软件协同设计课程之

2.1 项目启动与项目章程

2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

1. 项目启动仪式作用

2. 项目启动细节（1/2）

- 产品名称：XXX项目
- 产品目标：
  - The critical product objectives that must be satisfied
  - The optional and desirable objectives
  - The criteria for evaluating the finished product
- 项目人员与责任分配
- 项目启动书实例：报告撰写提示\项目启动书.doc

2. 项目启动细节 (2/2)

- 项目章程
  - 项目经理、小组成员角色与职责。
  - 经费预算
  - 完成时间
  - 项目目标
  - 问题解决机制
- 项目章程范本：报告撰写提示\项目范围与章程.doc

3. 小组会议（1/3）

- 小组会议：
  - 第一次会议：开发周期内，完成的目标，形成统一认识。
  - 小组会议：小组计划、交流、决策
    - 时间：每周一次
    - 地点：教室或实验室。
    - 组织方式与内容：下页
    - 需要填写的表格：（week报告，每周以小组为单位上写入文档）

3. 小组会议：组织方式（2/3）

- 小组人员到齐
- 小组领导：议题、记录、形成报告
- 角色报告：每个角色介绍与自己任务相关的任务和行为现状。
  - 小组领导（项目经理）
  - 用户需求分析经理:用户需求获取、分析、模型描述
  - 开发经理：设计、实现、测试产品
  - 计划经理报告：计划的工作时间、实际的投入时间
  - 质量经理：需求审查、设计审核、每次集成测试方面的数据，可疑的质量问题
  - 测试经理：测试计划执行、测试用例设计、测试记录、测试报告，测试问题反馈
- 小组领导：风险排除，下周的任务安排

3. 小组会议（3/3）

小组会议报告：

- 小组必须一致同意：每个成员每周把项目进展情况，报告给计划经理，
- 小组领导每周给老师一个项目进展报告
  - 计划
  - 完成的部分
  - 未完成的部分，对策

- **小组会议报告模版：** 报告撰写提示\小组会议模板.doc
- **由Team leader维护项目手册：记录项目的关键信息。**

4. 采用什么开发策略

- 采用什么开发策略？
  - 所谓开发策略就是软件开发过程模型，即软件开发过程、方法与工具，根据项目的不同，开发人员可以选用不同的策略
    - 顺序模型
    - 演化模型:evolutionary model
      - incremental model, prototype, spiral model
    - 敏捷开发方法
    - 形式化方法模型：cleanroom method(US)，Vienna Development Method, Z notation(EU)

5. 敏捷开发策略

- 敏捷开发
  - 敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。目标是提高开发效率和响应能力
  - 遵循：沟通、简单（图表）、反馈、勇气、谦逊（不要过度自信）。
- 敏捷开发的原则
  - 快速迭代
  - 让测试人员和开发者参与需求讨论
  - 编写可测试的需求文档
  - 多沟通，尽量减少文档沟通
  - 做好产品原型。用草图和模型来阐明用户界面，但人人都会看图。
  - 及早考虑测试。写测试用例时，可以发现需求与设计中的问题

6. 敏捷开发阅读

- 敏捷开发_百度百科
- http://www.cnblogs.com/taven/archive/2010/10/17/1853386.html
- http://blog.csdn.net/uxyheaven/article/details/49618097

软件协同设计课程之

2.2 项目风险识别与管理

2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

1. 识别风险

- 识别风险的目的
  - 预先采取措施对风险进行控制
  - 例如企业项目：工人不能熟练使用鼠标与键盘录入数据（识别风险）
  - 措施：提前培训
- 软件项目的风险无非体现在以下几个方面：需求、技术、管理、成本、进度、质量

2. 其他风险

- 识别其他风险
  - 与人员与经验相关的风险
  - 产品规模风险（不现实的进度与预算）
  - 外包问题(outsourcing)，外部组件有缺陷
  - 技术风险（实时问题、计算机能力）
  - 客户相关的风险（需求不断变更，参与不充分）
  - 开发环境风险（工具的使用、缺陷）

3. 本项目风险识别

- 本课程项目潜在的风险
  - 组织不利
  - 分组不合理
  - 项目太大或太小
  - 大家太忙
  - 同学之间沟通问题，互有矛盾。
  - 产品过大：先设计内核

3. 项目风险识别

- 本课程项目实施的风险识别
  - 功能复杂：简化/向专家咨询
  - 系统支持问题：小产品进行试验
  - 测试时间：遵循TSP 过程，缺陷应能控制
  - 产品控制：产品提交、改变需要严格的手续
  - 协同工作原则：讨论并达成一致。

4. 风险管理

- 列出风险
- 每种风险的预防措施：尽量预防
- 每种风险的应对措施：一旦发生，给出风险的解决机制

4. 问题与讨论

- 如果同学们组建一个开发团队开发一个项目，针对重要的风险，给出如何预防方法与应对措施

软件协同设计课程之

2.3 软件过程

2025年1月

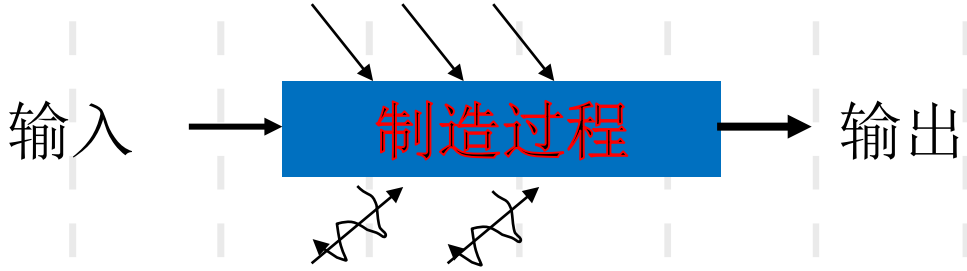
上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

1.从失败中学习到的策略

- 需求工程：需求详尽
- 工程管理：周密的计划：其他项目的经验、估计、质量计划、测试计划
- 质量保障：各阶段进行严格的复审
- 过程控制：整个开发过程的协同工作
- **过程改进**：计划--实施--检查--改进，积累经验

2 过程与过程能力 (1/2)

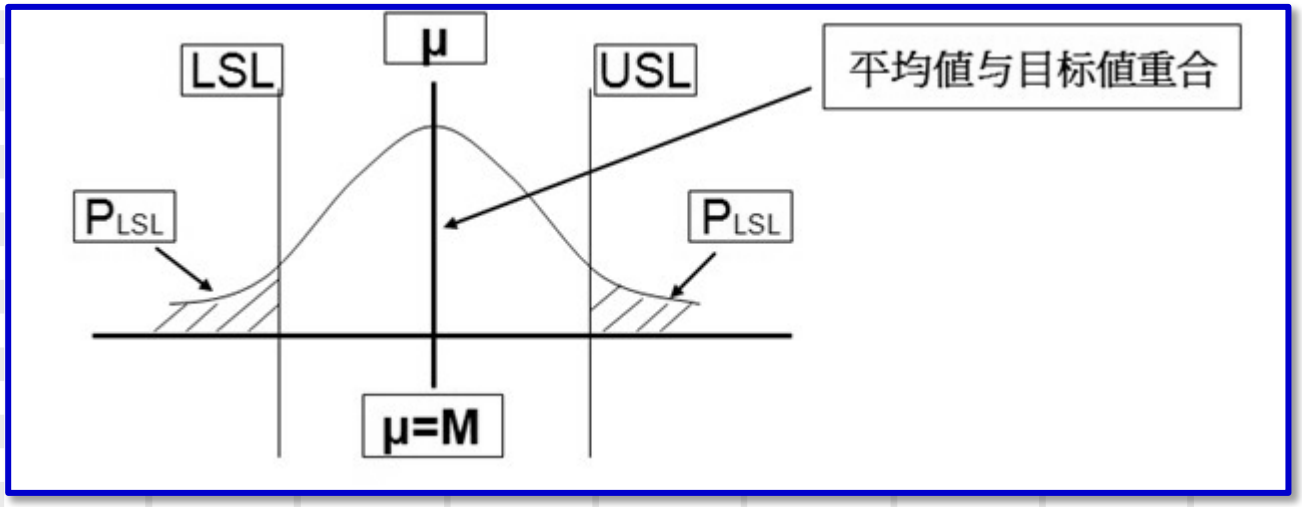
- 制造过程与过程能力



- 过程：将输入转化为输出的系统
- 过程能力高/低如何定义？

2 过程与过程能力(2/2)

