**Chapter1 软件工程介绍-----------------------------------------------------------------------------------------**

**\*软件**：软件是指令的集合（计算机程序）；数据结构：文档。定义：①软件是产品：表达了由计算机硬件或可通过局部硬件访问的计算机网络所体现的计算潜能；信息变换器—产生，管理，获取，修改，显示或传送信息②软件也是产品交付使用的载体：是计算机的控制(操作系统)的基础，信息通信(网络)的基础呼喝其他程序的创建和控制的基础(软件工程和环境)

**\*软件特性**：①被开发或设计的，而非制造的②不会磨损，但会退化③大多是用户定制，而非基于构件的组装④复杂的

**\*软件危机**：软件的可靠性没有保证，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，造成软件开发局面失控的状态

**Chapter2 软件过程综述-----------------------------------------------------------------------------------------**

**\*软件工程**：①将系统化的，规范的，可量化的方法应用于软件的开发、运行和维护，即将工程化方法应用于软件②在①中所述方法的研究

**\*软件过程**：①软件开发中所遵循的路线图，可以定义为一系列模式的组合，这些模式定义了一系列的软件开发中所需的活动、动作、工作任务、工作产品及其相关的行为。②过程定义一个框架，为有效教父软件工程技术，这个框架必须建立。软件过程构成了软件项目管理控制的基础，并且建立了一个环境以便于技术 方法的采用、工作产品的产生、里程碑的建立、质量的保证、正常变更的正确管理。

**\*软件过程模型—框架活动Framework activities(for 工程)**：①沟通②策划③建模：需求分析、设计④构建：代码开发、测试⑤部署

**\*保护伞活动Umbrella activities(for 管理)**：①软件项目管理②正式技术审查③软件质量保证④软件配置管理⑤工作产品准备和生产⑥重用性管理⑦测度measurement⑧风险管理

**Chapter3 过程模型-----------------------------------------------------------------------------------------------**

**\*软件生命周期**：软件有一个孕育、诞生、成长、成熟、衰亡的生存过程，这个过程为计算机软件的生命周期。包括分析、设计、实现、维护四个过程或者①应用需求确定②软件需求细化③结构设计④详细设计⑤开发⑥集成⑦维护；**作用**：对问题进行分解，从时间角度对软件开发和维护的复杂问题进行分解，为软件人提供一个公共的框架以便软件人的相互交流

**\*软件过程模型**

**\*1.瀑布模型(waterfall)**

**//特点**：①从上一项活动中接受该项活动的活动成果作为输入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。

**//优点**：强调开发的阶段性，强调早期计划及需求调查，强调产品测试

/**/缺点**：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。③造成软件错误的积累和放大效应

**//适用**：清楚了解问题需求，工作以现行方式进行的项目

**\*2.增量模型(incremental)**

**//特点**：①融合先行顺序模型的基本成分和原型的迭代特征②随着日程时间的进展而交错的线性序列③与原型不同，强调每个增量均发布一个可操作产品④典型应用是同一产品的不同项目版本

**//优点**：①有计划的管理技术风险，如早期增量版本中避免采用尚未成熟的技术②若开发前期找不到足够的人员，早期的增量可以由少量人员完成，若核心产品口碑不错，可以成为下个增量投入更多人员

**//缺点**：①至始至终开发者和客户纠缠在一起，知道完成版本出来②尽管其灵活性能适应需求变化，但也容易退化成边做边改模型，使软件过程控制缺乏整体性

**//适用**：需求经常变化的软件开发，市场急需而开发人员和资金不能再设定的市场期限之前实现一个完善的产品的软件开发

**\*3.原型模型(prototyping)**

**//特点**：快速建立原型的目的是获取需求，原型可用来帮助分析和设计工作

**//优点**：①一开始就能弄清楚所有的产品需求，或至少可以帮助导引出高质量的产品要求②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③使用户可在功能开发测试后，立即参加验证提供有价值的反馈④可使销售工作提前进行⑤心理对开发人员鼓舞

