第一章 软件工程概述

1. 什么是软件？软件是包含**程序、数据及其相关文档**的完整集合，围绕数据满足用户需求的人工制品。

①程序：按实现设计的功能和性能要求执行的指令序列

②数据：使程序能正常操纵信息的数据结构

③文档：与程序开发、维护和使用有关的图文材料

2. 特征：形态、智能、开发、质量、生产、管理、环境、维护、废弃、应用

特征的本质：复杂、一致、可变、不可见

3. 软件危机：计算机软件开发&维护过程中遇到的严重问题（成本和进度难估计、不可靠、不安全、维护难）

4. **软件工程定义：**①系统化的、规范的、可以度量的方法运用于软件开发、运行和维护的过程（工程化在软件方面的作用）②对①所描述方法的研究。**第二章到第九章就是软件工程的流程**。

5. ★★★软件生存期模型★★★

①瀑布模型：**串行过程、需求明确、反馈环、顺序性**。

优点：**强迫**开发人员采用规范方法，**严格规定**了每个阶段必需的文档，要求每个阶段交出的产品是**经过验证**的

缺点：最终产品满足不了用户需要、只能用于项目开始的时候需求已确定的情况（**需求固定**）。

②快速原型模型：

优点：**需求可变、需求更明确，更贴近用户需求，基本线性，避免了后期返工，快速实现应付市场竞争**。

缺点：**过程不可见，结构性差，需要特殊工具和技术**

③增量模型（演化模型）：迭代和演进。产品拆解为增量模型。模块化。先前的增量为之后的增量服务（“极限程序设计”）。

优点：**短时间出产品、失败风险低、重要部分接受最多的测试**

缺点：**难将用户映射到模型上，难以定义增量用到的服务**

④螺旋模型：螺旋上升的**制定计划、风险分析、实施工程、客户评估->下一阶段，相比①和③新增了风险分析**

⑤统一过程

工作流：**①业务建模②需求③分析和设计④实现⑤测试⑥部署**

阶段：**①初始②细化③构造④移交**

第二章 需求工程与结构化分析方法

1. 软件需求：用户解决问题 or 达到目标所需的条件 and 能力；系统和系统部件要满足合同、标准、规范所需的条件 or 能力；反映上述的条件 or 能力的文档。

2. 业务需求：对系统高层次目标要求，是远景。确定产品发展方向、功能范围、目标客户和价值来源

3. 用户需求：只涉及系统外部行为的非功能性要求

4. 系统需求：详细地描述系统该做什么，包括了分析模型（对象、数据、状态模型）

5. **功能需求：描述系统应该提供的功能和服务，比如人机交互，系统间交互，是抽象的**。

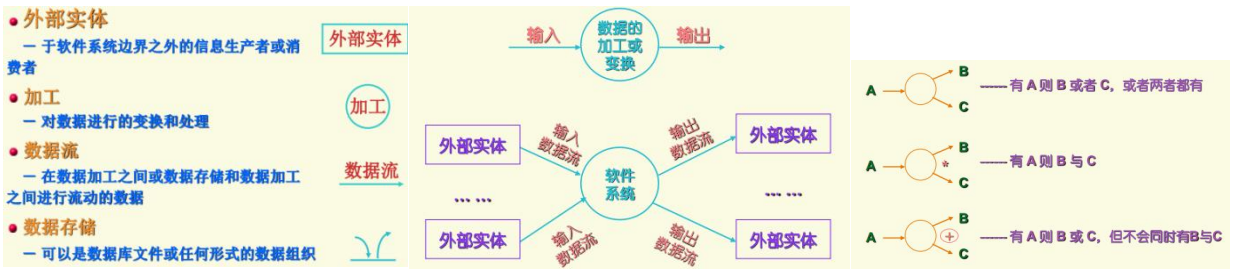
6. **非功能需求：各角度对系统的约束和限制，反映应用对系统质量和特性的额外需求，比如响应时间、数据精度、可靠性、开发过程标准等**。

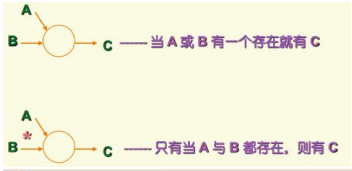
7. 需求工程：与需求相关的活动。（调查、分析、定义、规格说明、分析模型、会议记录）