- 过程能力cp可以用加工过程得到的输出产品的波动程度σ来度量。



- cp = (USL-LSL)/6σ

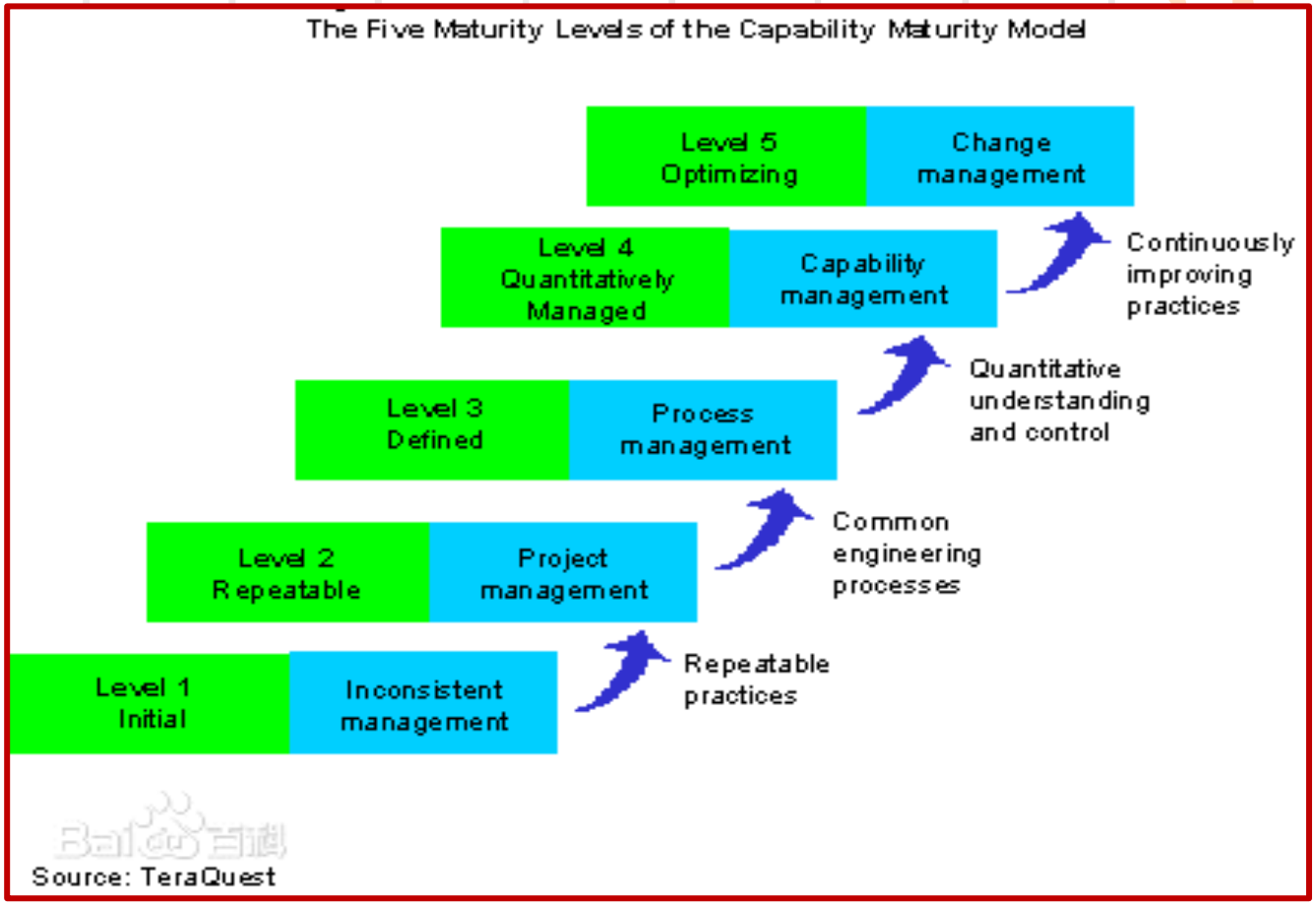
3 软件过程

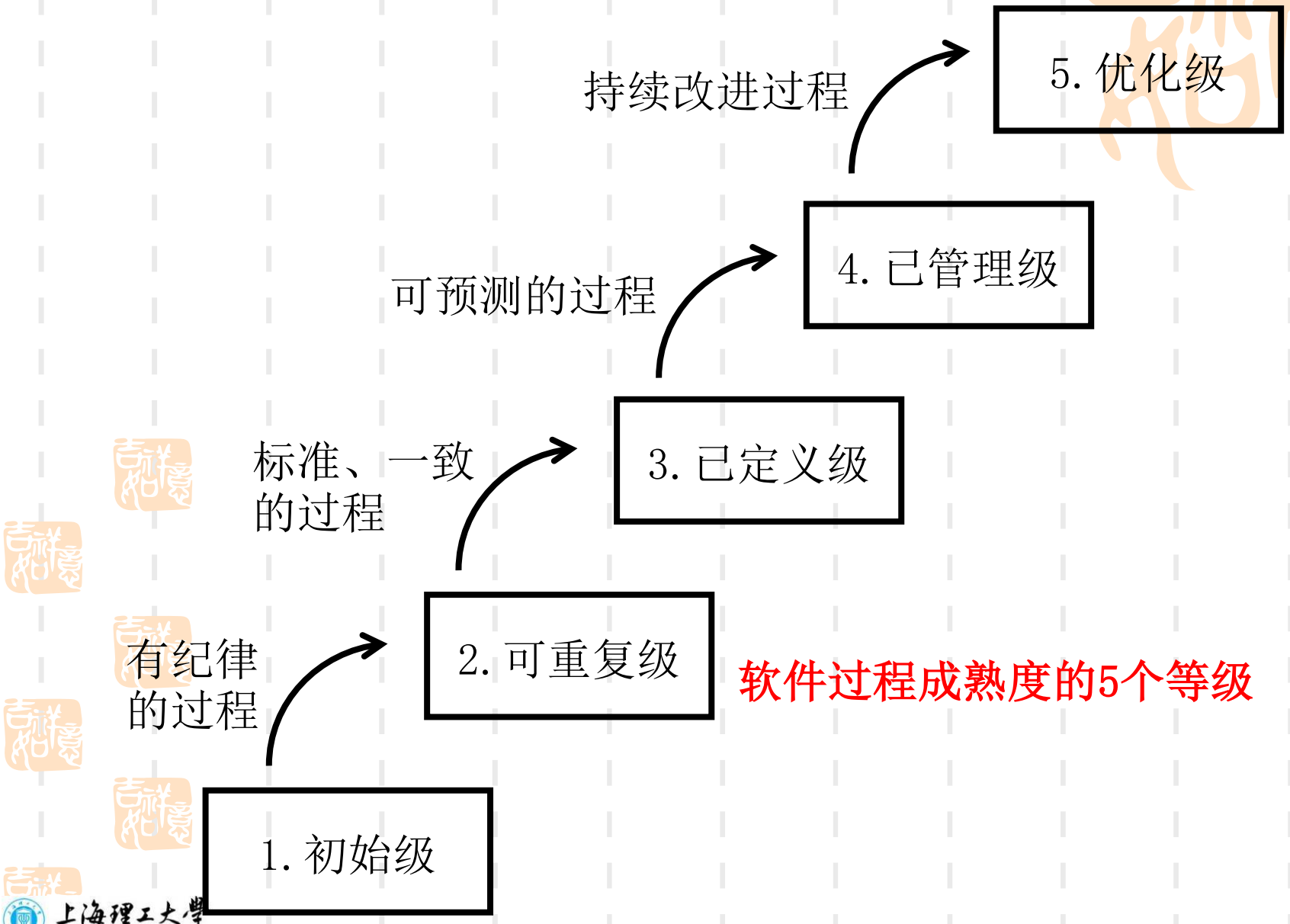
- 第一届软件过程讨论会(1984.10)
  - 在软件生存周期中，所实施的一系列活动的集合，且每个活动可由一些任务组成。任务则起到把输入加工成输出的作用
  - 分为三类：软件开发过程、软件管理过程、软件支持过程
- 软件开发过程:软件需求分析、设计、实施、测试运行、维护
- 管理过程：对软件开发过程的管理，如项目策划、计划、跟踪、质量管理
- 支持过程：技术与管理支持，如评审、培训等

4 软件过程能力度量

- CMM(Capability Maturity Model for Software)
- 故事背景
  - DOD招标时如何选择中标公司?
  - 委托Carnegie Mellon SEI 给出一套度量标准
- CMM主要用于软件过程的改进，不适合于管理过程、支持过程。
- CMMI（Capability Maturity Model Integration)，即能力成熟度模型集成。使之不仅能评价软件过程，还能评价公司的管理过程、支持过程

4 软件过程能力度量


The Five Maturity Levels of the Capability Maturity Model
Level 5
Optimizing
Change management
Level 4
Quantitatively Managed
Capability management
Continuously improving practices
Level 3
Defined
Process management
Quantitative understanding and control
Level 2
Repeatable
Project management
Common engineering processes
Level 1
Initial
Inconsistent management
Repeatable practices
Source: TeraQuest



软件过程成熟度的5个等级

5 软件过程成熟度等级（1/4）

1. 初始 (initial)级

软件过程的特点是无秩序的，甚至是混乱的。几乎没有什么过程是经过妥善定义的，成功往往依赖于个人或小组的努力

2. 可重复(repeatable)级

建立了基本的项目管理过程来跟踪成本、进度和功能特性。制定了必要的过程纪律，能重复早先类似应用项目取得的成功

3. 已定义(defined)级

已将管理和工程活动两方面的软件过程文档化、标准化，并综合成该机构的标准软件过程。所有项目均使用经批准、剪裁的标准软件过程来开发和维护软件

4. 已管理(managed)级

收集对软件过程和产品质量的详细度量值，对软件过程和产品都有定量的理解和控制

5. 优化(optimizing)级

整个组织关注软件过程改进的持续性、预见及增强自身，防止缺陷及问题的发生。过程的量化反馈和先进的新思想、新技术促使过程不断改进

问题与讨论

- CMM与CMMi之间的关系？
- CMM能为软件企业带来什么？

软件协同设计课程之

3.1 软件度量

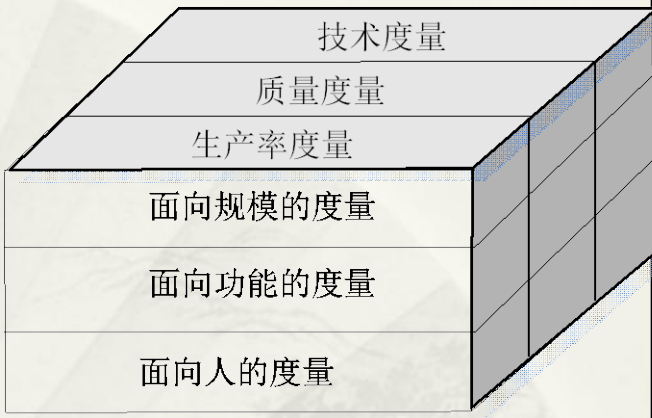
2025年1月

1.为何要度量软件

- 没有软件度量，就不能进行科学分析。
- 目的：对软件开发加以理解、预测、评估、控制和改善。
- 软件度量（software measurement）
  - 项目度量：度量项目规模、项目成本、项目进度、人力等
  - 产品度量：功能、质量、用户满意度
  - 过程度量：获得有关软件过程的数据和问题，并进而对软件过程实施改善

2 软件度量分类

- **面向规模的度量：**度量软件大小
- **面向功能的度量：**利用软件的“功能性”和“实用性”间接度量软件
- **软件质量度量：**可指明软件满足明确的和隐含的用户需求的程度
- **技术度量：**对软件产品的某些特征(如逻辑复杂性、模块化程度)的度量
- **面向人的度量：**有关人们开发计算机软件所用方式、人们理解有关工具的方法和效率等
- **软件生产率度量：**软件工程活动的生产效率



3. 面向规模的度量

- 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
- 测量出软件规模后可方便地度量其他软件属性

度量名	含义及表示
LOC或KLOC	代码行数或千行代码数
生产率P	P=LOC/E，E为开发的工作量(常用人月数表示)
每行代码平均成本C	C=S/LOC，S为总成本
文档代码比D	D=Pe/KLOC，其中Pe为文档页数
代码错误率EQR	EQR=N/KLOC，其中N为代码中错误数

4 面向功能的度量

- 功能特性度量：

  度量“功能性”和“实用性”方面

- 功能点度量：基于软件信息域的特征(可直接测量)和软件复杂性进行规模度量

- 功能点度量方法步骤：
  - 计算信息域特征的CT值
  - 计算复杂度调整值
  - 计算功能点FP

5 功能点度量（1/4）

计算信息域特征的CT值：软件分成五个信息域特征(见下页)，统计相应的特征数，然后根据信息域特征的复杂程度选择适当的加权因子进行计算(下表)，得到总计的CT值

CT计算表

测量参数	特征数	加权因子			结果（=特征数X加权因子）
		简单	中间	复杂	
用户输入数		×3	×4	×6	
用户输出数		×4	×5	×7	
用户查询数		×3	×4	×6	
文件数		×7	×10	×15	
外部接口数		×5	×7	×10	
总计CT					

