

第一章 软件工程概述

1. 开设软件工程课程的目的？/为什么需要软件工程？

软件危机：

●软件危机（落后的软件生产方式无法满足迅速增长的计算机软件需求，从而导致软件开发与维护过程中出现一系列严重问题的现象）

软件产业的发展长期滞后，与硬件发展不协调。软件开发成本过高。软件质量得不到保证。软件开发效率低。难以控制开发进度，工作量估计困难

软件不能满足社会发展的需求，成为社会、经济发展的制约因素

●软件危机原因（软件开发方法仍然沿用早期的个体化软件开发方式，但软件的数量急剧膨胀，软件需求日趋复杂，维护的难度越来越大）

- 软件的规模加大、复杂性提高、性能增强

- 软件是逻辑产品，尚未完全认识其本质和特点

- 缺乏有效的、系统的开发、维护大型软件项目的技术手段和管理方法

- 用户对软件需求的描述和软件开发人员对需求的理解往往存在差异，用户经常要求修改需求，开发人员很难适应

- 软件开发的技术人员和管理人员缺乏软件工程化的素质和要求，对工程化的开销认识不足

2. 软件危机包含哪两方面的问题？

一、如何开发软件，以满足不断增长，日趋复杂的需求；

二、如何维护数量不断膨胀的软件产品。

3. 软件是什么？

计算机系统的重要组成部分；/逻辑产品，需要计算机硬件和系统软件的支撑；/是计算机控制系统的指挥中枢；/是信息转换器，它能对信息进行加工、处理或变换；/是工具，在人们的生活、工作、休闲，在社会的经济、军事、政治、文化、科学技术、教育中发挥巨大作用；

软件=程序+数据+文档

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序，数据及其相关文档的完整集合

4. 软件与硬件的区别？//软件的特点（从和硬件比较的视角）

- 软件是被开发或设计的,而不是传统意义上被制造的

- 软件不会“磨损”，硬件会磨损

- 软件产业逐步走向基于构件的组装,但还是定制的

5. 软件工程最关注的是什么？质量

6. 质量的观点包括哪些？

质量的观点：

1. 超越的观点（Transcendental View）：质量是可以认识而不能定义的

2. 用户的观点（User View）：质量是恰好达到目的

3. 制造的观点（Manufacturing View）：质量要与需求说明一致

4. 产品的观点（Product View）：质量与产品的内在特性相联系

5. 基于价值的观点（Value-based View）：质量取决于用户愿意支付的金额

至少得从三个方面考虑

产品质量（the quality of product）：

产品修改：可维护性，适应性，可测试性

产品变迁：可移植性，可复用性，互用性

产品运行：正确性，可靠性，完整性，可用性，效率性

过程质量 (the quality of process)

过程质量模型

CMM (Capability Maturity Model)

CMMI (Capability Maturity Model integration)

ISO model

商业环境中的质量 (the quality in the context of the business environment)

7. 软件开发的方法？

系统的方法 (定义系统边界 边界指出了包含的内容和不包含的内容)

确定活动和对象 活动：触发器引发的事件和行为，活动将改变事物

对象：活动中涉及的所有要素，系统中对象是有联系的)

考虑嵌套的系统、相互关联的系统

工程的方法 (Requirements analysis and definition, System design, Program design, Writing programs, Unit testing, Integration testing, System testing, System delivery, Maintenance)

8. 软件工程的规范？(8个概念)

Abstraction 抽象：从复杂且多样性的现象得到事物的本质特性

Analysis and design methods and notations 分析、设计方法和符号：标准化的方法和符号避免歧义

User interface prototyping 用户界面原形：帮助用户捕获真实的需求，验证设计或方法的可行性

Software architecture 软件体系结构：解决方案的分解，系统如何分解为若干单元，这些单元间是如何联系的及单位的外部可见性。

Software process 软件过程：保证了系统的可控性

Reuse 复用

Measurement 度量：必须对目标和完成工作进行量化

Tools and integrated environments 工具和集成环境

9. 工程化的两个重点？

1、“科学和数学的**应用**，通过这种应用将自然界中的物质属性和能量源变得对人类有用”

2、工程化的基本概念是以**可重复**的一致的方式来解决

10. 计算机科学和软件工程区别？

计算机科学：研究计算机相关的理论

软件工程：把计算机技术看作工具，用以解决问题

补充：

11. 软件是什么？

计算机系统的重要组成部分；/逻辑产品，需要计算机硬件和系统软件的支撑；/是计算机控制系统的指挥中枢；/是信息转换器，它能对信息进行加工、处理或变换；/是工具，在人们的生活、工作、休闲，在社会的经济、军事、政治、文化、科学技术、教育中发挥巨大作用；

软件=程序+数据+文档

软件是计算机系统中与硬件相互依存的另一部分，它是包括程序，数据及其相关文档的完整集合

12. 改变工程实践的七个关键因素

1. 交付时间的重要性
2. 计算行业经济的变化趋势
3. 功能强大的桌面运算
4. 互联网络
5. 面向对象的技术
6. 图形用户界面
7. 瀑布模型的不可预知性

13. 软件工程定义（两种）

1：软件工程 Fritz Bauer：为了经济的获得可靠的，在实际机器上高效运行的软件，而建立和使用的好的工程原则。

2：软件工程[IEEE93]：软件工程是（1）将系统的、规范的、可度量的方法应用于软件的开发、运行和维护的过程。即工程化方法应用于软件。（2）在（1）中所述方法的研究