**//缺点**：①若缺乏严格的过程管理，生命周期模型可能退化为无计划的“试-错-改”循环②心理上，产生尽最大努力的想法③不加控制让用户接触未稳定功能，对开发人员和用户不好

**//适用**：需求模糊的项目，质量不能保证，是为定义需求而服务

**\*4.螺旋模型(spiral)**

**//特点**：结合了原型的迭代性质和瀑布模型的系统性和可控性特点，在每个演进过程(四个：制定计划、风险分析、工程实施、客户评估)要**标记里程碑**，增加了风险分析

**//优点**：①强调严格的全过程风险管理②强调各开发阶段的质量③提供机会检讨项目是否有价值继续下去

**//缺点**：①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高要求②需要人员、资金和时间的大量投入

**//适用**：大型开发系统和软件

**\*5.喷泉模型(fountain)**

**//特点**：①是软件生命周期的各个阶段是相互重叠和多次反复的，各阶段没有明显的界线②是一种线性开发模型，与瀑布模型类似，只是从串行改并行

**//优点**：提高软件项目开发效率，节省开发时间

**//缺点**：①在各个开发阶段是重叠的，因此在开发过程中需要大量的开发人员，不利于项目的管理②要求严格管理文档，使得审核的难度加大，尤其是面对可能随时加入各种信息、需求与资料的情况

**//适用**：用于面向对象方法，面向对象的分析和设计重叠，交叉、并行进行

**\*6.统一过程(UP)**

**//特点**：用例驱动，以架构为核心，迭代并且增量，一种利用UML进行面向对象软件工程的框架，是一种增量模型。四个阶段：初始inception，细化elaboration，构造construction，交付transition

**//优点**：①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。②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。

**//缺点**：在实际应用上，需要更多的工具的支持和普及推广工作

/**/适用**：现代大型面向对象的工程

**Chapter4 敏捷视角下的过程----------------------------------------------------------------------------------**

**\*敏捷开发：**宣言：①个体和交互胜过过程和工具②可工作软件胜过宽泛的文档③客户合作胜过合同谈判④响应变化胜过遵循计划

**\*XP极限编程**：应用最广泛的敏捷过程，包含四个框架活动的规则实践：策划、设计、编码、测试。XP使用面向对象方法作为推荐的开发泛型。//适用：小团队，高风险，快速变化或不稳定的需求，强调可测试性

**Chapter6 系统工程------------------------------------------------------------------------------------------------**

\***系统工程中的概念**：①系统定义：一组或一系列相关的元素，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。②基于计算机的系统：通过处理信息来完成某些预定义目标而组织在一起的元素的组合③元素：软件、硬件、人员、数据库、文档、规程④系统工程：关注目标系统各种相关要素的分析设计，并将其组织成有机的系统

**\*系统建模**：①建立模型：（1）定义在多考虑视图中满足需要的过程（2）描述过程行为和

该行为所依据的假设（3）明确定义模型在外在和内在输入（4）描述有助于工程师理解视图

的全部联系（包括输出）②制约因素：假设、简化、限制、约束、客户偏好③本质上倾向

于分级分层

**\*系统模型分类：**①Hatley-Prabhai建模：利于用户界面，输入，系统功能和控制，输出，维护和自检的表达方式，可以建立一个系统构建模型，为后面每个工程规范步骤建立良好的基础②UML系统模型：(1)部署图:部署图每个三位方和描述一个属于系统物理架构的硬件要素 (2)活动图：Ⅰ圆角矩形：特定的系统功能Ⅱ箭头：系统流程Ⅲ菱形：分支Ⅳ水平线：并发事件(3)类图和用例图：椭圆表示一个用例

**Chapter7 需求工程-----------------------------------------------------------------------------------------------**

**\*需求工程任务**：为以下工作提供良好机制：理解客户需要什么，分析需求，评估可行性，协商合理的方案，无歧义的详细说明方案，确认规格说明，管理需求以至将这些需求转化为可执行系统。活动有：起始，导出，精化(是一个分析建模动作，最终结果是形成一个分析模型，定义问题提供的信息，功能和行为域)，协商，规格说明，确认和需求管理

**\*需求工程工作产品**：①必要性和可行性陈述②系统或产品范围的界线说明③参与需求导出的客户，用户和其他共利益者的列表④系统技术环境的说明⑤需求列表（推荐按照功能加以组织）以及每个需求使用的领域限制⑥一系列使用场景，有助于深入了解系统或产品在不同运行环境下的使用⑦任何能够更好的定义需求的原型。