5 功能点度量（2/4）

五个信息域特征含义

特 征 名	含 义
用户输入数	对每个用户输入进行计数，它们向软件提供不同的面向应用的数据。输入应该与查询分开，分别计数
用户输出数	对每个用户输出进行计数，它们向用户提供面向应用的数据。这时，输出是指报表、屏幕、出错消息等。
用户查询数	一个查询被定义为一次联机输入，它导致软件以联机输出的方式产生实时的响应。每一个不同的查询都要计算
文件数	对每个逻辑上的主文件进行计数（即数据的一个逻辑组合，可能是数据库的一个表或是一个独立的文件）
外部接口数	对所有机器可读的接口（如存储介质上的数据文件）进行计数，利用这些接口可以将信息从一个系统传送到另一个系统

5 功能点度量（3/4）

复杂度调整值是基于对下表中14个问题值，计算调整因子F。对每个问题回答的取值范围是0到5

	问　　题	Fi（0-5）
1	系统需要可靠的备份和恢复吗？	
2	需要数据通信吗？	
3	有分布处理功能吗？	
4	性能很关键吗？	
5	系统是否在一个现存的、重负的操作环境中运行？	
6	系统需要联机数据登录？	
7	联机数据登录是是否需要在多屏幕或多操作之间切换以完成输入？	
8	需要联机更新文件吗？	
9	输入、输出、文件或查询很复杂吗？	
10	内部处理复杂吗？	
11	代码需要被设计成可复用的吗？	
12	设计中需要包括转换及安装吗？	
13	系统的设计支持不同组织的多次安装吗？	
14	应用的设计方便用户修改和使用吗？	
总　计		

值	定义
0	没有影响
1	偶然的
2	适中的
3	普通的
4	重要的
5	极重要的

5 功能点度量（4/4）

- 功能点计算 FP= CT*(0.65+0.01*F)
  - 其中：CT是步骤1得到的“总计数值”，F是步骤2得到的Fi之和
- 一旦计算出功能点，则用类似代码行的方法来计算软件生产率、质量及其他属性

度量名	含义表示
生产率P	P=FP/E，E为开发的工作量（常用人月数表示）
每个功能点成本C	C=S/FP，S为总成本
每个功能点文档数D	D=Pe/FP，其中Pe为文档页数
功能点错误率EQR	EQR=N/FP，其中N为错误数

6 功能点与LOC的换算

程序语言	每个FP之LOC值			
	平均	中等	低	高
Access	35	38	15	47
Ada	154	—	104	205
APS	86	83	20	184
ASP	62	—	32	127
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
Java	63	53	27	170
COBOL	77	77	14	400
SQL	40	37	7	110
VBScript	34	27	50	—
Visual Basic	47	42	16	158

7.扩展的功能点度量

- 功能点度量的不足
  - 最初主要用于商业信息系统的度量，强调数据维，即信息域特征值，而忽略了功能和行为(控制)
- 特征点(Feature Point)：扩展功能点度量方法
  - **在功能点信息域特征中增加了一个算法特征**
  - **算法：为计算机程序中一个界定的计算问题**

测量参数	计数	加权因子	结果
用户输入数		×4	
用户输出数		×5	
用户查询数		×4	
文件数(表)		×7	
外部接口数		×7	
算法		×3	
总计CT			

小结

- 软件项目中，软件度量是项目管理的基础
- 本讲对软件度量、项目估算相关技术和方法进行介绍

软件协同设计课程之

3.2 软件质量

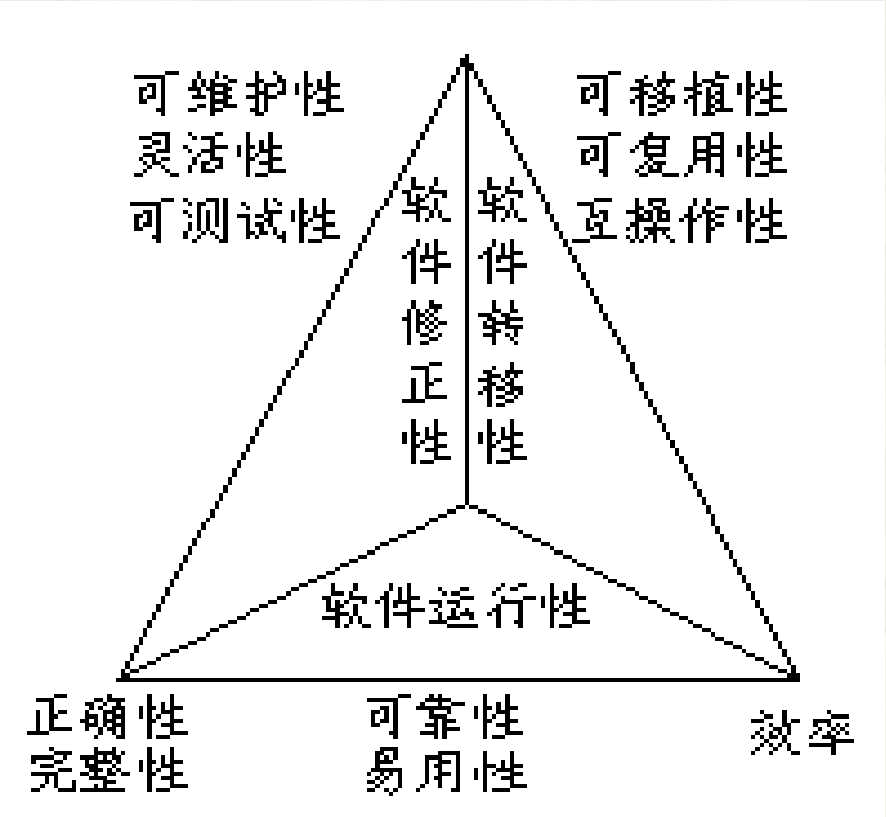
2025年1月

1.软件质量模型

- 课程目标之一：开发出高质量的产品
- 软件质量定义
  - GB/T25000.1-2010《软件产品质量要求与评价指南》中将软件质量定义为：**在规定条件下使用时，软件产品满足明确或隐含要求的能力。**
- 典型的软件质量模型
  - McCall模型
  - ISO9126 软件质量模型是评价软件质量的国际标准，由6个特性和27个子特性组成。

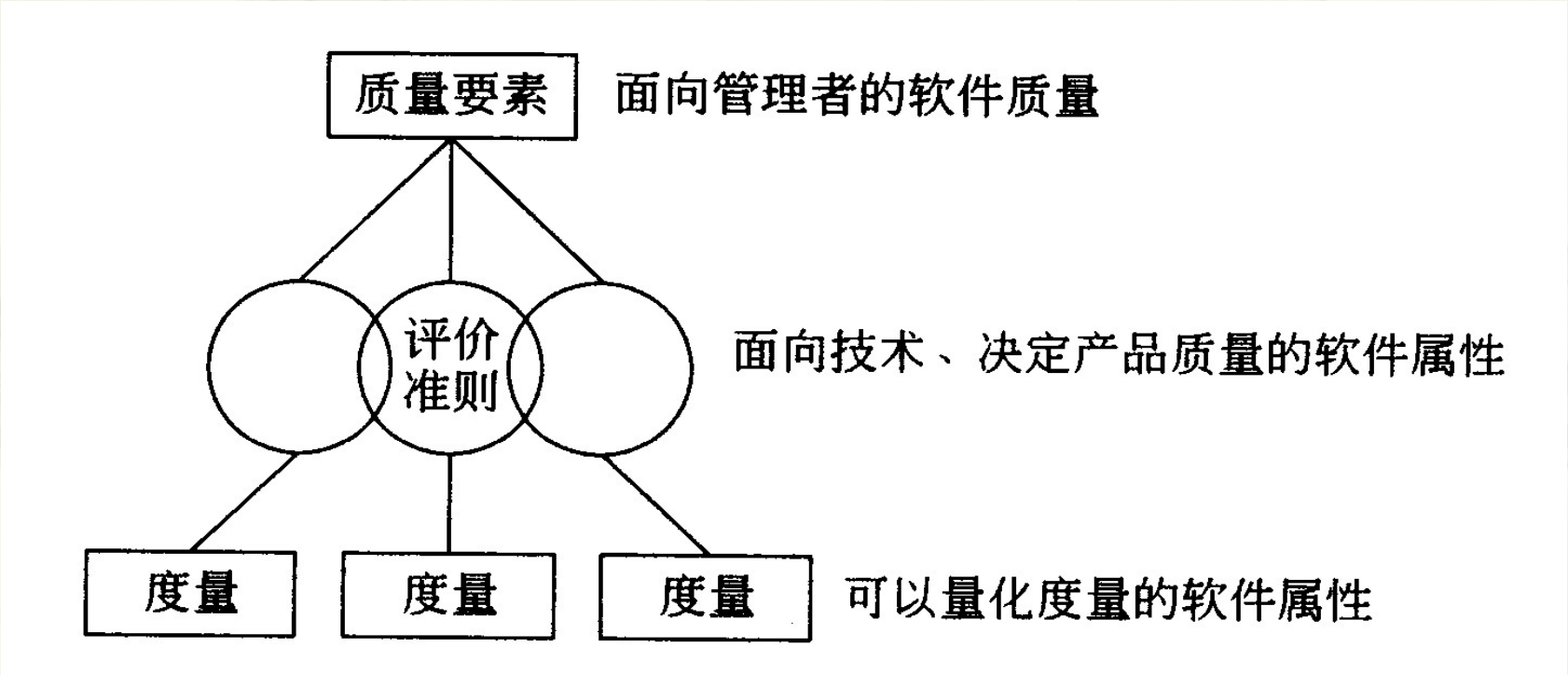
2. McCall模型

- 质量要素:又称为质量特征，反映软件的质量，如正确性、可靠性、效率等。
- 从软件产品的运行、修改和迁移三个方面分为11个软件质量要素



3. 软件质量要素评价准则

- 软件质量要素难以直接测量，因此需要为每个质量要素定义一组评价准则
- McCall定义了21种评价准则



4. 质量要素与评价准则的关系

关系		质量要素										
		正确性	可靠性	效率	完整性	可维护性	可测试性	可移植性	可复用性	互操作性	易用性	灵活性
评价准则	可审计性				√		√					
	准确性		√									
	通信共性									√		
	完备性	√										
	简洁性			√		√						√
	一致性	√	√			√						√
	数据共性									√		
	容错性		√									
	执行效率			√								
	可扩展性											√
	通用性							√	√	√		√
	硬件独立性							√	√			
	自检测性				√	√	√					
	模块性		√			√	√	√	√	√		√
	可操作性			√							√	
	安全保密性				√							
	自文档性					√	√	√	√			√
	简单性		√			√	√					√
	软件系统独立性							√	√			
	可追踪性	√										
	易培训性										√	

