估及二次排序。**
      * **所有在中截线以上的风险都必须进行管理**
      * **风险表的最后一列RMMM为其管理进行指示**
  + **风险规划（RMMM）**
  + **风险分析的主要目的就是辅助项目团队制定处理风险的策略**
  + **有效的风险规划必须考虑：**
    - **风险避免 （Risk Mitigation）**
    - **风险监测 （Risk Monitoring）**
    - **风险管理及应急计划（Risk Management Plan）**
  + **风险监控**
    - **监控可以提供风险指示(是否正在变高或变低)的因素**
    - **监控目的：**
      * **评估所预测的风险是否真的发生了**
      * **保证正确实施了各项缓解步骤**
      * **收集能够用于今后风险分析的信息**

软件质量

McCall质量模型

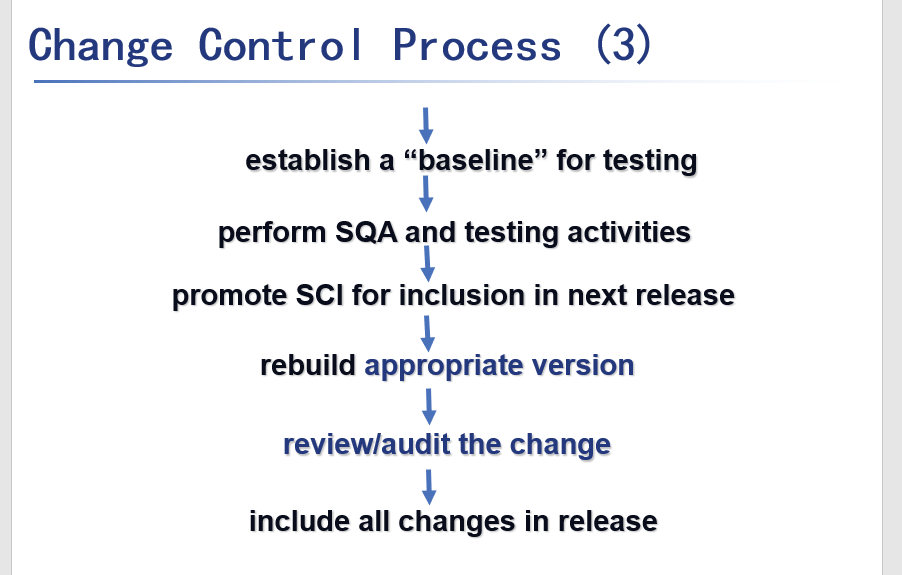
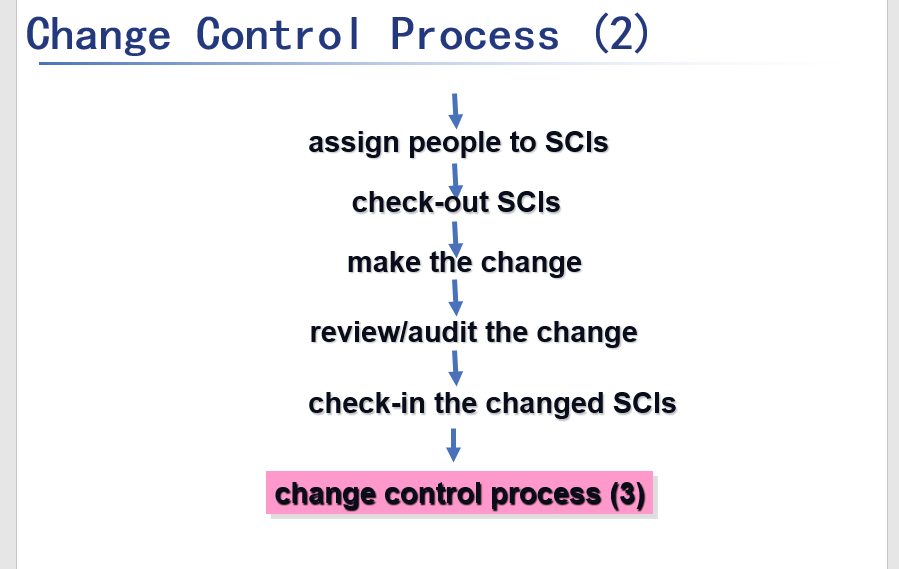
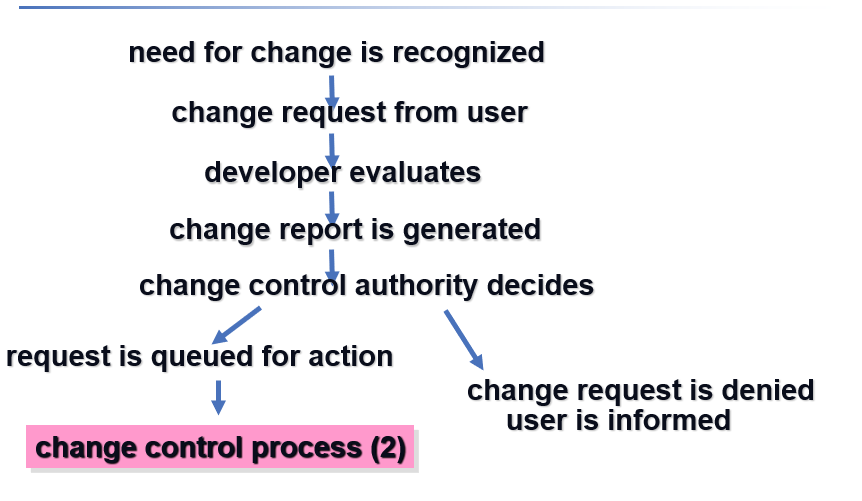