**\*需求开发的方法**：①需求获取：1)起始过程：与客户建立初步交流2)导出过程：通过访谈与调查（会议，问卷，原型）获得需求的描述②需求分析：精化过程：通过分析建模，建立精确的技术模型，说明软件的功能，特征和约束③需求处理：1)协商过程2)形成规格说明④需求确认

**Chapter8 构建分析模型-----------------------------------------------------------------------------------------**

**\*分析模型的作用**：分析建模使用文字和图表的综合形式，以相对容易理解的方式描绘需求的数据、功能和行为，更重要的是，可以更直接地评审他们的正确性、完整性和一致性。

**\*分析模型的构建原则**：①模型应关注在问题域或任务域内可见度的需求，抽象的级别应该相对高一些②分析模型的每个元素都应该增加对软件需求的整体理解，并提供对信息域、功能和系统行为的深入理解③基于基础结构和其他非功能的模型应推延到设计阶段再考虑④最小化整个系统内的关联⑤确认分析模型为所有共利益者都带来好处⑥尽可能保持模型简洁

**\*方法：**

**\*场景建模**：从用户的角度描述软件需求，①用例——参与者和原件之间的某个交互的叙述或模板驱动说明——是主要的建模元素。在需求获取过程中提取的用例，定义了特定功能或交互的关键步骤，用例的形式化和详细程度各不相同，但是最终结果为所有的其他分析建模活动提供了必须的输入。②还可以使用活动图说明场景，描述在特定场景中的处理流。③泳道图显示了处理流如何分配给不同的用户和类。

\*（1）用例图：从用户角度描述系统功能，并指出各功能的操作者。

\*（2）部署图：是用来显示系统中软件和硬件的物理架构

**\*类建模**：使用从基于场景和面向流的建模元素中提取的信息确定分析类，可以使用语法分析从文本叙述中提取候选类，属性和操作，并制定了用于定义类的标准，CRC索引卡可以用于定义类之间的联系，此外，可以使用各种UML建模符号定义类之间的层次、联系、关联、聚合和依赖，使用分析包可以将类分类和分组，在某种意义上为大型系统提供了更好管理

\*(1)类图：描述系统中类的静态结构。不仅定义系统中的类，表示类之间的联系如关联依赖聚合等，也包括类的内部结构（属性和操作）

\*(2)协作图：描述对象间的协作关系，跟顺序图相似，显示对象间的动态合作关系

**\*行为建模**： 描述了软件的动态行为，使用基于场景，面向流和基于类的元素作为输入，从整体上表现分析类和系统的状态，要实现这一点，要识别状态，定义导致类或系统做出状态转移的事件，以及确认当转移完成时所发生的动作，状态图和顺序图常使用

\*(1)状态转换图：描述类的对象所有可能的状态以及事件发生时状态的转移条件

\*(2)活动图：通过提供特定场景内交互流的图形化表示来补充用例

\*(3)顺序图：显示对象之间的动态合作关系，强调对象之间消息发送的顺序，同时显示对象之间的交互

**\*数据字典**：由字典条目组成，每个条目描述DFD中的一个元素。数据流图上所有成分的定义和解释。要求：完整性、一致性、可用性

**\*数据建模**：①用ER图(实体---关系图)来建立数据模型②数据建模的基本元素：数据对象、属性和关系③ 数据属性特征：（1）为数据对象的实例命名（2）描述这个实例（3）建立对另一个表中的另一实例的引用④作用：数据建模工具为软件工程师提供表现数据对象，数据对象的特点和数据对象的关系的能力，主要用于大型数据库系统和其他信息系统项目数据建模工具以自动化的方式创建全面的实体关系图，数据对象词典以及相关模型

**\*流建模**：关注当数据对象通过处理函数转换时的流动，在结构化分析的过程中提取的流模型使用数据流图，即一种建模符号，描述当数据对象在系统中移动时输入如何准变为输出，使用处理规格说明来说明处理数据的每一个软件功能，除了数据流，该建模元素还说明了控制流，这是显示事件如何影响系统行为的一种表现形式