5. 软件质量要素的评价准则-1

(1)可审计性(auditability)

和标准的符合性、可被检查的容易程度。

(2)准确性(accuracy)

计算和控制的准确度。

(3)通信共性(communication commonality)

标准接口、协议和带宽的使用程度。

(4)完备性(completeness)

所需功能完全实现的程度。

(5)简洁性(conciseness)

以代码行数来评价的程序的简洁程度。

(6)一致性(consistency)

在软件开发项目中一致的设计和文档技术的使用。

(7)数据共性(data commonality)

在整个程序中对标准数据结构和类型的使用。

5. 软件质量要素的评价准则-2

(8)容错性(error tolerance)

当程序遇到错误时所造成的损失。

(9)执行效率(execution efficiency)

一个程序的运行性能。

(10)可扩展性(expandability)

结构、数据或过程设计可被扩展的程度。

(11)通用性(generality)

程序构件潜在的应用宽度。

(12)硬件独立性(hardware independence)

软件独立于其运行于之上的硬件的程度。

(13)自检测性(instrumentation)

程序监视它自身操作并且标识产生的错误的程度。

(14)模块性(modularity)

程序部件的功能独立性。

5. 软件质量要素的评价准则-3

(15)可操作性(operability)

程序操作的容易度。

(16)安全性(security)

控制和保护程序和数据的机制的可用度。

(17)自文档性(self-documentation)

源代码提供有意义的文档程度。

(18)简单性(simplicity)

一个程序可以没有困难地被理解的程度。

(19)软件系统独立性(software system independence)

程序独立于非标准编程特性、操作系统特性和其他环境限制的程度。

(20)可追踪性(traceability)

从一个设计表示或实际程序部件跟踪到需求的能力。

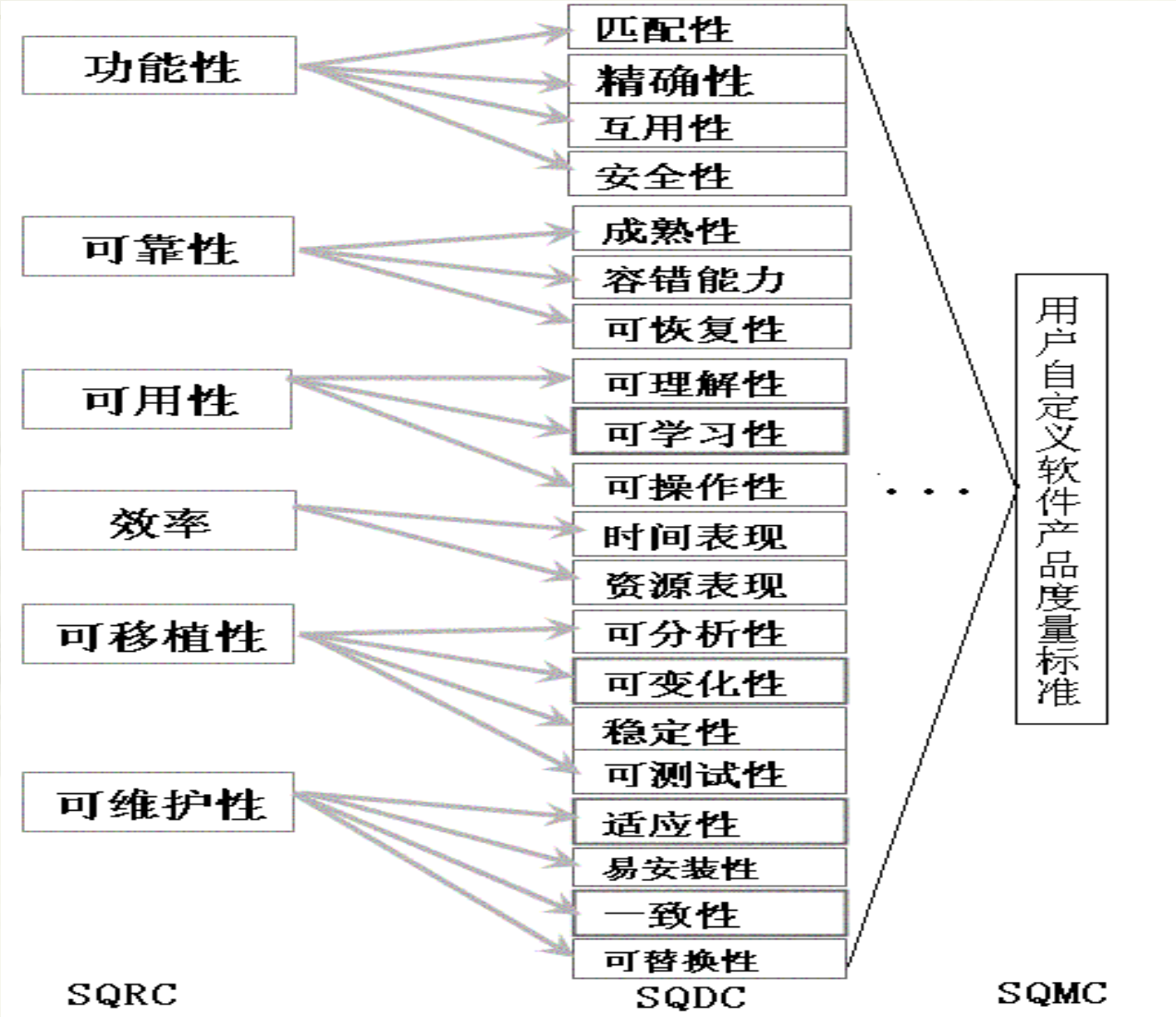
(21)易培训性(training)

软件支持使得新用户使用系统的能力。

6. 软件质量特征 (ISO 9126)

- **功能**：是满足明确或隐含的需求的那些功能。
- **可靠**：在规定的一段时间和条件下，软件维持其性能水平的能力。
- **易用**：由用户为使用软件所需作的努力和评价。
- **效率**：在规定条件下，软件的性能水平与所使用资源量之间关系。
- **可维护**：进行指定的修改所需的努力的大小。
- **可移植**：软件从一个环境转移到另一个环境的能力。

ISO 9126软件质量三层模型



小结

本讲介绍了软件质量的概念、两种质量模型、以及质量模型的质量要素与评价准则

软件协同设计课程之

3.3 软件协同开发中的软件质量管理

2025年1月

1. 软件质量管理

- 高质量的软件必须具备以下条件：
  - 满足软件需求定义的功能和性能
  - 文档符合事先确定的标准
  - 软件的特点和属性遵循软件工程的目标和原则
- 除此以外，还有迁移、维护等方面质量要素

2. 质量控制和质量保证

- 质量控制是为了保证每一件工作产品都满足预先规定的要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：实际是一个反馈循环，当发现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

3. 质量控制和质量保证

- 质量保证：由管理层的审核和报告构成
  - 目标是为管理层提供获知产品质量信息所需的数据
  - 判断获得的产品质量是否符合预定目标

4. 软件质量保证执行

- 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。基于评审结果，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小组 (Software Quality Assurance)：辅助软件工程小组得到高质量的最终产品

5. SQA小组的活动(CMU SEI)

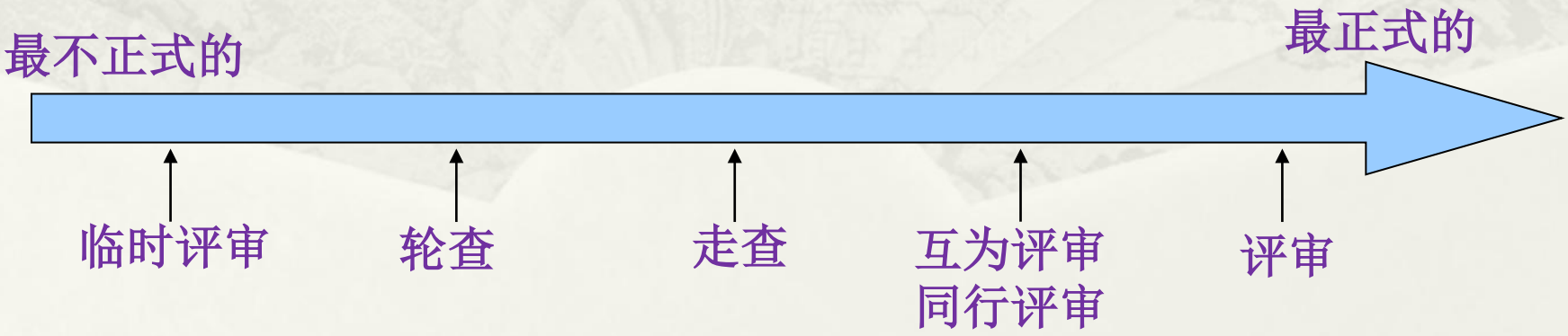
I. 为项目准备SQA计划
II. 参与开发该项目的软件过程描述
III. 评审各项软件工程活动，以验证其是否符合定义的软件过程
IV. 审核指定的软件工作产品，以验证其是否符合定义的规范
V. 确保软件工程及工作产品中的偏差已被记录在案，并根据预定规程进行了处理
VI. 记录所有不符合的部分并报告给高层管理者
VII. SQA小组还需要协调变化的控制和管理，并帮助收集和分析软件度量信息

6. 软件评审

- 软件评审:评审是对软件元素或者项目状态的一种评估手段，以确定其与计划的结果是否保持一致，并使其得到改进，是软件质量保证的重要手段
- 通过软件评审，可以更早地发现需求工程、软件设计等各个方面的问题，大大减少后期返工
- 通常在软件项目的每个活动完成后(如需求分析、设计、编码)，进行正式的软件评审

7. 评审的形式/方法

- 互为评审 (Peer review)
- 轮查 (Pass-round)
- 走查 (walk-through)
- 会议评审 (Inspection)



8.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，通过正式评审的活动标志着该活动到达了一个里程碑，该活动的制品也就成为一个基线
- 非正式评审(informal reviews)通常是一种由同事参加的即兴聚会，大多采用“走查”(walkthrough)的方式