第二章 软件过程

1. 过程是什么？过程的特征/特点

--过程是指为了达到给定目的而执行的实践的集合；它可能包括工具、方法、资料和/或人。

--过程是指为了达到给定目的而执行的一系列活动的有序集。

过程的特征：

- all major process activities 所有主要过程活动
- resources used, subject to set of constraints 资源占用、服从于一组约束(such as schedule)
- intermediate and final products 中间和最终产品
- subprocesses, with hierarchy or links 分层或链接的子过程
- entry and exit criteria for each activity 每个活动的入口和出口标准
- sequence of activities, so timing is clear 活动的顺序，活动期是很清楚的
- guiding principles, including goals of each activity 指导原则，包括每个活动的目标
- constraints for each activity, resource or product 对每一个活动、资源或产品的约束

2. 传统的软件开发生命周期？什么是软件过程？

软件过程又称软件生命周期（Life cycle），是软件生命周期内为达到一定目标而必须实施的一系列相关过程的集合。软件过程将各个技术层次结合在一起，并实施合理地、及时地开发计算机软件

软件生命周期过程包括：

早期：立项、需求分析、设计、编码、测试、交付、维护、退役

现在：在早期的基础上又加入了：管理各种活动、质量保证、环境基础设施配置、文档管理等。

三个时期、七个阶段

△软件定义（可行性研究和计划，需求分析）

问题定义：确定系统的总体目标

可行性分析：研究经济、技术、操作等的可行性

需求分析：收集需求，需求建模

△软件开发（概要设计/总体设计，详细设计，实现/编码，集成测试，确认测试）

系统设计：软件结构设计、数据设计、接口设计和过程设计

编码：产生源程序清单

测试：产生软件测试计划和软件测试报告

△软件运行（维护时期：使用和维护）

维护：修改、完善、扩展软件

3. 软件通用过程框架？（可适用于绝大多数的软件项目的过程框架：沟通-策划-建模-构建-部署）

沟通：这个框架活动包含了与客户（包括所有共利益者 Stakeholder）之间大量的交流协作，还包括需求获取以及其他相关活动。

策划：为后续的软件工程工作制定计划。它描述了需要执行的技术任务、可能的风险、资源需求、工作产品和工作进度计划

建模：包括创建模型和设计两方面。创建模型有助于客户和开发人员更好地理解软件需求，设计可以实现需求。

构建：包括编码和测试

部署：软件交付给用户，用户对其进行评测并给出反馈意见

4. 软件过程各个阶段内容

可行性研究：

任务：了解用户要求和现实环境。从技术、经济、市场等方面研究并论证开发该软件系统的可行性。

△技术可行性：当前的软件开发方法和工具能否支持需求的实现；

△操作可行性：用户能否在特定的环境下使用这个软件；

△经济可行性：开发和使用、维护这个软件的成本能否被用户所接受。

工作产品：可行性报告

需求分析：

任务：确定用户对软件系统的需求：

△功能需求：软件必须要完成的功能；

△性能需求：软件的安全性、可靠性、可维护性、精度、错误处理、适应性、用户培训等；

△运行环境约束：待开发的软件产品必须满足的环境要求

过程：

△需求分析人员必须与用户不断、反复地交流和商讨，使用户需求逐步准确、一致、完全。

△方法：结构化分析方法、面向对象的分析方法等

△工具：Rational Rose 等

工作产品：△软件需求规格说明书 SRS △用户需求定义文档。

概要设计：

任务：根据 SRS 建立目标软件系统的总体结构、设计全局数据库和数据结构，规定设计约束，制定组装测试计划等等。

原则：△坚持功能模块内部高内聚，功能模块之间松耦合的 △坚持与需求规格说明书的一致性

方法：△结构化方法 △面向对象方法

工作产品：概要设计规格说明书，数据库或数据结构设计说明书，集成测试计划

详细设计：

任务：细化概要设计所生成的各个模块，并详细描述程序模块的内部细节(算法，数据结构等)，形成可编程的程序模块，制订单元测试计划

工作产品：详细设计规格说明书，单元测试计划

实现：

任务：根据详细设计规格说明书编写源程序，并对程序进行调试、单元测试、系统集成，验证程序与详细设计文档的一致性

方法：以详细设计规格说明书为依据、基于某种程序设计语言进行编码

结构化程序设计/面向对象程序设计

工作产品：源程序代码

集成测试：(概要设计)

任务：组装测试应满足概要设计的要求。

途径：测试模块连接的正确性；测试系统或子系统的 I/O；测试系统的功能和性能。

工作产品：满足概要设计要求的程序、组装测试报告

单元测试：(详细设计)

任务：对模块进行测试

途径：黑盒测试、白盒测试

工作产品：单元测试报告

确认测试：

任务：根据软件需求规格说明书，测试软件系统是否满足用户的需求

方法：用户参与，以软件需求规格说明书为依据进行确认测试

工具：专用测试工具

工作产品：可供用户使用的软件产品(文档，源程序)

使用和维护

任务：软件工作环境不断变化，软件也必然跟着变化，软件必须不断进化以满足客户的需求变化，这是软件产品最根本的特性。

软件维护占用软件开发 60%以上的工作量。

正确性维护

扩充性维护

适应性维护

软件产品的新版本

5. 瀑布模型

软件开发过程与软件生命周期是一致的，相邻二阶段之间存在因果关系，需对阶段性产品进行评审

优点：

1. 软件生命周期模型,使软件开发过程可以在分析、设计、编码、测试和维护的框架下进行
2. 软件开发过程具有系统性、可控性，克服了软件开发的随意性

缺点：(瀑布模型到后端没有反馈，瀑布模型线性阻塞，人的认知是螺旋上升的)