**Chapter9 设计工程------------------------------------------------------------------------------------------------**

**\*概念**

**\*抽象**abstraction：①数据抽象：描述数据对象的冠名数据集合②过程抽象：具有明确和有限功能的指令序列

**\*体系结构**architecture：是程序构件（模块）的结构或组织，这些构件交互的形式及这些构件所用数据的结构。

**\*模式**patterns：描述在某个特定场景与可能影响模式应用和使用方式的影响力中解决某个特定的设计问题的设计结构

**\*逐步求精/精化**refinement：在原始陈述上细化，并伴随每个精化的持续进行将提供更多的细节

**\*模块化**modularity：软件被划分成为独立命名的，可寻址的构件，有时被成为模块，集成在一起可以满足问题的需求

**\*信息隐藏**information hiding：每个模块对其他所有模块都隐藏自己的设计决策

**\*模块独立**functional independence：每个模块仅涉及需求的某个特定的功能，只有一个简单的接口

**\*重构**refactoring：不改变代码的外部行为，改变其内部结构，使其更易被理解与维护

**\*四种设计模型**：构件级设计，接口设计，体系结构设计，数据（类）设计

**Chapter10 体系结构设计--------------------------------------------------------------------------------------**

**\*为何进行体系结构设计**：①软件体系结构的表示有助于对计算机系统开发感兴趣的各方开展交流；②体系结构突出早期设计决策，这些决策对随后的所有软件工程工作有深远的影响，同时对系统作为一个可运行实体的最后成功有重要作用；③体系结构“构建了一个相对较小的，易于理解的模型”，该模型描述了系统如何构成以及其构件如何一起工作

**\*体系结构风格**：①以数据为中心的体系结构 ②数据流体系结构 ③调用和返回体系结构④面向对象体系结构 ⑤层次体系结构

**Chapter11 构件级建模-------------------------------------------------------------------------------------------**

**\*什么是构件**：系统中某一定型化的，可配置的和可替换的部件，该部件封装了实现并暴露一系列接口。面向对象观点：构件包括一个协作类集合；传统观点：构件是程序的一个功能要素。

**\*构件的设计原则**：开关，替换，依赖倒置，接口分离（1）开关原则，模块应该对外延具有开放性，对修改具有封闭性（2）Liskov替换原则，子类可以替换它们的基类

（3）依赖倒置原则，依赖于抽象而非具体实现（4）接口分离原则，多个用户专用接口比一个通用接口要好（5）发布复用等价性原则，（6）共同封装原则，（7）共同复用原则

**\*内聚性**：在为面向对象系统进行构件级设计时，内聚性意味着构件或者类只封装那些相互关联密切，以及与构件或类自身有密切关系的属性和操作。类型有：功能，分层，通信，顺序，过程，暂时，实用内聚

**\*耦合性**：耦合是类之间彼此联系程度的一种定性度量。耦合分类：内容，共用，控制，印记，数据，外部耦合，非直接耦合（例程调用，类型使用，包含或导入）

**\*构件设计方法**：程序流程图，决策表，PDL

**\*流程图**：方框表示处理步骤，菱形表示逻辑条件，箭头表示控制流。顺序型，条件型，重复型

**\*决策表**：提供一种表示方法，可将动作和条件翻译成表格，表格难被误解，且能够做为某一个表驱动算法的机器可识别的输入。决策表分为四部分：左上部列出了所有的条件，左下部列出了所有可能的动作，右半部构成了一个矩阵，表示条件的组合以及特定条件组合（上）对应的动作（下）。

**\*PDL程序设计语言**：结构化的英文或伪代码，是“一种混合语言，采用一种语言（英语）的词汇和另一种语言的语法（机结构化编程语言）”。

**\*比较**：①异：传统（设计需要充分表现出数据结构，接口和程序块的算法以指导变成语言源代码的产生，可使用图形、表格或文本格式。一个构件就是程序的一个功能要素，细化控制模块、问题与模块和基础设施模块三种）OO（以类为基础，构件包含一个协作类集合，注意细化来自于问题域和基础设施域的设计类）②同：都来自于分析模型，都需要应用那些导致高质量软件的基础设计原则和概念

**Chapter13-14 测试------------------------------------------------------------------------------------------------**

**\*Validation(确认)VS Verification(验证)** ：①确认：我们在构造正确的产品吗？验证：我们在正确地构造产品吗？②相同点：使用同样的技术；③不同点：修改的参考不同（验证：用上一步的结构做参考； 确认：用用户的需求做参考）