9. 评审分类

- 技术评审
- 管理评审
- 文档评审
- 流程评审

10. 技术评审

- 技术评审的任务是：帮助他人对软件交付物进行检查
  - 它们是完备的
  - 它们符合标准和规范
  - 对它们的更改是正确地实施的
- 以发现软件交付物缺陷、获得改进机会
- 是强有力的质量改进获得

11. 正式的技术评审过程

- 评审会议
  - 由评审会主席和若干名评审员组成，参加者大多是与评审内容相关的技术专家，参加人员不宜太多，通常为3～5人
  - 必要时(如需求评审)可请用户代表参加
- 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

12. 正式的技术评审过程

- 评审报告
  - 评审会结束时应形成评审总结报告，总结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
  - 评审总结报告不必很长(通常一页纸就够了)
  - “评审问题列表”可作为评审总结报告的附件。

13.评审的指导原则

- 评审产品，而不是评审生产者
- 制定议事日程且遵守日程
- 限制争论和辨驳
- 对各个问题都发表见解，但不要试图解决所有记录的问题
- 做书面笔记
- 限制参与者人数并坚持事先做准备
- 为每个可能要评审的工作制品建立一个检查表
- 为正式技术评审分配资源和时间
- 对所有评审者进行有意义的培训
- 评审以前所做的评审

小结

本讲介绍了质量保证、软件评审是质量保证的重要手段

小组对产品评审时，应该遵循评审指导原则

- 软件协同设计课程之

4.1 软件需求

- 2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

内　　容

- 需求的重要性
- 需求的理解与沟通
- 从业务需求到功能需求

1. 需求获取的重要性

- 软件需求
  - 用户对目标软件系统在功能、行为、性能、设计约束等方面的期望。
  - 内容包括：FURPS +，
    - Functionality
    - Usability
    - Reliability
    - Performance
    - Supportability
    - +

1. 需求获取的重要性

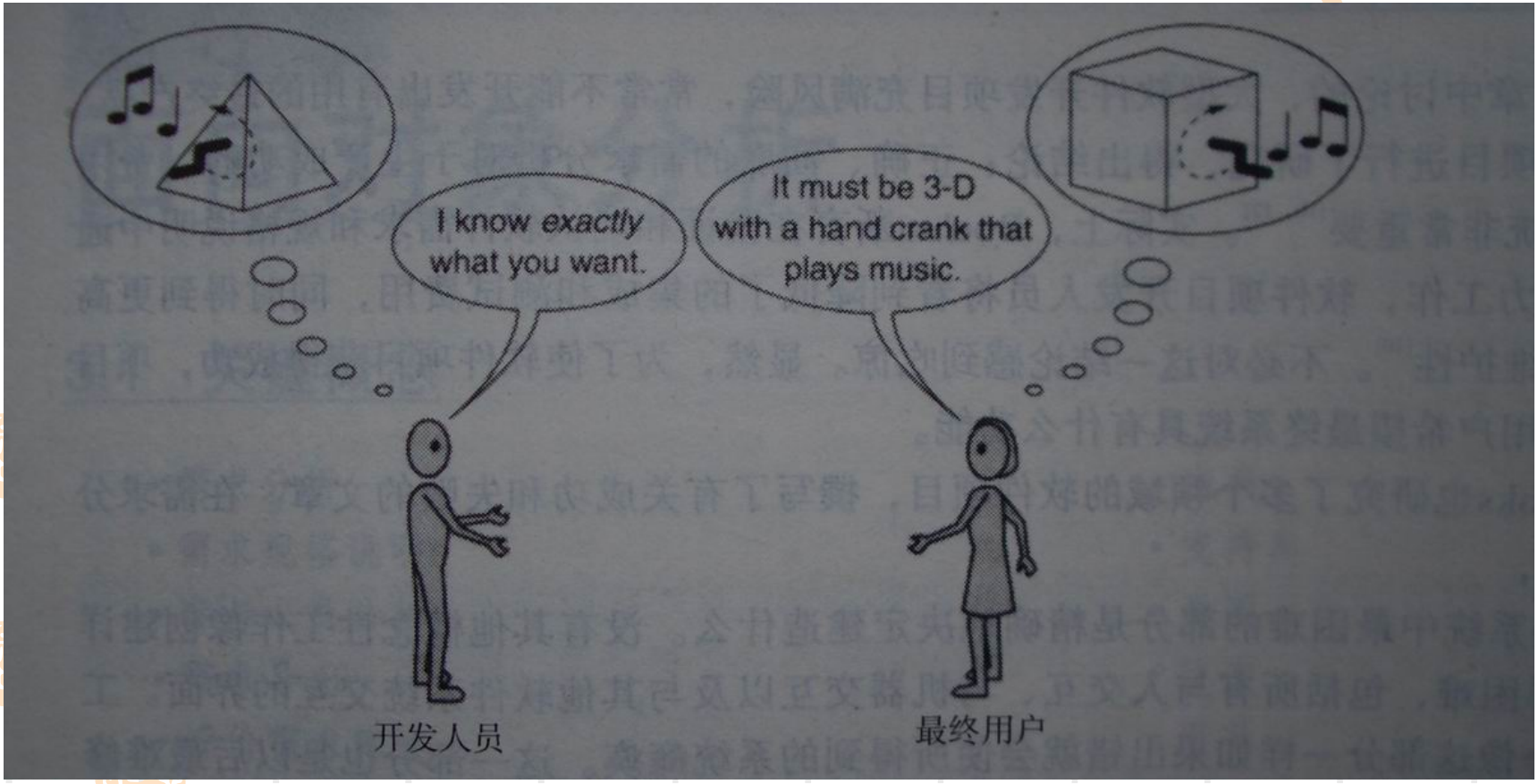
- 三个问题：
- 理解需求
  - 开发者只有在充分理解了用户需求后，才能开始设计系统
- 需求正确
  - 只有需求是正确的，给出的解决方案才可能是正确的
- 需求完备
  - 在开发的后期，需求的变更会引起大量的返工。

1. 需求获取的重要性

Top 10 factors for software failure--CMU SEI

No.	Top 10 Factors	平均值
1	Inadequate requirements specification 不充分的需求规范	4.5
2	Changes in requirements 需求的改变	4.3
3	Shortage of systems engineers 缺乏系统工程师	4.2
4	Shortage of software managers 缺乏了解软件特性的经理人	4.1
5	Shortage of qualified project managers 缺乏合格的项目经理	4.1
6	Shortage of software engineers 缺乏软件工程师	3.9
7	Fixed - price contract 固定价合同	3.8
8	Inadequate communications for system integration 系统集成阶段，交流与沟通不充分	3.8
9	Insufficient experience as team 团队缺乏经验	3.6
10	Shortage of application domain experts 缺乏应用领域专家	3.6
Scale: 5 = Very Serious 3 = Serious 1 = No Serious		

2. 需求理解问题


I know *exactly* what you want.
It must be 3-D with a hand crank that plays music.
开发人员
最终用户

2. 需求理解问题

Maria：喂，Phil吗？我是人事部的Maria。我们在使用你开发的人事系统时遇到一个问题。有位职员刚刚把她的名字改成Sparkle Starlight，但我们无法在系统中改。你能帮个忙吗？

Phil：那么她是结婚了，随老公姓Starlight？

Maria：没有，她没结婚，只是改名字了，问题就出在这里。好像我们只能在某人婚姻状况发生变化时才能在系统中改名。

Phil：好吧，是，我从来没想过有人可能会改自己的名字。当初我们在讨论系统的时候，你可没告诉过我有这种可能性。

Maria：我以为你知道任何人随时都可以合法更改名字呢，我们得在星期五之前解决这个问题，否则Sparkle就领不到工资了。你可以在此之前修复这个bug吗？

Phil：这不是什么bug，好吗？！ 我从没想过你们需要这项功能。我现在正忙着做一个新的绩效评估系统。你所说的问题我只能在月底修复，但周五之前肯定不行，抱歉。下次如果再有类似情况，请早点告诉我，并请提供书面材料。

Maria：那我怎么和Sparkle说呢？如果她领不到工资，会很难过的。

Phil：嗨，Maria，这不是我的错，如果当初你早提醒我你需要能够随时更改某人的姓名，这种事情就不会发生。你不能因为我没猜透你的想法就怪我。

2. 需求理解问题

- 软件需求开发：使小组对用户要求有一个统一的理解。
- 小组开发人员与用户交流差异：有不同知识域，沟通困难
- 用户往往不清楚自己需要什么
  - 不好描述, 如“反应快、“界面友好真正含义？”
  - 考虑不周，如“根据教学计划排课太笼统“
  - 想当然，以为软件万能

3. 从业务需求到功能需求

- 业务需求-----用户需求-----功能需求
  - 得出软件需求规范
- 例如：字处理的拼写检查与修正
  - 业务需求：产品允许用户轻松地更正文档中的拼写错误。
  - 用户需求：找出拼写错误、修改错误、把单词加到词典中等
  - 功能需求：
    - 找到并突出显示拼错的单词
    - 用对话框显示修改建议
    - 用正确的单词替换整篇文档中同一单词所有拼写错误。

3. 从业务需求到功能需求

- 项目需求开发
  - 老师作为甲方 提出基本需求
  - 开发组与甲方产品代表 就关心的问题深入交流
  - 需求建模与SRS
  - 需求评审
- 本课程的项目基本需求介绍
- 需求会议（教师、小组）：
  - 讨论、制定SRS
- 小组交一份SRS