1. 项目开始阶段用户很难精确的提出产品需求，由于技术进步，用户对系统深入的理解，修改需求十分普遍。
2. 项目开发晚期才能得到程序的运行版本，这时修改软件需求和开发中的错误代价很

大。

3. 采用线性模型组织项目开发经常发生开发小组人员“堵塞状态”，特别是项目的开始和结束。

6. V 模型：软件开发过程与测试相呼应

7. 快速原型模型 原型使用→理解用户需求

- 用户/客户给出软件产品的一般需求
- 开发小组和用户共同定义软件总体目标，标识已知需求
- 对界面、功能、人机交互方式等，进行设计并建造原型
- 强调“快速”，采用基于构件的软件开发方法,尽量缩短软件开发周期,不宜采用过多的新技术
- 用户/客户对原型进行评估
- 修改需求、更新设计、完善原型直至确定需求。

注意：原型可抛弃也可不抛弃。

优点：

- 原型模型支持软件需求开发，帮助用户和开发人员理解需求，是软件需求工程的关键。
- 它产生的正式需求文档，是软件开发的基础。
- 如果开发的原型是可运行的，它的若干高质量的程序片段和开发工具可用于工作程序的开发。
- 原型的开发和评审是系统分析员和用户/客户共同参予的迭代过程，每个迭代循环都是线性过程。

8. RAD 模型（快速应用开发模型）：侧重于短暂开发周期，多个团队可并行进行下面的工作

业务建模，数据建模，过程建模，应用生成，测试修正

缺点

- 对于大型软件项目，需要足够的人力资源以建立多个相对独立的 RAD 团队。
- 如果开发者和客户没有短时间内没有为急速完成整个系统做好准备，RAD 项目将会失败
- 如果系统难以模块化，建造原型所需构件就有问题
- 如果高性能是一个指标，原型模型也可能不奏效。
- 技术风险很高的情况下不宜采用该模型

9. 螺旋模型（适用于计算机软件整个生命周期）

螺旋模型=线性模型+迭代原型+系统化

优点：

1. 符合人们认识现实世界和软件开发的客观规律
2. 支持软件整个生命周期
3. 保持瀑布模型的系统性、阶段性
4. 利用原型评估降低开发风险
5. 开发者与用户共同参与软件开发，尽早发现软件开发过程中的错误
6. 不断推出和完善软件版本，有助于需求变化，获取用户需求，加强对需求的理解

10. 增量模型中的增量是什么？

增量：小而可用的软件

•特点：在前面增量的基础上开发后面的增量，每个增量的开发可用瀑布或快速原型模型，迭代的思路

14. RUP 是什么？包括哪几个阶段？（Rational Unified Process）

用例驱动、以体系结构为核心、迭代及增量

RUP 以适合于大范围项目和机构的方式捕捉了许多现代软件开发的最佳实践

Rational Unified Process 有四个阶段：

先启：定义整个项目的范围（确定开发的目标和范围，定义主要的需求分析，构建一个基本架构，估算开发周期和成本，估计潜在风险，用例建模）

精化：制定项目计划、描述功能、建立体系架构框架和可执行的“体系结构基线”

构建：构造软件产品（迭代实现核心架构并降低高风险的阶段）

产品化：将软件产品移交到最终用户手中

迭代是一个基于确定计划和评估标准并且产生一个可执行发布版（内部的或外部的）的独特活动序列。

15. 什么是软件体系结构基线？

1. 可以运行的真实的框架或软件
2. 你如何构建，你项目最核心的功能=》核心用例

补充：

16. 如何看待过程模型？它给我们带来什么好处？（答案自拟）

一种针对软件过程的各个阶段提供的一种开发策略，使得软件开发过程变得可控从而达到预期目的。

第三章 敏捷开发

1. 敏捷开发的核心观点是什么？

人最重要（强调人是获取成功最为重要的因素）（以人为本，为客户提供尽早可交付的软件）

2. 对于敏捷开发什么是最重要的？

可以交付给客户的产品以及帮助客户实现最大的价值

3. 敏捷联盟宣言是什么？（给实例对应哪一条）

个体和交互	胜过	过程和工具
可以工作的软件	胜过	面面俱到的文档
客户合作	胜过	合同谈判
响应变化	胜过	遵循计划

个体和交互胜过过程和工具（人是获得成功的最为重要的因素，合作、沟通以及交互能力要比单纯的编程能力更为重要。）

虽然右项也有价值，但我们认为左项具有更大的价值

4. 极限实践编程（eXtreme Programming）

由一系列简单却相互依赖的实践组成。

5. Xp 模式中迭代又叫什么？

计划游戏，为了开发人员进行愉悦的开发，像玩游戏一样

6. Xp 模式中 2 周一次反应了什么？

响应变化：响应变化大于逻辑计划

补充：

7. 如何看待敏捷中的个体与交互胜于过程和工具？

人是获得成功的最为重要的因素。合作、沟通以及交互能力要比单纯的编程能力更为重要。我们根据实际情况来选择所用的过程。

8. Xp 中为什么要由完整的团队？（自拟）

面对面交流，工作在开发的场所，可以提高工作效率。

9. XP 的四个要点

完整团队，计划游戏，测试驱使开发，重构

第四章 项目管理与计划

1. 管理的五要素（计划、组织、指挥、协调、控制）

2. 拿到项目第一步是什么？项目分解（具体看笔记）

3. 为什么要用 WBS/WBS 用途（软件的成本、进度、质量、顺利完成需估算，都需要依赖于 WBS 做项目管理）

- 确定工作范围
- 配备人员
- 编制资源计划
- 监视进程
- 明确阶段里程碑
- 内容的验证

4. 三种常见的程序设计小组的组织形式，具有不同的通信路径数

- 核心程序员制小组(高结构化)
- 民主制小组(松散结构)
- 层次式小组(高结构化和松散结构两者之间)