8. 需求管理：验证、跟踪、变更控制

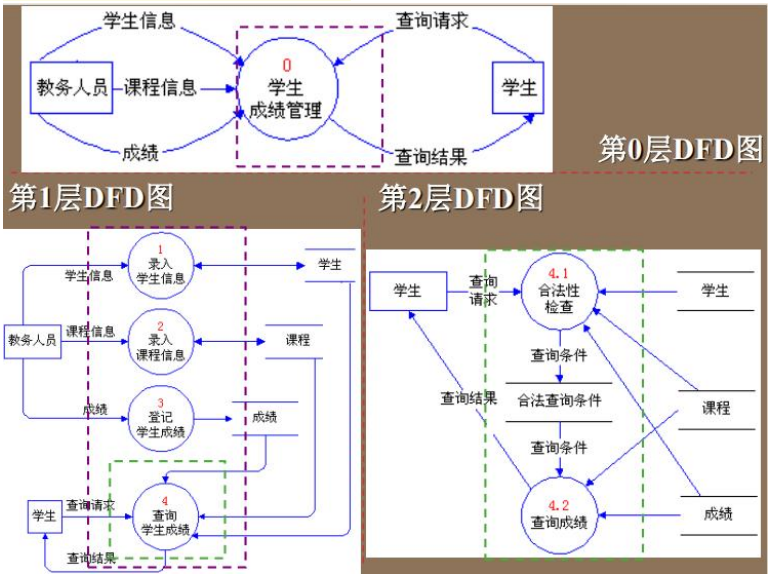
9. **★★★结构化分析方法★★★**

①数据流图（DFD）





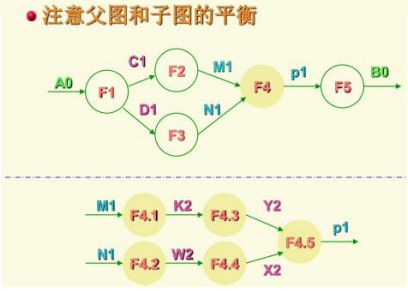
在该系统中，教务人员录入学生信息、课程信息和成绩信息，学生可以随时查询自己所选课程的成绩。由于学生成绩属于敏感信息，系统必须提供必要的安全措施以防非法存取。



- 区分局部文件和局部外部项
- 掌握分解的速度

  一般来说，每一个加工每次可分为 2-4个子加工，最多不得超过 7 个。
- 遵守加工编号规则

  顶层加工不编号。第二层的加工编号为1, 2, 3, ..., n号。第三层编号为1. 1, 1. 2, 1. 3...n. 1, n. 2...等号，依此类推。
- 注意父图和子图的平衡



0

层箭头上的文字，在 1 层必须出现且不重复，而 2 层则是最细的加工，在题干会说明，比如本题提到的学生信息的防范和合法检查。

②实体关系图（ERD）：就是数据库的 ER 图，在这里记得标注 1、M、N 用于表示一对一、多对多关系

③状态转换图（STD）：双圆圈是开始和结束，其他的同状态图

10. 数据字典（DD）：将分析模型中图形元素作为词条来定义，让他们有确切的解释。

符号	含 义	例 子
=	被定义为	年 = “1900”..“3000”
+	与	x=a+b，则表示 x 由 a 和 b 组成
[ ]	或	x=[a,b]，则表示 x 由 a 或由 b 组成
{ }	重复	x={a}，则表示 x 由 0个或多个 a 组成
m{ }n	重复	x=3{a}8，则表示x中至少出现3次a ,最多出现8次
( )	可选	x=(a)，则表示 a 在 x 中出现，也可不出现
...	注释符	表示在两个 * 之间的内容为词条的注释