总结

- 需求获取是困难的
- 沟通交流常有误解
- 有业务需求向软件需求转换有一定复杂性

- 软件协同设计课程之

4.2 需求建模

2025年1月

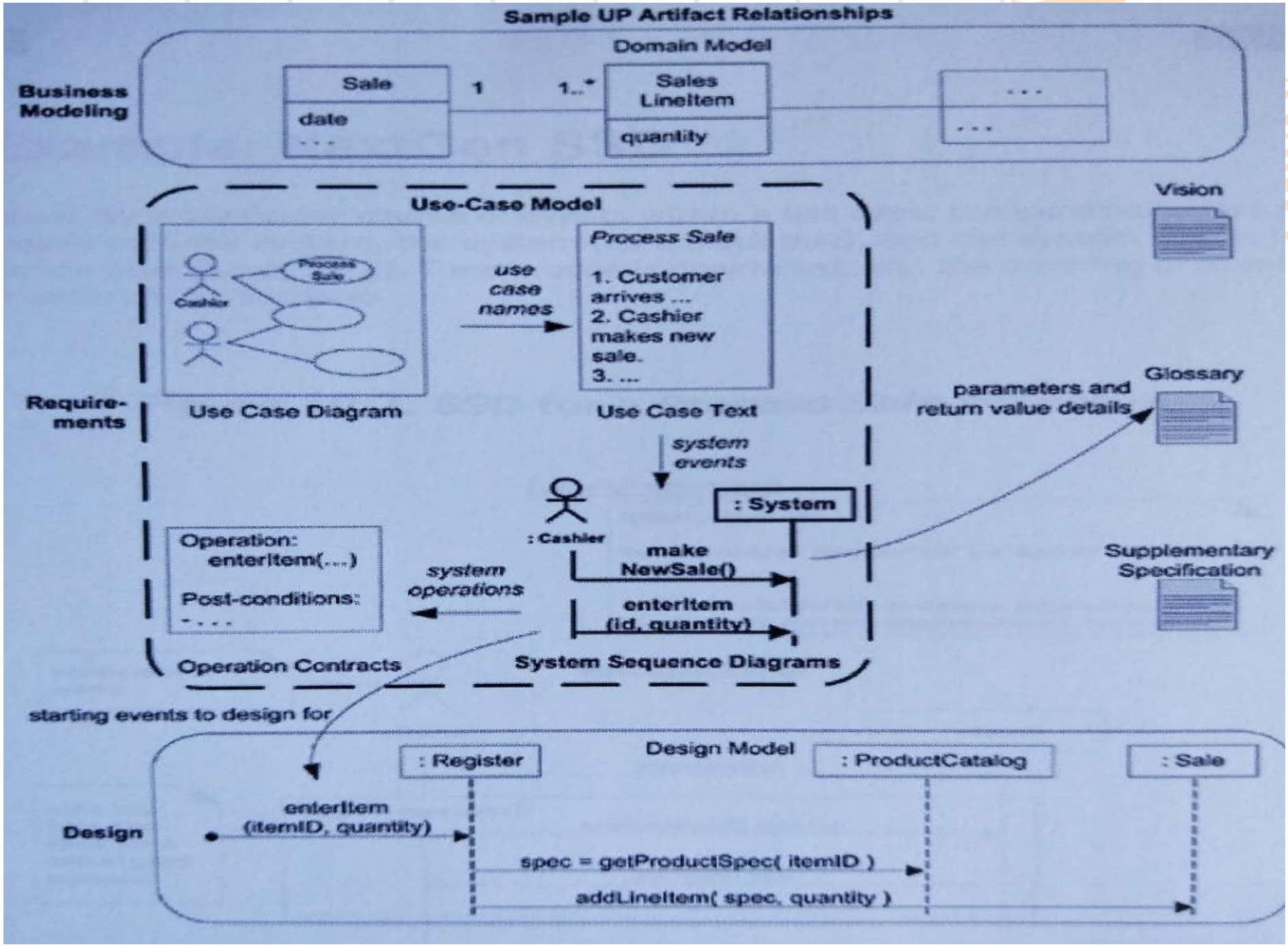
上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

内容

- 需求建模过程
- 什么是领域模型
- 需求建模工具
- 软件需求规范SRS
- SRS的检查与用户的复查

1. 需求建模过程

- 建模过程
  - 需求获取
  - 需求分析
  - 需求规格编写
  - 需求验证
- 需求重要模型
  - Domain Model
  - Use-Case Model


Sample UP Artifact Relationships
Domain Model
Business Modeling
Sale
date
...
1
1..*
Sales LineItem
quantity
...
...
Use-Case Model
Vision
Process Sale
Cashier
use case names
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
Use Case Diagram
Use Case Text
Require-ments
parameters and return value details
Glossary
system events
: System
: Cashier
Operation:
enterItem(...)
Post-conditions:
- ...
system operations
make NewSale()
enterItem (id, quantity)
Supplementary Specification
Operation Contracts
System Sequence Diagrams
starting events to design for
Design Model
: Register
: ProductCatalog
: Sale
Design
enterItem (itemID, quantity)
spec = getProductSpec(itemID)
addLineItem(spec, quantity)

2. 什么是领域模型（Domain Model）

- Domain Model，Business Object model
  - 业务中的概念类和业务对象之间关系的可视化表示
  - 实际的业务模型，主要用于业务建模
- 使用没有方法的类图表示
  - 业务对象、概念类
  - 概念类之间的关联

3. 需求建模工具

- Use-Case Model + Use-Case Text
  - 识别出Actor
  - 识别出用例
  - 给出用例图
  - 用例的文字描述(Use-Case Text)，或者用活动图描述（activity diagram），或者状态机图（state machine diagram）

4. 软件需求规范SRS

- 用户需求转化成SRS
  - 由模糊到清晰
  - 用需求到功能
    - 主要从功能与操作方面描述，Use-Case 图+文字描述
  - 用户到开发的基线baseline
- 小组领导与开发经理
  - SRS文档的设计与任务分配

5. 需求规约提纲

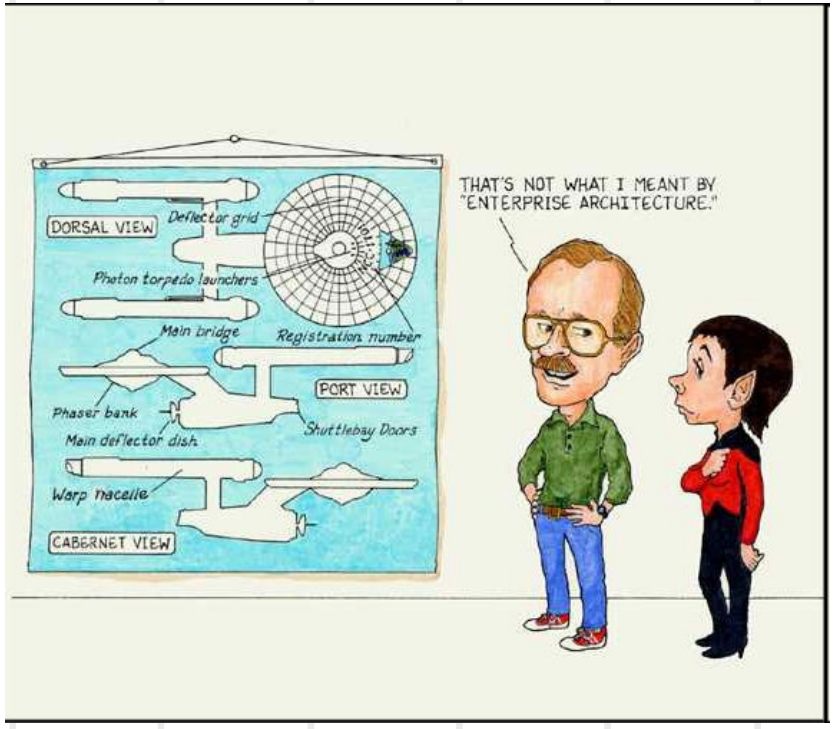
1. 目录
2. 引言
3. 需求概述
4. 需求分析
5. 系统环境

5. 需求规约提纲

- 报告撰写提示\需求说明书编写说明（新）.doc

6. 需求审查

- 审查认真阅读SRS文档，真正理解客户的需求和技术上的设计，检查需求说明书对产品描述的准确性、一致性等，检查系统设计的合理性和可测试性等
- 最好有客户参加



6. 需求审查

- 质量经理负责
  - 列出已清楚的问题
  - 梳理尚不清楚、有待与客户讨论的问题
  - 质量经理负责，并记下每个错误，审查时间
  - 小组决定对错误的修改
- 报告撰写提示\需求分析审查提纲.doc

7. SRS的用户复查

- 最后请用户审查
  - 由老师完成

软件协同设计课程之

5.1 软件开发计划的困惑

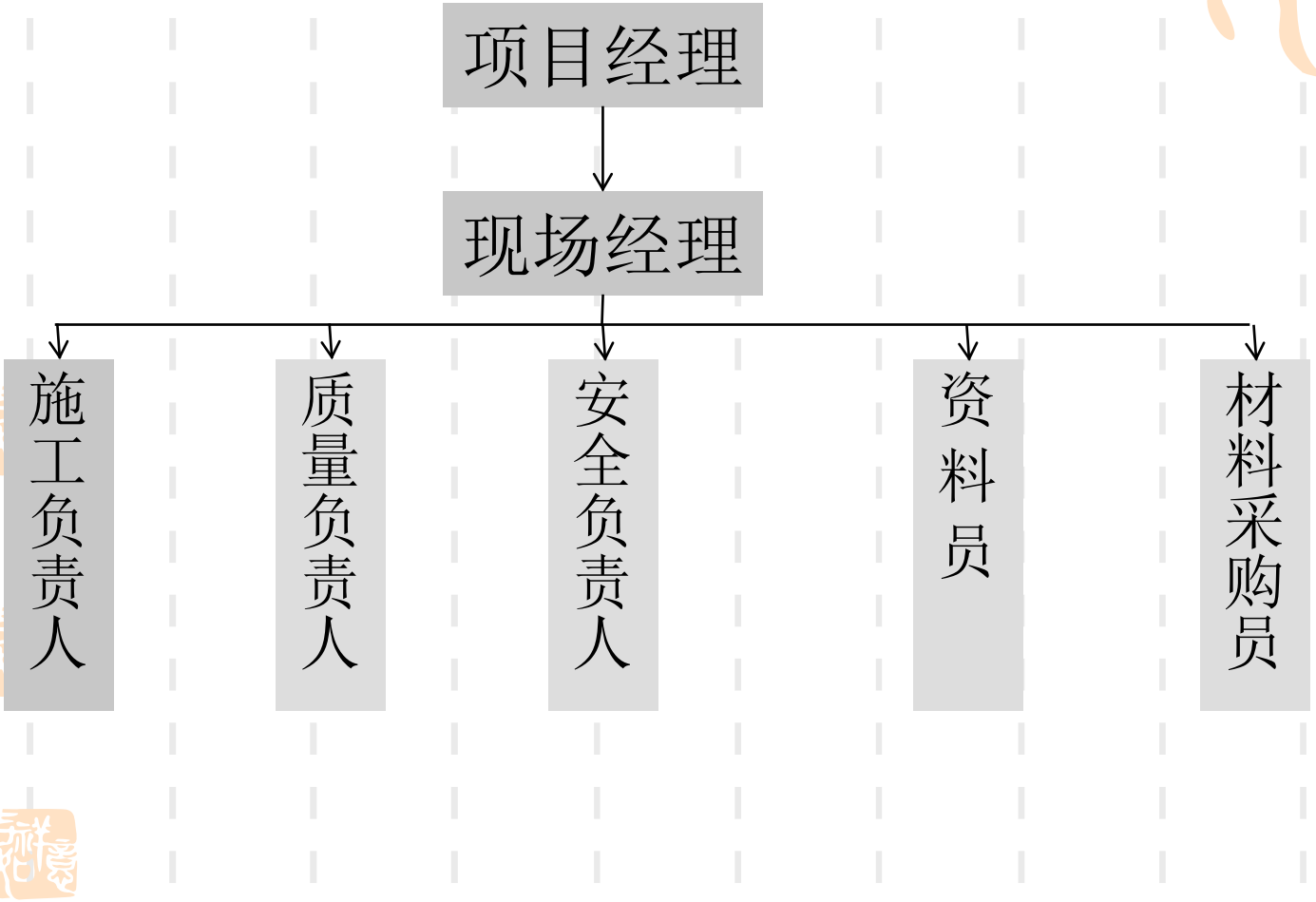
2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