组织机构的比较：

高度结构化	松散结构
高确定性	不确定
重复	新技术或工艺
大项目	小项目

5. 规模估算从什么开始？：软件分解（wbs 最好不超过 7 层）

6. 定量的标准是什么？

LOC 估算（代码行）和 FP 估算（功能点）

LOC 优缺点：

优点：

用软件代码行数估算软件规模简单易行。

缺点：

- 代码行数的估算依赖于程序设计语言的功能和表达能力；
- 采用代码行估算方法会对设计精巧的软件项目产生不利的影响；
- 在软件项目开发前或开发初期估算它的代码行数十分困难；
- 代码行估算只适用于过程式程序设计语言（C,Pascal），对非过程式的程序设计语言不太适用等等。

7. FP (功能点估算) 功能点分析法是可在需求分析阶段基于系统功能的一种规模估算方法，它的运用可以贯穿于整个软件生命周期。是基于应用软件的外部、内部特性以及软件性能的一种间接的规模测量

功能点分析的一个主要的目标就是从用户的角度定义系统的能力

8. FP 功能点分类：

数据类型功能点：内部逻辑文件，外部接口文件

人机交互类型功能点：外部查询，外部输入，外部输出=》都是逻辑处理

FP 优缺点：

优点：

- 与程序设计语言无关，它不仅适用于过程式语言，也适用于非过程式的语言；
- 软件项目开发初期就能基本上确定系统的输入、输出等参数，功能点度量能用于

软件项目的开发初期。

缺点

- 它涉及到的主观因素比较多，如各种权函数的取值；
- 信息领域中的某些数据有时不容易采集；
- FP 的值没有直观的物理意义。

9. FP 估算和 LOC 估算的比较

- 代码行度量依赖于程序设计语言，而功能点度量不依赖于程序设计语言。
- FP 法需要某种“人的技巧”，因为计算是基于主观的而非客观的数据；
- 两者可以结合使用。例如：在实践应用中，在全生命周期采用 FP 的同时，在项目内部局部结合使用 LOC 作为补充手段
- Albrecht 和 Jones 等人对若干软件采用事后处理的方式分别统计出不同程序设计语言每个功能点与代码行数的关系，用 LOC/FP 的平均值表示。

10. EI、EO、EQ、ILF、EIF？

外部输入 EI 是指一个基本处理，它处理来自本应用边界之外的一组数据或者控制信息，外部输入的基本目的是为了维护一个内部逻辑文件或者改变系统的行为。

外部输出 EO 是指一个向应用边界之外发送数据或者控制信息的基本处理。外部输出的基本目的是为了向用户展示一组经过了除了提取之外的其他逻辑处理的数据或者控制信息。这里的其他逻辑处理包括至少一个数据演算或者对衍生数据的生成。外部输出可能包括对内部逻辑文件的维护或者对系统行为的改变。

外部查询 EQ 是指一个向应用边界之外发送数据或者控制信息的基本处理。外部查询的基本目的是为了向用户展示提取的数据或者控制信息。外部查询里面不包含数学公式或者计算以及对衍生数据的生成。外部查询不会维护内部逻辑文件，不会引起系统行为的改变。

内部逻辑文件是指一组以用户角度识别的，在应用程序边界内且被维护的逻辑相关的数据或者控制信息。内部逻辑文件的主要目的是通过应用程序的一个或者多个基本处理过程来维护数据。

外部接口文件是指一组在应用程序边界内被查询的，但它是在其他应用程序中被维护的，以用户角度来识别的，逻辑上相关的数据。一次一个应用程序中的外部接口文件必然是其他应用程序的内部逻辑文件。外部接口文件的主要目的是为边界内的应用程序提供一个或多个通过基础操作过程来引用的一组数据或信息。

11. 成本估算估算类型：专家判定、算法模型、机器学习

12. 估算模型的评估模型：

MMRE 相对误差平均值，期望越小越好，目标低于 0.25

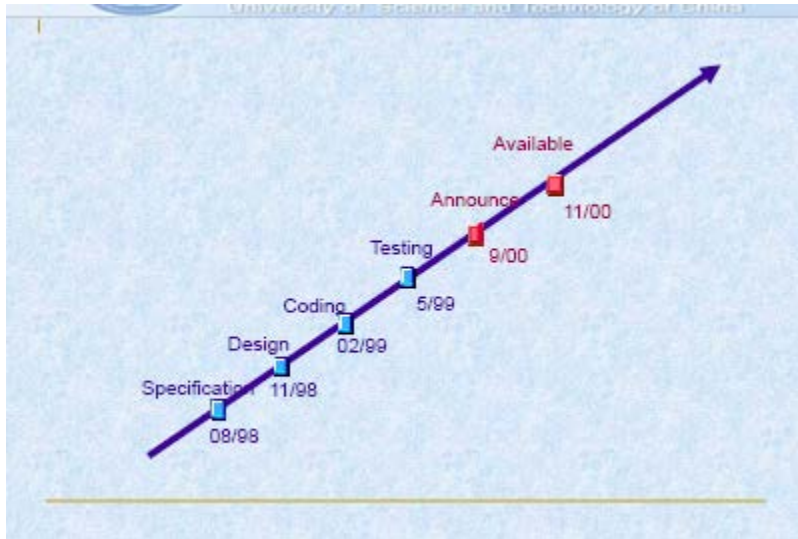
Pred(x/100): 估计在实际值 x%范围内的项目的百分比，对离散进行限制越大越好，目标是高于 0.75