**\*测试策略**

**\*单元测试unit test**：①侧重于软件设计的最小单元的验证工作。单元测试侧重于构件中的内部处理逻辑和数据结构。单元测试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主要内容: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④包括单元测试考虑、单元测试规程（在大多数应用中，驱动程序只是一个“主程序”，它接受测试用例数据，将这些数据传递给构件并打印相关结果。桩程序的作用是替换那些从属于将要测试的构件或被其调用的构件。）

**\*集成测试integration test**：①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。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。②一步到位的集成方法：所有构建事先连接在一起，全部程序作为一个整体进行测试。③增量集成之自顶向下：一种构造软件体系结构的增量方法，模块的集成顺序从主控模块开始，沿着测试层次结构逐步向下，利用深度优化或广度优化的方式将从属于主控模块的模块集成到结构中去     ④自底向上：从程序结构的最底层模块（即原子模块）开始，然后按程序结构图的控制层次将上层模块集成到整个结构中，并对其进行测试。⑤回归测试：重新执行已进行测试的某个子集，以确保变更没有传播不期望的副作用。⑥冒烟测试：是时间关键性项目的步进机制，让团队频繁对项目进行评估。daily bulid,尽早发现不相容性和业务阻塞错误

**\*确认测试validation test**：①确认测试始于集成测试的结束，那时已测试完的单个构件，软件已组成完整的软件包，且接口错误已被发现和改正。②确认测试是根据软件需求规约对集成的软件进行确认，主要揭露不符合**需求规约**的错误。

**\*系统测试system test**：①系统测试实际上是对整个基于计算机的系统进行一系列不同考验的测试。②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。③目的：发现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）④系统测试=功能测试+非功能测试

**\*测试用例**：① 以设计或源代码为基础，画出相应的流程图 ② 确定所得流图的环复杂性 ③ 确定线性独立路径的基本集合 ④ 准备测试用例，强制执行基本集合中的每条路径。 用例内容有输入，输出，条件

**\*测试技术**：①穷举：程序可能执行的路径数为巨型指数，这对开发进度造成巨大的障碍。②不可能原因：穷举测试分为穷举输入测试和穷举路径测试。穷举输入模式是绝对不可能实现的，穷举路径测试即使能行得通，仍然可能有错误。1.穷举路径测试绝不能查出程序违反了设计规范，即程序本身是个错误的程序。2.穷举路径测试不可能查出程序中因遗漏路径而出错3.穷举路径测试可能发现不了一些与数据相关的错误。

**\*黑盒**：①也称行为测试，指在接口处执行测试，侧重软件的功能需求，而不考虑程序的内部结构。侧重于软件的信息域。②用处：发现错误： （1）功能不正确或遗漏（2）接口错误（3）数据结构或外部数据库访问错位（4）行为或性能错误（5）初始化和终止错误③ 生成满足下述准则的测试用例集：减少达到合理测试所需的附加测试用例数、能够告知某些错误类型是否存在，而不是仅仅知道与特定测试相关的错误④方法：等价划分。边界值分析。正交数组测试

**\*白盒**：①也称结构测试或逻辑驱动测试。侧重于程序控制系统。基于过程细节的封闭检查。通过提供检查特定条件及和循环的测试用例，测试贯穿软件的逻辑路径和构件间的协作②用处：保证一个模块中的所有独立路径至少被执行一次、对所有的逻辑值均需测试真假，在上下边界及可操作的范围能执行所有的循环、检查内部数据结构以确保其有效性③用例：基本路径测试利用程序图（或图矩阵）生成保证能覆盖率的线性无关的测试集。条件和数据流测试进一步检查程序逻辑，循环测试检查不同复杂度的循环④白盒测试在测试过程的早期执行，而黑盒测试倾向于应用在测试的后期阶段。

**\*手工测试VS自动化测试：**①手工测试：发现缺陷率高，容易实施，创造性、灵活性，覆盖率量化困难，重复测试效率低，不一致性、可靠性低，依赖人力资源②自动化测试：高效率、速度，高复用性，覆盖率容易度量，准确、可靠，不知疲劳，激励团队士气③两者相互补充：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；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；由于工具问题，自动测试只能发现15%错误，而手工测试可以发现85%缺陷。