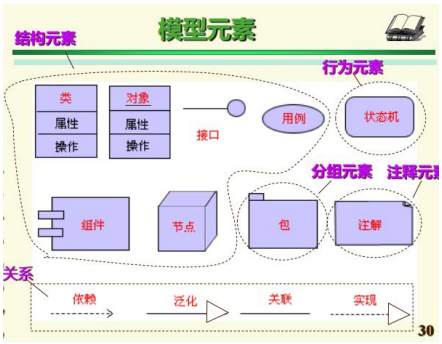
第三章 面向对象分析（OOA）

1. 面向对象：对象+类+继承+通信

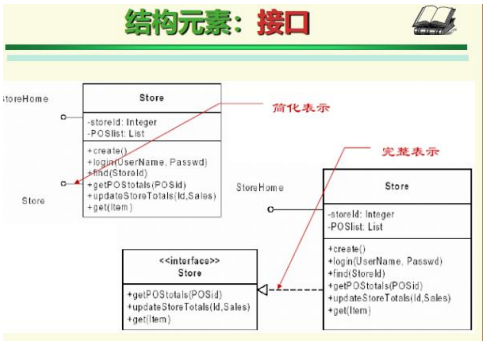
2. ★★★统一建模语言：UML★★★

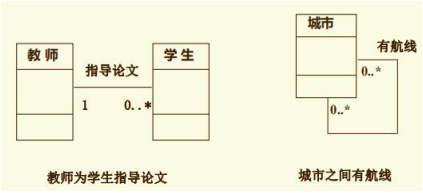
①定义：**直观、明确**，构建和文档化软件系统产物的通用**建模语言**

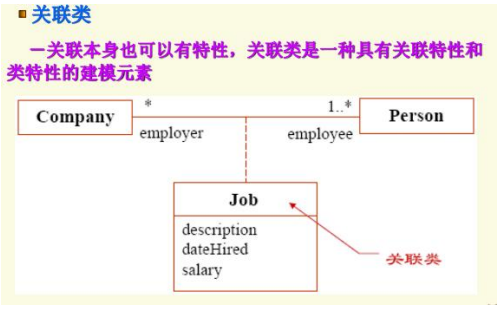
②特点：统一标准、面向对象、可视化、独立于过程、易掌握、与程序设计语言相关

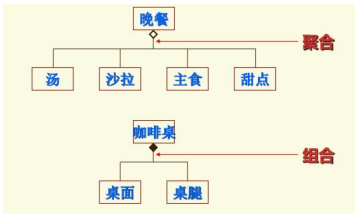


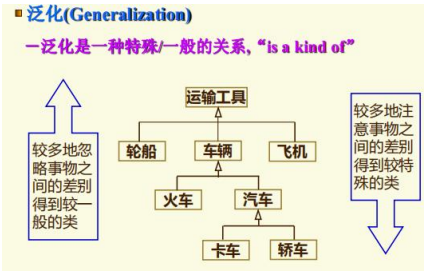
- 可见性
  - public +：公有的（缺省值），对于一个给定的类，任何一个带有可见性的外部类都可以使用该特征
  - protected #：受保护的，类的任何子类都可以使用该特征
  - private －：私有的，只有类本身可以使用该特征
- 抽象类
  - 抽象类是不能直接产生实例的类
  - 在UML中，将类名写成斜体字来表示抽象类，也可以使用构造型<<abstract>>



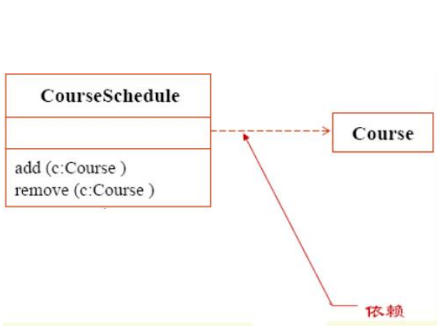




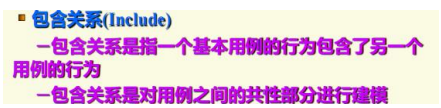


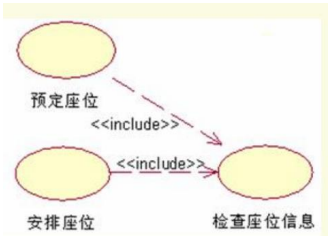


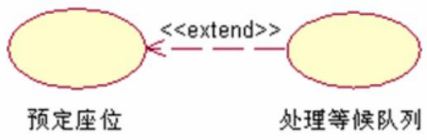
9



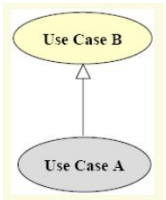
3. 用例图







用况：登记借书
1. 目标
本用况允许图书管理员登记普通读者的借书记录。
2. 事件流
2.1 基本流程
当普通读者希望借书，图书管理员准备登记有关的借书记录时，本用况开始执行。
(1) 系统请求图书管理员输入读者的注册号和所借图书的书目；
(2) 图书管理员输入有关信息后，系统产生一个唯一的借书记录号；
(3) 系统显示新生成的借书记录；
(4) 图书管理员确认后，系统增加一个新的借书记录。
2.2 可选流程
(1) 读者没有注册
在主流程中，如果系统中没有读者的注册信息，系统将显示错误信息，用况结束。
(2) 所借图书书目不存在
在主流程中，如果所借图书已被借出或者系统中没有该图书的书目，系统将显示错误信息，用况结束。
3. 特殊需求
无。
4. 前提条件
用况开始之前，图书管理员必须在系统登录成功。
5. 后置条件
如果用况执行成功，该读者的借书记录被更新，否则，系统状态不变。