13. 计划评审技术 (PERT)、图形评审技术 (GERT) 和 CPM (关键路径)

计划评审技术：利用经加权平均的所需时间估算，计算各项活动所需时间。PERT 同 CPM

(关键路径)的主要差别在于 PERT 使用期望值来估算时间,而不使用 CPM 所用的最大可能估计。PERT 有不确定,持续时间不确定,活动时间的关系也不确定的。

14. 软件开发进度的里程碑是什么？



需求规格说明, 设计, 编码, 测试, 发布, 维护

15. 风险是什么？

在给定情况下和特定时间内, 那些可能发生的结果与预期结果之间的差异, 差异越大, 风险越大 (现实与期望的差异)

八、 风险管理

风险

在给定情况下和特定时间内, 那些可能发生的结果与预期结果之间的差异, 差异越大, 风险越大

- 风险后果: 与该事件有关的损失
- 风险概率: 事件发生的可能性
- 风险控制: 我们能改变结果的程度

风险成本 = 风险概率 × 风险后果

16. 风险管理的 4 个活动？

风险管理就是识别评估风险, 建立、选择、管理和解决风险的可选方案和组织方法

—包括了风险标识 (潜在地风险列表)、风险预测(优先级高的风险列表)、风险评估 (风险规划和应急计划, 风险评估体系, 定义风险的参考水准, 建立三元组) 和风险管理 with 监控 (风险评估) 四个活动

—强调通过对项目目标的主动控制做到防患于未然以避免或减少损失

17. QC 和 QA

质量控制 (QC)

为保证每一件工作产品都满足需求而应用于整个开发周期中的一系列审查、评审和测试

- 质量控制在创建工作产品的过程中包含一个反馈循环，通过对质量的反馈，使得能够在得到的工作产品不能满足其规约时调整开发过程

- 所有工作产品都应该具有定义好的和可度量的规约，这样就可以将每个过程的产品与这一规约进行比较

质量保证（QA）

软件质量保证（SQA）是建立一套有计划，有系统的方法，来向管理层保证拟定出的标准、步骤、实践和方法能够正确地被所有项目所采用。

QA 保证 QC 行为与措施符合，同时汇总后报告给管理层

18. 软件质量控制

采用技术手段保证软件质量/组织技术评审/加强软件测试/推行软件过程标准/对软件的修改、变更进行严格控制/对软件质量进行度量

19. 什么是配置管理？为什么要进行配置管理？主要是为了解决什么问题？ 减少混乱，关键字 混乱

软件配置定义：软件配置是由在软件工程过程中产生的所有信息项构成的，它可以看作该软件的具体形态（软件配置项）在某一时刻的瞬间影像。

软件配置管理：协调软件开发使得混乱减到最小的技术叫做软件配置管理，它是一种标识、组织和控制修改的技术，目的是使错误达到最小并最有效地提高生产效率

软件配置管理的特点是贯穿整个软件生命周期与软件工程过程

软件配置管理的目标：标识变更，控制变更，确保变更，报告变更

补充：

项目管理：项目管理，就是项目的管理者，在有限的资源约束下，运用系统的观点、方法和理论，对项目涉及的全部工作进行有效地管理。

项目分解的结果：WBS 图（任务分解结构图）

WBS 通常是一种面向“成果”的“树”，其最底层是细化后的“可交付成果”，该树组织确定了项目的整个范围。但 WBS 的形式并不限于“树”状，还有多种形式。

WBS 的目的：WBS 主要是将一个项目分解成易于管理的几个部分或几个细目，以便确保找出完成项目工作范围所需的所有工作要素它是一种在项目全范围内分解和定义各层次工作包的方法。

项目人员 5 类：

项目管理人员：负责软件项目的管理工作，其负责人通常称为项目经理

高级管理人员：可以是领域专家，负责提出项目的目标并对业务问题进行定义

开发人员：掌握了开发一个产品或应用所需的专门技术，可胜任包括需求、分析、设计、编码、测试、发布等各种相关的开发岗位

客户：一组可说明待开发软件的需求的人，也包括与项目目标有关的其它风险承担者

最终用户：产品或应用提交后与产品/应用进行交互的

可靠性估算：错误植入法，分别测试法，软件平均故障间隔时间估算

高质量的软件应该基本哪些条件：（用户观点，制造观点，产品观点）（详细见课件）

第五章 需求工程：

1.需求工程的重点？

挖掘用户的真实需求，保证这些需求能真实的无歧义的表现出来

2.获取需求的四个阶段？

获取需求的过程可分为四个阶段：导出需求、分析建模、规格说明、需求确认和校验，最终的工作产品软件需求规格说明（SRS Software Requirements Specification）

3.导出需求做哪些工作？工作产品是什么？

University of Science and Technology of China

2.1 导出需求

需求导出是整个需求工程中极为重要的一环，有多种技术和方法来用来确定用户和顾客的想法。

- 需求启动
 - 确定利益共同者（Stakeholder）
 - 识别多种观点
 - 协同合作
 - 首次会议
 - 初步“产品要求”文档
- 需求协同收集
 - 准备列表
 - 召开评审会议
 - 初步需求说明文档

需求说明文档

4.需求启动做什么工作？

5.导出需求的手段方法？

6.mini specification 作用？初步的规格说明(Mini-specification)