* **质量保证**
  + **质量保证由管理层的审计和报告构成，目标是为管理层提供获知产品质量信息所需的数据，从而获得产品质量是否符合预定目标的认识和信心**
* **软件质量保证活动由两类不同的角色承担**
  + **负责技术工作的软件工程师：通过采用可靠的技术方法和措施、进行正式的技术评审、计划周密的软件测试来考虑质量问题，并完成软件质量保证和质量控制活动**
  + **负责质量保证工作的SQA (Software Quality Assurance)小组：辅助软件工程小组得到高质量的最终产品**
* **SQA小组的活动(CMU SEI)**
  + **为项目准备SQA计划**
  + **参与开发该项目的软件过程描述**
  + **评审各项软件工程活动, 核实其是否符合定义好的软件过程中的相应部分**
  + **审计指定的软件工作产品, 并进行核实**
  + **确保软件工程及工作产品中的偏差已记录在案，并根据预定规程进行处理**
  + **记录所有不符合的部分并报告给高管**
  + **此外，SQA小组还需要协调变化的控制和管理，并帮助收集和分析软件度量信息**
* **评审**
  + **评审是软件质量保证的重要手段**
  + **评审时机**
    - **通常在软件工程过程的每个活动(如需求分析、设计、编码)的后期进行**
  + **主要评审活动**
    - **项目管理评审**
    - **技术评审**
* **技术评审**
  + **任务：评价软件产品或服务，并提供下列证据**
    - **完整性**
    - **符合标准和规范**
    - **其更改是正确地实施的，并且仅仅影响配置管理过程所标明的区域**
    - **它们遵循适用的规程**
    - **它们已为下一个活动做好准备**
* **正式评审和非正式评审**
  + **正式评审(formal reviews)**
    - **通常在软件工程过程的每个活动的后期进行**
    - **采用正式的会议评审方式**
    - **通过正式评审意味着里程碑和基线**
  + **非正式评审(informal reviews)**
    - **由同事参加的即兴聚会**
    - **可采用“同行评审”（pear review）或“走查”(walkthrough)的方式**
* **正式的技术评审过程**
  + **评审会议**
  + **评审记录**
    - **指派专人记录会上提出的所有问题**
    - **会议结束后将其整理成一份“评审问题列表”并存档**
  + **评审报告**
    - **评审结束后形成报告，报告应指明被评审的制品，参加评审的人员，评审中发现的问题以及评审的结论**
    - **“评审问题列表”可作为评审总结报告的附件。**

软件配置

* **软件配置项(Software Configuration item，SCI)**
  + **为配置管理设计的软件工作产品的集合，它在配置管理过程中作为单个实体对待。**
* **相关概念**
  + **4）版本(Version)：与计算机软件配置项的完全编篡或重编篡相关的计算机软件配置项的初始发布或再发布。**
  + **5）发布(Release)：一项配置管理行为，它说明某配置项的一个特定版本已准备好用于特定的目的(例如发布测试产品)**
  + **6）基线(baseline)：业已经过正式审核与同意，可用作下一步开发的基础，并且只有通过正式的修改管理过程方能加以修改的规格说明或产品**

配置管理流程



SCM活动的目标就是为了标识变更、控制变更、确保变更正确实现并向其他有关人员报告变更。从某种角度讲，SCM是一种标识、组织和控制修改的技术，目的是使错误降为最小并最有效地提高生产效率。

过程：配置项、工作空间管理、版本控制、变更控制、状态报告、配置审计