**Chapter21 项目管理----------------------------------------------------------------------------------------------**

**\*4P’s**： 人员，产品，过程，项目

**\*W5HH**: 为什么开发这个系统？将要做什么？什么时候做？某功能由谁负责？他们的机构组织位于何处？如何完成技术和管理工作？每种资源需要多少

**Chapter15 22 度量------------------------------------------------------------------------------------------------**

**\*McCall的质量因素：**①影响因素分为两大类：可以直接测量的因素（如测试期间发现的缺陷）；只能间接测量的因素(如易用性和可维护性)②正确性、可靠性、效率、完整性、易用性、可维护性、灵活性、可测试性、可移植性、可复用性、互操作性。

**\*Measures测量:** ”测度(measure)“为产品或过程的某些属性的程度、数量、维数、容量或大小提供量化的指标，而”测量(measurement)“是确定测度的动作

**\*Metrics度量:**度量是一个系统、构件或过程具有给定属性量化测量程度。

**\*Indicators指标:**是一个度量或者多个度量的组合，它提供了对软件过程，软件项目或产品本身的深入理解

**\*度量的作用**：①过程度量作用：提供能够引导长期的软件过程改进的一组过程指标。②项目度量作用：使得软件管理者能够（1）评估正在进行中的项目的状态（2）跟踪潜在的风险（3）在问题造成不良影响前发现他们（4）调整工作流程或任务（5）评估项目团队控制软件工作产品质量的能力③产品度量作用：为分析、设计、编码和测试能更客观的执行和更定量的评估提供基础

**Chapter23 估算----------------------------------------------------------------------------------------------------**

**\*项目计划任务和内容**：①项目计划任务 1、规定项目范围 2、确定可行性3、分析风险 4、确定需要的资源（a、确定需要的人力资源；b、确定可重用的软件资源 c、标识环境资源）5、估算成本和工作量（a、分解问题b、使用规模，功能点，过程任务或用例等方法进行两种以上恩德估算c、调和不同的估算） 6、制定项目进度计划（a、建立一组有意义的任务集合b、定义任务网络c、使用进度计划工具指定时间表d、定义进度跟踪机制）②项目计划内容估算，进度安排，风险分析，质量管理计划和变更管理计划

**\*LOC(面向规模的度量)&FP(面向功能的度量)：**LOC以**代码行**作为其他测量的规范化因子，FP(功能点)则是从**信息域及对问题复杂度的主观评估**中导出的

\*基于规模的估算：LOC估算和FP估算是两种不同的估算技术，但具有共性：①均以**问题分解**为基础，LOP更甚，问题分解越详细越好。②要求WBS(work breakdown structure)中每个工作包都是可以分别独立估算的③每个工作包都有基线生产率度量值可以参考，或者可以预估其生产率。④基线生产率度量：根据LOC和FP的测量，计算出软件开发组织对某类问题的生产率，LOC/pm或FP/pm

**Chapter24 进度-----------------------------------------------------------------------------------------------------**

**\*项目工作量分配原则40-20-40**：总体工作量的40%分配给前期的分析和设计，20%用于编码，40%用于后期测试

**\*任务网络的作用**：活动网络，是一个项目任务流程的图形表示。①优点：很好地展示优先关系；定义关键路径的能力；执行“如果”分析的能力 ②缺点：默认模型假定资源是无限的，我们需要加紧资源依赖关系当确定自己的“真正”的关键路径；难以跟随大的项目

**\*关键路径：**①机动时间：在不影响整个工期的情况下，当前任务允许延迟的最长时间②关键路径：由机动时间为0的任务组成的项目路径。

**\*甘特图、里程碑的作用**：菱形--里程碑，粗黑线--总任务，细黑线--任务，箭头--任务间的相互关系

**\*甘特图：**①也称时间表/时序图，用来建立项目进度表，每项任务的完成以必须交付的文档和通过评审为标准，因此它们作为**项目里程碑**。②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。

**\*里程碑**：敏捷、详细、可度量、可分配、现实性、期间时限

**Chapter25 风险-----------------------------------------------------------------------------------------------------**

**\*被动和主动风险管理：**①被动：项目团队在发生风险时对其做出反应；缓解 - 预计消防的额外资源计划；(救火模式)修复失败-当风险发生，才找到资源并应用；危机管理 - 失败不响应应用资源，项目处于危险之中②主动：执行正式风险分析；组织纠正风险的根本原因③主动风险管理过程：风险识别，分析，规划，监控