根据每个项的 Mini-specification 生成产品或项目初步的需求说明文档

7.为何要进行分析建模？

- 建立分析模型。从不同的角度、不同的抽象级别精确地说明对问题的理解、对目标软件的需求。
- 模型可帮助用户和分析人员发现、排除用户需求不一致，不合理的部分，挖掘潜在的用户需求。
- 模型是分析人员根据初步导出的需求而创建的软件系统结构，包括相关的信息流、处理功能、用户界面、行为及设计约束。
- 模型是形成需求规格说明、进行软件设计的基础。

分析模型的目的是提供必要的信息、功能和行为域的说明。模型应能动态的改造，以便于软件工程师更多地了解将要实现的系统，以便于共同利益者更多的了解他们到底需要什么。分析模型是任意给定时刻的需求的快照。


如何构建分析模型将在后面课程中详细讨论

8.SRS 内容？

软件需求规格说明文档

规格说明对于不同的人有不同的含义。一个规格说明可以是一份写好的文档、一套图形化的模型、一个形式化的数学模型、一组使用场景、一个原型或上述各项的任意组合。规格说明是需求分析人员的最终工作产品。

9.需求确认和需求校验的区别？



中国科学技术大学
University of Science and Technology of China

2.4 需求确认和校验

需求确认
需求确认的目的是用来检查获得的需求定义是否准确的反应了用户的实际需求。

需求校验
需求校验的目的是用来检查需求规格说明文档和需求定义文档是否一致。

补充：

1.需求工程要解决什么问题？

用户的真实的需求，统一的建模工具、语言、符号、不产生歧义。

2.需求的主要工作产品：

需求定义文档（客户角度），需求规格说明（技术人员角度），配置管理文档（支持两文档间直接对应）

3.在需求导出部分，要做什么事？

（1）需求启动 确定利益共同者，识别多种观点，协同合作，首次会议，初步“产品要求”文

档

(2) 需求协同收集：准备列表 召开评审会议 初步需求说明文档

第六章：

1.结构化分析建模包括那些内容？

二、结构化分析方法概述

- 1、什么是结构化分析方法？
 - 结构化分析方法是一种建模技术
 - 基于计算机的系统是数据流和一系列的转换构成的
 - 在模型的核心是数据词典，它描述了所有的在目标系统中使用的和生成的数据对象。围绕这个核心的有三种图：ERD、DFD、STD

2、结构化分析模型的组成

- 数据建模和对象描述
- 功能建模和数据流程图
- 基本加工逻辑说明
- 行为建模
- 数据词典

2.如何得到 DFD 图？

三、结构化分析模型

- 数据字典是模型的核心，包含软件使用和产生所有数据的描述。
- DFD数据流程图：用于功能建模，描述系统的输入数据流如何经过一系列的加工变换逐步变换成系统的输出数据流。
- ER实体关系图：用于数据建模，描述数据字典中数据之间的关系。
- STD状态迁移图：用于行为建模，描述系统接收哪些外部事件，以及在外部事件的作用下的状态迁移情况。

3.DFD 顶层是什么？

补充：结构化分析方法，主要关注是什么？过程

结构化分析方法：以过程为核心，逻辑和数据是可分离的，数据在逻辑推动下变化的过程，数据推动逻辑变化，最终找到问题的解。

第七章：

1.什么是抽象？

2.为何要抽象？



3.抽象后能得到什么？

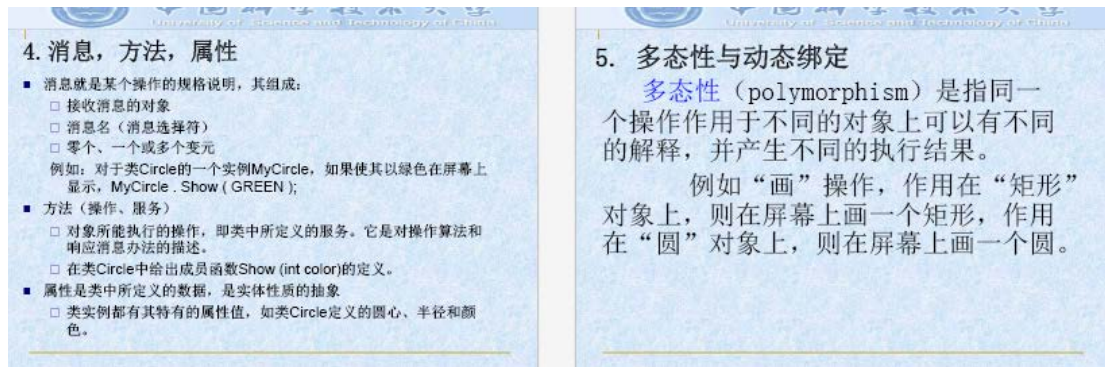
抽象层次树

4.抽象层次树上的两个节点能进行比较？为何？

同一层次上可以比较

5.什么是继承、多态、泛化？

 中国科学技术大学 University of Science and Technology of China	 中国科学技术大学 University of Science and Technology of China
<p>1. 对象 (object)</p> <p>对象是指一组属性以及这组属性上的专用操作的封装体。</p> <p>属性 (attribute) 通常是一些数据, 有时它也可以是另一个对象。每个对象都有它自己的属性值, 表示该对象的状态。对象中的属性只能通过该对象所提供的操作来存取或修改。</p> <p>操作 (operation) (也称方法或服务) 规定了对象的行为, 表示对象所能提供的服务。</p>	<p>封装 (encapsulation) 是一种信息隐蔽技术, 用户只能看见对象封装界面上的信息, 对象的内部实现对用户是隐蔽的。</p> <p>封装的目的是使对象的使用者和生产者分离, 使对象的定义和实现分开。</p>
<p>2. 类 (class)</p> <p>在对系统的分析中可通过抽象可以到类, 在设计和实现中, 类具体表现为一组具有相同属性和相同操作的对象的集合。一个类中的每个对象都是这个类的一个实例。</p> <p>在面向对象的实现中, 类是创建对象的模板, 从同一个类实例化的每个对象都具有相同的结构和行为。</p>	
<p>3. 继承 (inheritance)</p> <p>继承表明了两个类在抽象层次树上的关系。在设计和实现中, 通过继承可以实现不同类共享数据和操作。</p> <p>父类中定义了其所有子类的公共属性和操作, 在子类中除了定义自己特有的属性和操作外, 可以继承其父类 (或祖先类) 的属性和操作, 还可以对父类 (或祖先类) 中的操作重新定义其实现方法。</p> <p>继承可以实现代码的重用。</p>	