4. 类图

①实体类：存放信息　②边界类：人机交互　③控制类：描述用例

5. 用例描述（用例规格说明）（第一次作业）：用例目的、事件流（参与者、系统）、基本流和可选流、前置条件后置条件

6. 时序图：动态视图，基于类的元素作为输入，整体上表现分析类和系统的状态。一个流对应一个图，备选流过于简单可以合并。复杂可以拆分。实体对象在该图中一般不访问边界和控制对象。

第四章 总体设计

1. 内容和过程：

①任务：需求开发->系统结构设计（总体设计）->数据、过程、界面设计（详细设计）->编程和测试

②软件体系结构：软件系统的总体组织、全局控制、数据存取、与子系统的通信协议

③典型的软件体系结构：仓库、分层、模型/视图/控制器、客户机/服务器

MVC 模式和三层架构

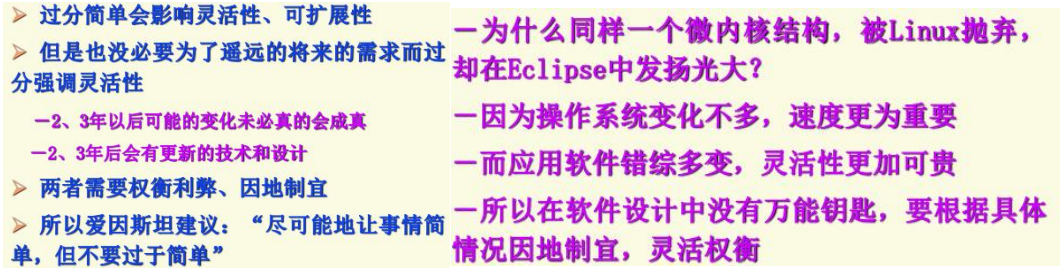
2. 设计原则

①简单设计②因地制宜③套用成解（用现成）**④模块化：分而治之**⑤逐步求精

⑥信息隐藏：减少局部设计对全局的影响、强调接口通信、不鼓励用全局数据、有助于封装、有助于提高软件质量

⑦模块独立：耦合衡量模块之间联系的紧密度，**低耦合**代表了模块良好的独立性

内聚衡量模块内部元素的紧密度，**高内聚**代表了模块良好的独立性



3. 启发式规则：①提高模块独立性②模块大小适中（50-100 句）③控制深度（分层）、宽度（同一层模块数）、扇出（某模块直接调用/控制的模块数，3 到 9 合适）、扇入（直接调用该模块的模块数，越大越好）