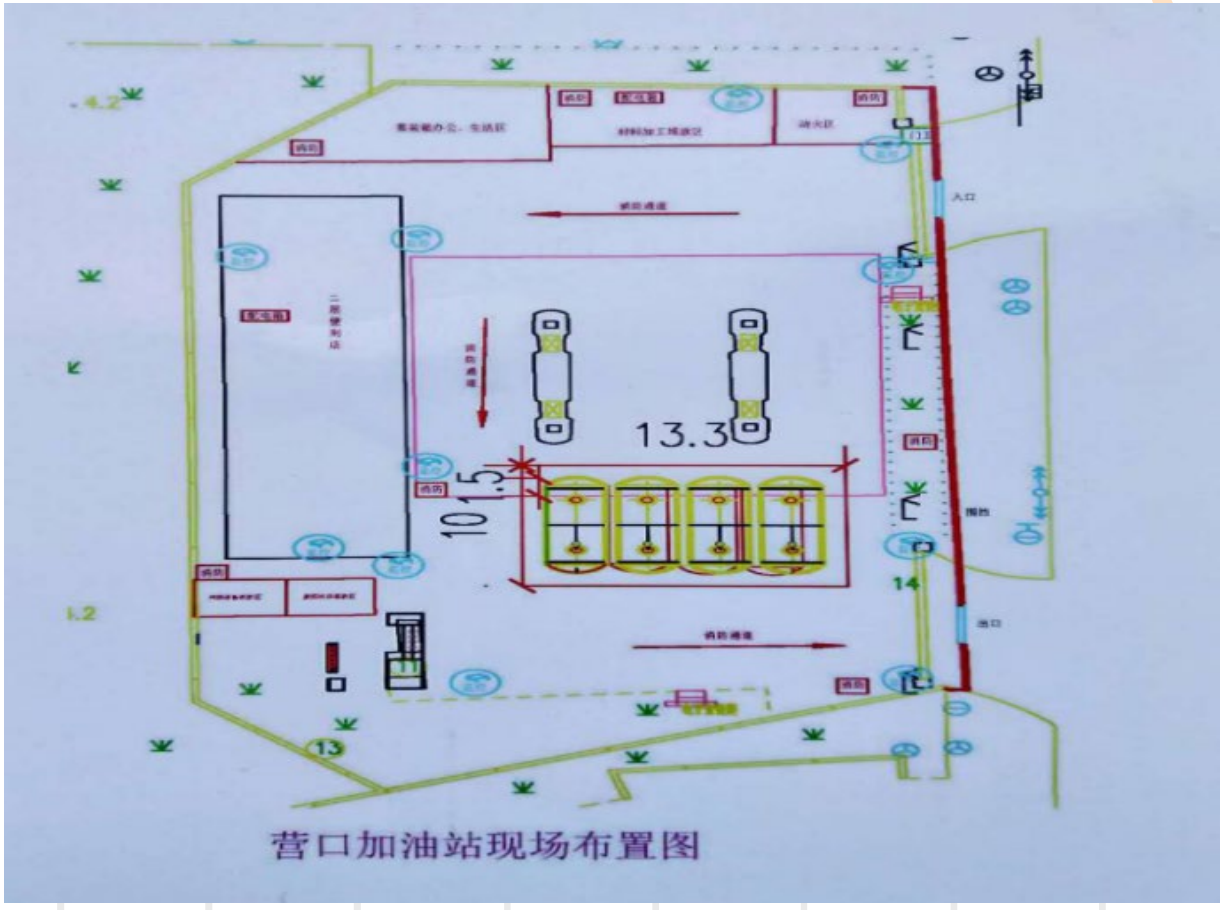
1. 开发计划举例

- 两个例子
  - 工程项目计划举例
  - 软件项目计划举例

2. 工程项目举例：某加油站改造项目

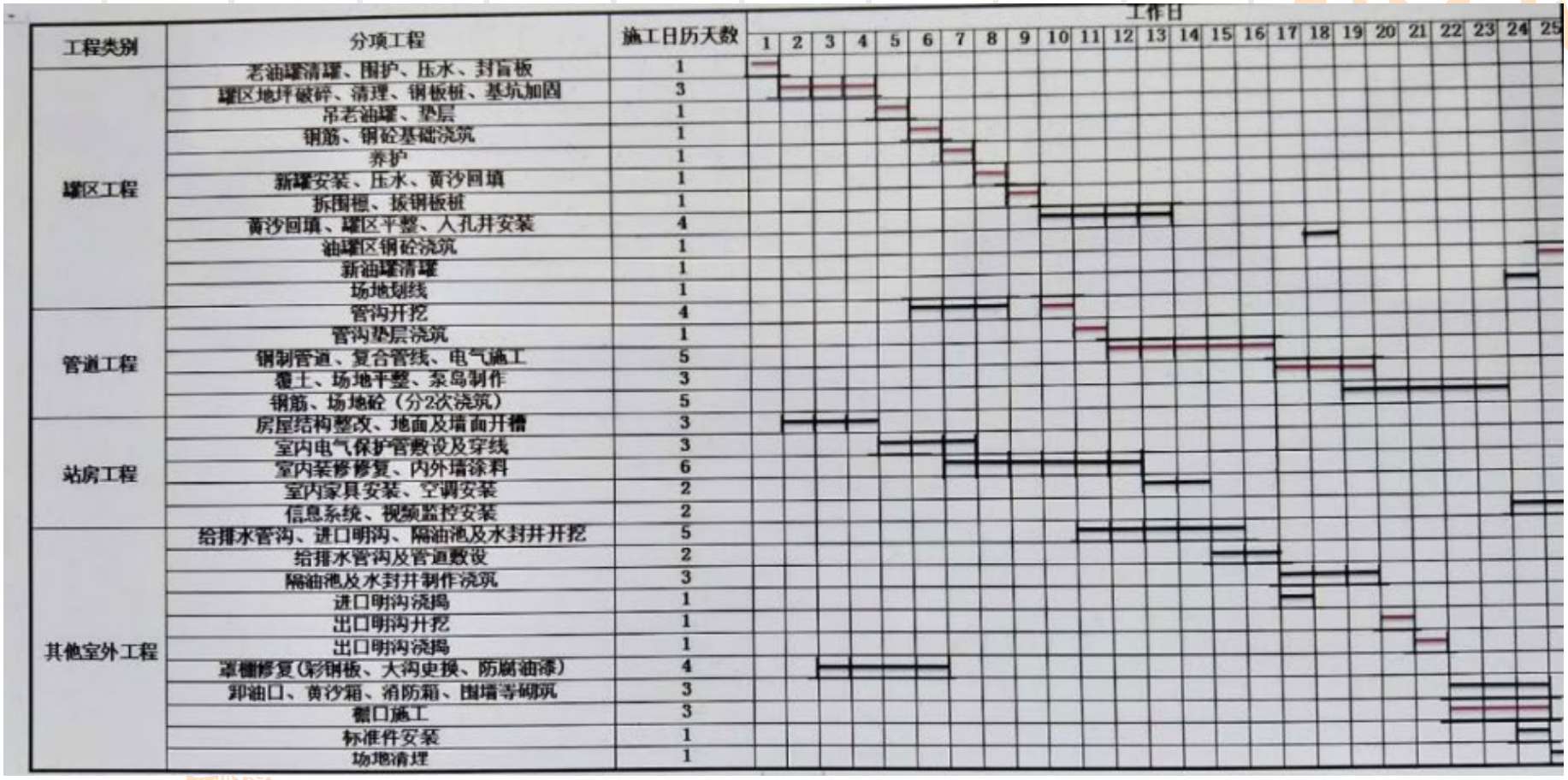


2. 工程项目举例：某加油站改造项目



营口加油站现场布置图

2. 工程项目举例：施工计划进度表

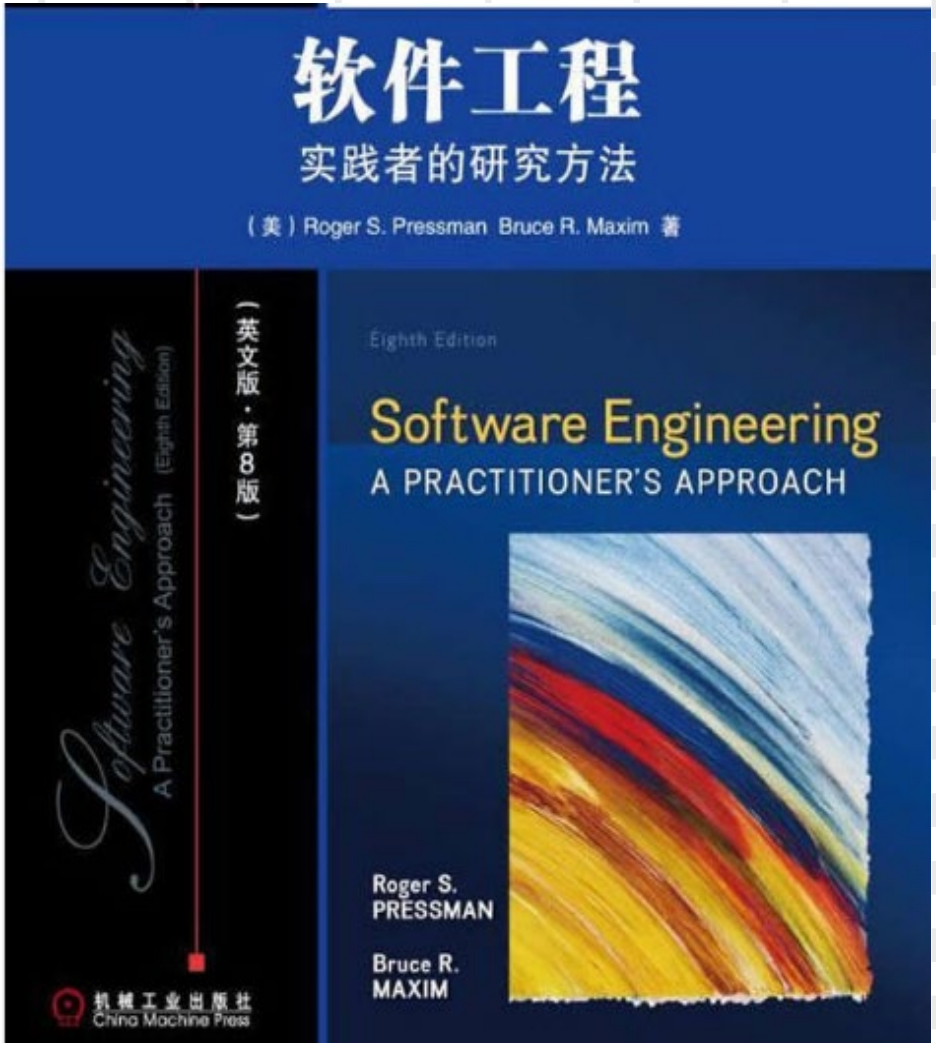


工程类别	分项工程	施工日历天数
罐区工程	老油罐清罐、围护、压水、封盲板	1
	罐区地坪破碎、清理、钢板桩、基坑加固	3
	吊老油罐、垫层	1
	钢筋、钢砼基础浇筑	1
	养护	1
	新罐安装、压水、黄沙回填	1
	拆围檩、拔钢板桩	1
	黄沙回填、罐区平整、人孔井安装	4
	油罐区钢砼浇筑	1
	新油罐清罐	1
	场地划线	1
管道工程	管沟开挖	4
	管沟垫层浇筑	1
	钢制管道、复合管线、电气施工	5
	覆土、场地平整、泵岛制作	3
	钢筋、场地砼（分2次浇筑）	5
站房工程	房屋结构整改、地面及墙面开槽	3
	室内电气保护管敷设及穿线	3
	室内装修修复、内外墙涂料	6
	室内家具安装、空调安装	2
	信息系统、视频监控安装	2
其他室外工程	给排水管沟、进口明沟、隔油池及水封井开挖	5
	给排水管沟及管道敷设	2
	隔油池及水封井制作浇筑	3
	进口明沟浇捣	1
	出口明沟开挖	1
	出口明沟浇捣	1
	罩棚修复(彩钢板、大沟更换、防腐油漆)	4
	卸油口、黄沙箱、消防箱、围墙等砌筑	3
	翻口施工	3
	标准件安装	1
	场地清理	1

3. 一个缺乏计划软件项目

- 例1(Pressman)
  - 60年代后期
  - 编写一个自控机床应用软件（用c,Fortran）
  - 一堆手册，两个月完成
  - 阅读了手册，想好了解决方案
  - 两周后，老板把他叫进办公室询问情况
  - 很简单，完成75%
  - 一周后，遇到一点小问题，老板问能否按时完成？90%
  - 整个工期内90%
  - 在别人的帮助下，一个月后完成

3. 软件项目举例-Pressman

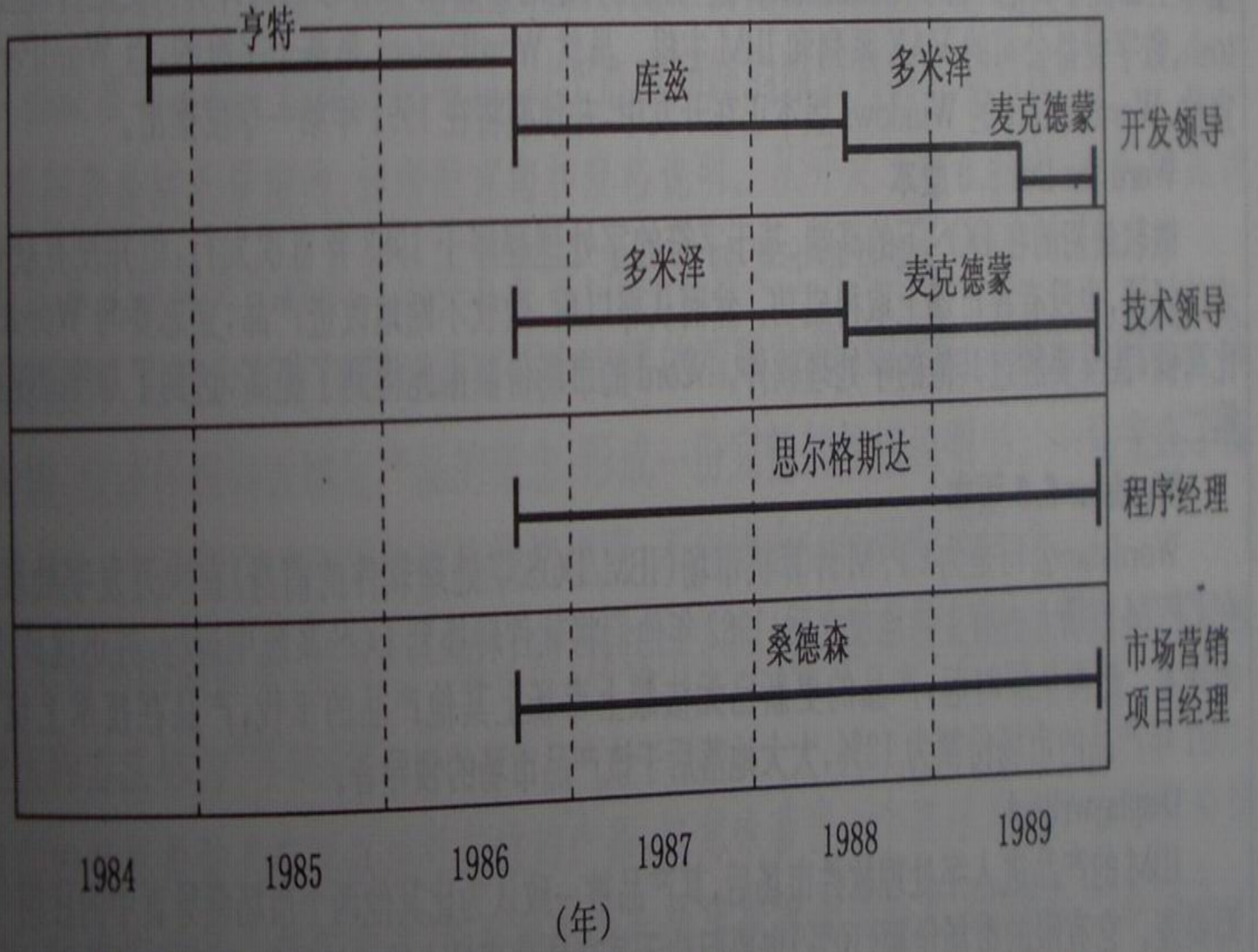


4. 计划变更13次的项目： Word for windows

报告日期	估计完成日期	估计距完成需要的天数	实际距完成需要的天数
1984年9月	1985年9月	365	2 187
1985年6月	1986年7月	395	1 614
1986年1月	1986年11月	304	1 400
1986年6月	1987年5月	334	1 245
1987年1月	1987年11月	334	1 035
1987年6月	1988年2月	245	884
1988年1月	1988年6月	152	670
1988年6月	1988年10月	122	518
1988年8月	1989年1月	153	457
1988年10月	1989年2月	123	396
1989年1月	1989年5月	120	304
1989年6月	1989年9月	92	153
1989年7月	1989年10月	92	123
1989年8年	1989年11月	92	92
1989年11月	1989年11月	0	0

4. 计划变更13次的项目： Word for windows

日 期	事 件
1984 年 8 月	开始 WinWord 项目
1985 年 8 月	开始编码
1985 年 9 月	首次向比尔·盖茨演示
1985 年 11 月	加紧编码工作
1987 年 1 月	开始测试工作