6. 面向对象中如何使这些对象进行交互?
通过建立动态模型

补充:

对象是有层次的, 抽象层次理论是 oo 中许多内容的理论基础。

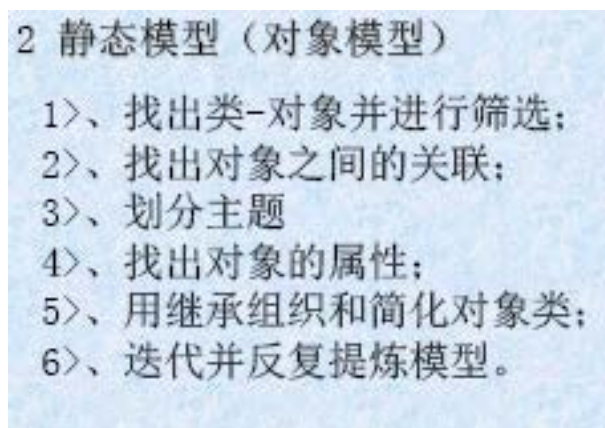
从哲学和认识论的角度来看: 先有对象->后有类; 先有子类后父类是一种自底向上形成的体系。

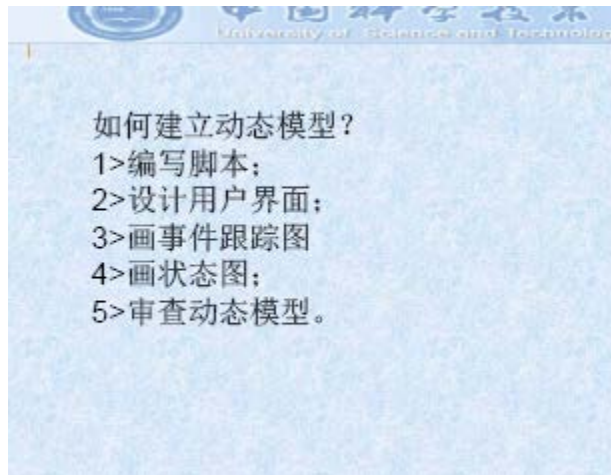
第八章:

1. 面向对象分析模型应该包括哪些内容?

需求描述 静态模型 (对象模型)、动态模型 (交互次序)、功能模型 (数据交换)

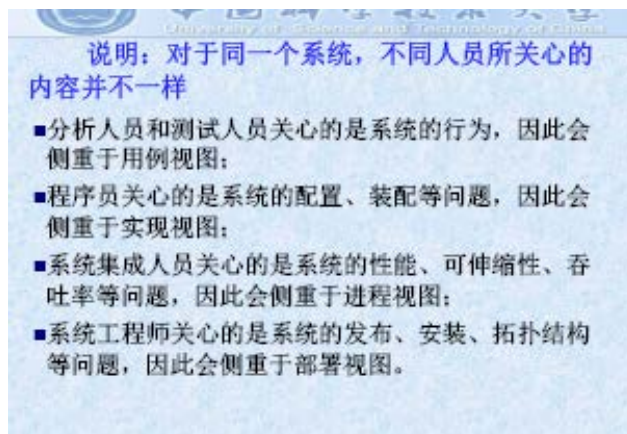
2. 静态模型分别包括哪些内容?





3. UML 的五种视图？**程序员最关注的哪种视图？**

用例视图、逻辑视图、实现视图、进程视图、配置视图



第九章：

1.概念设计的核心？

功能分解

2.什么是概念设计与技术设计？分别做什么工作？

概念设计:告诉顾客系统将要做什么

中国科学技术大学
University of Science and Technology of China

概念设计的任务与步骤

概念设计的过程：

- (1) 设想可能的方案
- (2) 选取合理的方案
- (3) 推荐最佳方案
- (4) 功能分解
- (5) 设计软件结构
- (6) 数据库设计
- (7) 制定测试计划
- (8) 编写文档
- (9) 审查与复审

概念设计确定：

- 软件系统的结构
- 各模块功能及模块间联系(接口)

技术设计:告诉程序员系统做什么

技术设计主要任务

- 编写技术设计说明书：
 - 确定每个模块的算法，用工具表达算法的过程，写出模块的详细过程性描述。
 - 确定每一模块的数据结构。
 - 确定模块接口细节。
- 技术（详细）设计是编码的先导。