第五章 面向对象设计（OOD）

1. 定义:按照事先条件对 OOA 模型调整，补充新的组成部分，是**对 OOA 的调整和增补**

2. OOA 产生**满足用户需求，独立于实现（问题域）**的模型；OOD 产生**可实现（计算机世界）**的模型。

①编程语言：多继承——>单继承　②性质　③重用已有类　④继承：LSP 原则 ⑤抽象类

LSP：派生类必须通过基类接口被使用，使用者无需了解其差异。基类定义的子程序，用在它任何一个派生类中的含义都是相同的。这个原则降低复杂度，十分抽象。

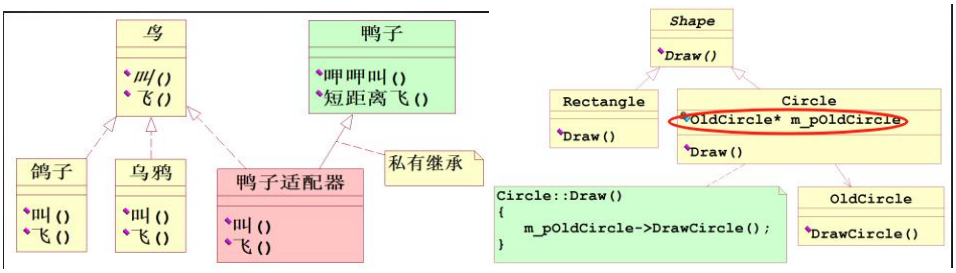
第六章 设计模式和面向对象设计的基本原则

1. 策略模式：算法独立变化，组合代替继承，封装可变性&开闭（OCP）（软件可以在不修改的前提下扩展）

2. 桥梁模式：抽象和实现独立变化，避免两个变化的耦合

3. **适配器模式**

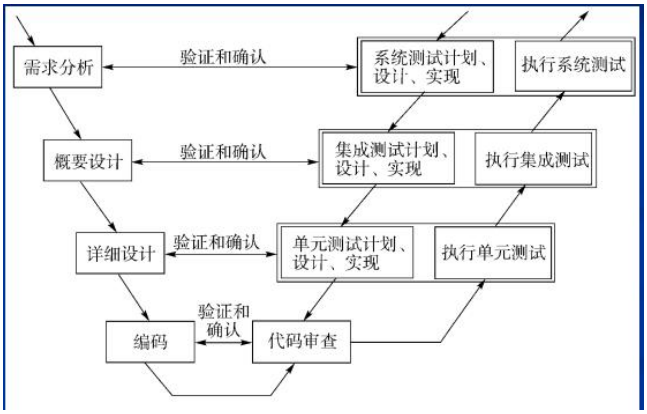
①类适配器（继承后，通过接口实现）　②对象适配器（内部创建类，只用到接口）



③依赖倒置原则：抽象不依赖于细节，细节依赖于抽象（具体层高层模块依赖抽象层低层模块）

第七章 软件测试

1. V 模型

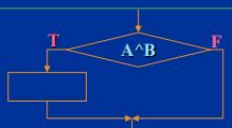
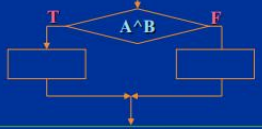


2. ★★★软件测试方法★★★

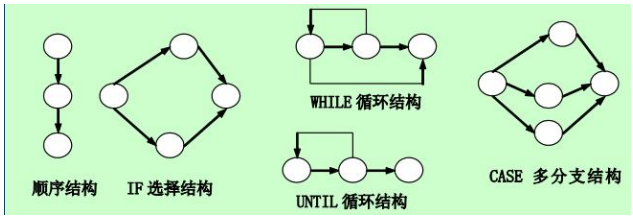
①白盒测试：逻辑覆盖、基本路径测试

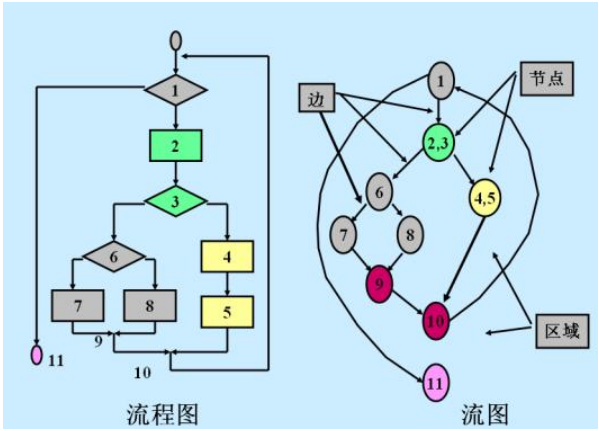
逻辑覆盖测试的五种标准

发现错误的能力	标 准	含 义
1(弱)	语句覆盖	每条语句至少执行一次
2	判定覆盖	每一判定的每个分支至少执行一次
3	条件覆盖	每一判定中的每个条件，分别按“真”、“假”至少各执行一次
4	判定/条件覆盖	同时满足判定覆盖和条件覆盖的要求
5 (强)	条件组合覆盖	求出判定中所有条件的各种可能组合值，每一可能的条件组合至少执行一次

覆盖标准	程序结构举例	测试用例应满足的条件
语句覆盖		A^B=.T.
判定覆盖		A^B=.T. A^B=.F.

覆盖标准	程序结构举例	测试用例应满足的条件
条件覆盖		A=.T. A=.F. B=.T. B=.F.
判定/条件覆盖		A^B=.T. ，A^B=.F. A=.T. A=.F. B=.T. B=.F.
条件组合覆盖		A=.T. ^ B=.T. A=.T. ^ B=.F. A=.F. ^ B=.T. A=.F. ^ B=.F.