**\*Risk Management Paradigm（风险管理过程）：**

**\*RMMM(风险规划)**: 风险避免，风险监测和风险管理计划（一个有效的策略、风险规划要考虑的三个问题）①风险避免：对付风险最好地方法是主动地避免风险②风险监测：监测可以提供风险指示的因素。目的：评估预测风险是否发生，保证正确实施各项缓解步骤，收集能够用于今后风险分析的信息③风险管理：前提：风险避免和缓解失败，风险已发生④RMMM目录：1.引言2.风险分析3.风险管理4.附录

**Chapter26 质量----------------------------------------------------------------------------------------------------**

**\*软件质量因素**：①软件要符合明确规定的功能和性能需求，符合已清晰文档化的开发标准，并且具有专业人员开发的软件所应有的隐含特征②当根据对象的可测量特征考察一个对象时，可以有两种不同质量：（1）设计质量：设计者为一个产品规定的特征，包括系统的需求，规格说明和设计（2）一致性质量：在制造产品过程中遵守设计规格说明的制度。③质量成本包括预防成本，失效成本，鉴定成本④用户满意度=合格的产品+好的质量+按预算和进度安排交付

**\*软件质量保证活动**：①SQA小组职责辅助软件团队实现高质量软件产品②SQA的工作：(1)为项目准备SQA计划(2)参与开发项目的软件过程描述(3)评审各项软件工程活动，以检验是否符合定义的软件过程(4)审核指定的软件工作产品，以验证其是否符合定义的软件过程中的相应部分(5)确保软件工作及工作产品中出现的偏差已文档化，并按文档化的规程进行了处理⑥记录所有不符合的部分，报告高层管理者

**\*正式技术评审：**①是一项由软件工程师(以及其他人)进行的软件质量控制活动。②目标：(1)发现软件的任何一种表示形式中的功能、逻辑或实现上的错误；(2)验证评审中的软件是否满足其需求；(3)保证软件表示符合预先定义的标准；(4)得到以统一的方式开发的软件；(5)使项目更易于管理。③作用：(1)提供培训机会，使初级工程师能够了解软件设计和实现的不同方法(2)使大量人员对软件系统中原本不熟悉的部分更了解(3)培训后备人员和促进项目连续性④评审会议条件：（1）通常3-5人（2）提前准备，占用每人工作时间不超过2小时（3）会议时间小于2小时

**\*软件质量的成本**：预防成本、鉴定成本、失效成本①预防成本：包括质量计划，正式技术评审，测试设备及培训②鉴定成本：包括为深入了解“首次通过”各个过程时的产品状态而开展的那些活动③失效成本：如果在将产品交付给客户之前没有缺陷的话就不会存在的成本。分为内部失效成本和外部失效成本。(1)内部失效成本是指在产品交付给客户之前发现了错误而引发的成本，包括返工、修复以及失效模式分析(2)外部失效成本是指与产品交付给客户之后所发现的缺陷相关的成本，如解决客户抱怨、退换产品、保修工作等。

**Chapter27 变更----------------------------------------------------------------------------------------------------**

**\*软件配置项SCI：**①在软件工程过程中创建的信息。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。②包括：计算机程序（源代码和可执行程序）：描述计算机程序的文档（针对技术开发者和用户）；数据（包含在程序内部的数据，或程序外部的数据）、开发环境

**\*版本**：与计算机软件配置项的完全按编篡或重编纂相关的计算机软件配置项的初始发布或再发布

**\*基线**：已经通过正式评审和批准的规格说明或产品，它可以作为进一步开发的基础，并且只有通过正式的变更控制规程才能修改它。

**\*发布**：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）

**\*软件配置管理SCM流程**：①SCM任务：标识，版本控制，变更控制，配置审核，报告②(1)标识：为控制和管理软件配置项，必须对每个配置项单独命名，然后用面向对象方法进行组织。可以进行标识的对象：基本对象和聚合对象。(2)版本控制：结合规程和工具，用来管理在软件过程中所创建的配置对象的不同版本。(3)变更控制：(4)配置审核：通过正式技术评审或软件配置审核来保证变更的有效性。是正式技术评审的补充。(5)报告：配置状态报告CSR（状态记录），解答了问题，发生什么事，是谁做的，什么时候发生的，会影响到别的什么。可使管理者和开发人员可以评估重要的变更。