3.有哪几种耦合？

内容耦合、公共耦合、控制耦合、标记耦合、数据耦合、非直接耦合


4.有哪几种内聚？

功能性内聚、顺序内聚、通讯内聚、过程内聚、时间内聚、逻辑内聚、偶然内聚

5 设计时可以通过什么途径来降低耦合？

降低接口复杂、选择合理调用方式

6.什么是扇入扇出，深度是什么？




中国科学技术大学
University of Science and Technology of China

深度、宽度、扇入、扇出适中

- 深度：软件结构中控制的层数。一般而言它与系统的复杂度和系统大小直接对应。
- 宽度：软件结构中同一个层次上的模块总数的最大数。
- 扇出：一个模块直接控制（调用）的模块数目。扇出过大说明模块过分复杂；过小也不好，不利于系统平衡分解，3到9为宜。
- 扇入：一个模块的扇入是指直接控制该模块的模块数目。扇入越大说明共享该模块的上级模块越多。
- 整个系统结构呈现“椭圆外型”。

7.什么是作用域，控制域？



中国科学技术大学
University of Science and Technology of China

模块的作用域应该在控制域之内

- 控制域：控制范围，是包括模块本身以及所有下属模块（直接调用模块和间接调用模块）的集合。模块D的控制域为D,G,H,I,J,K。
- 作用域：作用范围，它是一个与条件判定相联系的概念。是受该模块内一个判定影响的所有模块的集合。如果模块D中有一个条件判定仅仅影响到模块G和H，其作用域为G, H, I, J, K，作用域在控制域内。如D的判定影响到E，通常需要在D中为判定结果设置一个标记，并把这个标记通过上级模块A传递给E，导致控制耦合。
- 两种改进方法：判定上移和在作用域但不在控制域的模块下移。

补充：

什么是耦合？（在开发中应降低耦合，但不可以取消）



泛化耦合：

由于泛化（继承）关系的存在，在两个有祖孙、父子关系的类间形成的一种逻辑关联。

聚合：

一种弱的拥有关系，体现A对象可以包含B对象，但B对象不是A对象的一部分。

组合：

一种强的拥有关系，体现了严格的部分和整体的关系，部分和整体具有一样的生命周期。

依赖：

由于逻辑上相互协作可能，而形成的一种关系。

汽车和交通工具属于泛化耦合，轮子和方向盘组合于汽车，汽车聚合成车队，而汽车和司机具有依赖关系。



在设计和实现中，面向对象的方法是一种运用对象、类、继承、封装、聚合、消息传送、多态性等概念来构造系统的软件开发方法。

面向对象=对象(object)

+类(classification)

+继承(inheritance)

+通信(communication with messages)

可以说，采用这四个概念开发的软件系统是面向对象的。

一个对象通常可由对象名、属性和操作三部分组成。

永久对象是指生存期可以超越程序的执行时间而长期存在的对象。

结构化和 oo 在几方面比较：

从概念方面看：

- 结构化软件是功能的集合，通过模块以及模块和模块之间的分层调用关系实现；
- 面向对象软件是事物的集合，通过对象以及对象和对象之间的通讯联系实现；

从构成方面看

- 结构化软件 = 过程 + 数据，以过程为中心；
- 面向对象软件 = （数据 + 相应操作）的封装，以对象为中心；

从运行控制方面看

- 结构化软件采用顺序处理方式，由过程驱动控制；
- 面向对象软件采用交互式、并行处理方式，由消息驱动控制；

从开发方面看

- 结构化方法的工作重点是设计；
- 面向对象方法的工作重点是分析；

WHY?

在结构化方法中，分析阶段和设计阶段采用了不相吻合的表达方式，需要把在分析阶段采用的具有网络特征的数据流图转换进行转换。

从应用方面看

- 结构化方法更加适合数据类型比较简单的数值计算和数据统计管理软件的开发；
- 面向对象方法更加适合大型复杂的人机交互式软件和数据统计管理软件的开发；

UML 九种图的画法（具体看 ppt）

类图、对象图、用例图、顺序图、协作图、状态图、活动图、构件图、配置图（实施图）

	图名称	图定义	图性质
1	类图	一组类、接口、协作及它们的关系	静态图
2	对象图	一组对象及它们的关系	静态图
3	用例图	一组用例、参与者及它们的关系	静态图
4	顺序图	一个交互，强调消息的时间顺序	动态图
5	协作图	一个交互，强调消息发送和接受的对象的结构组织	动态图
6	状态图	一个状态机，强调对象按事件排序的行为	动态图
7	活动图	一个状态机，强调从活动到活动的流动	动态图
8	构件图	一组构件及关系	静态图
9	配置图 (实施图)	一组接点及它们的关系	静态图

UML 主要构成：视图，图，模型元素，通用机制。

UML 有哪些视图？静态表现和动态表现？

 中国科学技术大学 University of Science and Technology of China					
	视图名称	视图内容	静态表现	动态表现	观察角度
1	用户模型视图 (用例视图)	系统行为， 动力	用例图	交互图、 状态图、 活动图	用户、 分析员、 测试员
2	结构模型视图 (设计视图)	问题及解决 方案	类图、对 象图	交互图、 状态图、 活动图	类、 接口、 协作
3	行为模型视图 (进程视图)	性能、可伸 缩性，吞吐 量	类图、对 象图	交互图、 状态图、 活动图	线程、 进程
4	实现模型视图 (实现视图)	构件、文件	构件图	交互图、 状态图、 活动图	配置、 发布
5	环境模型视图 (实施视图)	部件的发 布、交付、 安装	配置图 (实施图)	交互图、 状态图、 活动图	拓扑结构 的节点

用例图三种关系：包含，继承，泛化

类与类之间关系：关联，依赖，组合，聚合，泛化

面向对象的设计的主要工作：

用例实现精化

体系结构设计

构件设计

用户界面设计

数据持久设计

迭代精化

类的设计原则：

SRP 单一职责原则

OCP 开放封闭原则

LSP 里氏替换原则

DIP 依赖倒置原则

ISP 接口隔离原则