在选择或多分支结构中，分支的汇聚处应有一个汇聚结点

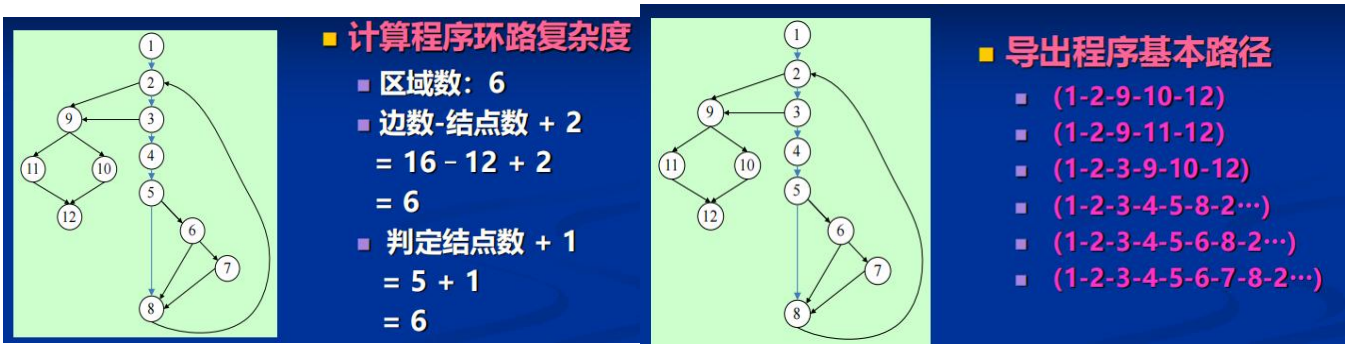
边和结点圈定的区域叫做区域，当对区域计数时，图形外的区域也应记为一个区域

如果判断中的条件表达式是由一个或多个逻辑运算符(OR, AND, ...)连接的复合条件表达式，则需改为一系列只有单个条件的嵌套的判断。

- if a and b then x else y

确定程序控制流图的环路复杂度

- 程序的环路复杂性给出程序基本路径集合中的独立路径条数
- 确保程序中每个可执行语句至少执行一次所必须的测试用例数目的上界
- 独立路径：指至少引入一个新处理语句或一条新判断的程序通路
- 程序环路复杂性的三种方法：
  - 程序控制流图中的区域数
  - 程序控制流图的边数 - 程序控制流图的结点数 + 2
  - 程序控制流图中的判定结点数 + 1



②黑盒测试：等价类（有效&无效）划分（等价分类表&测试用例表）、边界值分析、错误推测、因果图

<u>原则 1</u>：输入规定取值范围，就可以确定一个有效等价类和两个无效等价类

原则 2：输入的集合规定了必须得条件，就可以确定一个有效等价类和一个无效等价类

原则 3：输入为 boolean，就可以确定一个有效等价类和一个无效等价类

3. 软件测试阶段

①**<u>单元测试</u>**：　　• 构造环境：驱动程序、桩模块　　• 测试内容：单元接口、局部数据结构、独立路径

②集成测试：自顶向下，自底向上，回归测试

③系统测试：恢复测试、安全测试、压力测试、性能测试、α 测试、β 测试

第八章 软件维护

其他维护 5% 改正性维护 20% 适应性维护 25% 完善性维护 50%。维护总占 70.8%。

第九章 软件项目的管理

1. 团队

①项目经理：懂技术还是懂管理的问题（具体问题具体分析）

②团队组织：团队不是群体。合作性、思考性、自主性。

③民主制程序设计小组：小规模（2-8），无领导（平等民主）、开源，适合难度大易出错的项目，但缺乏协调沟通。

④主程序员组：技术+管理人才当负责人，后备程序员当替补，编程秘书负责测试编辑编译链接等，程序员当码农

2. 项目计划与进度跟踪

①甘特图（Gantt）：简单，动态反应进展；难以反应多任务间逻辑

②工程网络（工程评价技术 PERT）：顶点（事件瞬间）、边（权值代表所需时间）

- 机动时间：事件最早&最晚发生时间
- **关键路径（CPM）：机动时间为 0 的事件组成**