1987 年 4 月	与比尔·盖茨讨论规格说明书
1987 年 6 月	修改规格说明书
1987 年 10 月	补充形成最终的规格说明书
1987 年 12 月	视觉定位:正式确立面对用户的程序外观
1988 年 2 月	向比尔·盖茨陈述,增添附加的特征
1988 年 3 月	补充形成最终的规格说明书
1988 年 6 月	报告的所有特征几乎都是“在某种程度可以运行的”
1988 年 7 月	补充形成最终的规格说明书
1988 年 8 月	特征完成:此后不再加入任何特征
1988 年 10 月	编码完成
1989 年 3 月	性能的优化工作宣告完成
1989 年 10 月	正式发布 Word for Windows
1989 年 11 月	Word for Windows 交付完工


亨特
库兹
多米泽
麦克德蒙
开发领导
多米泽
麦克德蒙
技术领导
思尔格斯达
程序经理
桑德森
市场营销
项目经理
1984
1985
1986
1987
1988
1989
(年)

5. 大型工程项目-他山之石

- 维拉泽诺海峡大桥
  - 纽约连接斯塔滕岛和布鲁克林的维拉泽诺海峡大桥
  - 世界上最长的吊桥：中心长度1298米，造价3.25亿美元，工程始于1959年，计划1965年完工。
  - 实际1964年11月竣工。费用控制在预算内。
  - 值得学习：

source: Software Verification and Validation for practitioners and managers

The Verrazano-Narrows Bridge spans the Narrows in New York City, connecting the boroughs of Brooklyn and Staten Island.

Close up of the Narrows, as seen by satellite. Staten Island is on the left, and Brooklyn is on the right, connected by the Verrazano-Narrows Bridge (Public Domain photograph from NYSGIS).

由布鲁克林边望去的韦拉札诺海峡大桥。

韦拉札诺海峡大桥

Verrazano-Narrows Bridge

在早晨时自史泰登岛望向韦拉札诺海峡大桥

世界上最长的吊桥：中心长度1298米，造价3.25亿美元，
工程始于1959年，计划1965年完工。实际1964年11月竣工。
费用控制在预算内。

建造工程于1959年8月13日动工，并于1964年11月21日完工而上层通车，总造价共花费超过$3.2亿美元。
完工后的韦拉札诺海峡大桥因比金门大桥长所以成为当时全世界最长的悬索桥。

根据美国运输部：[1]

- 两座塔架各自含有1,000,000个螺帽以及3,000,000个螺钉。
- 桥上的四条悬索的直径各为11米（36英尺），每条悬索内部是由26,108条钢线所组成，而所有钢线的全长总和为230,000千米（143,000英里）。
- 由于桥的长度（1,290米）以及塔架的高度（210米），在设计的时候必须要将地球的球体曲线加入设计考量之中。
- 由于热涨冷缩的缘故，桥面的倾斜度在夏天的时候比冬天时少12度角。

总结

- 没有计划的项目是危险的
- 软件项目计划有许多不可控因素，制定计划难度大
- 大型工程项目的计划制定可以参考

软件协同设计课程之

5.2 任务分解WBS

2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

1. 为何使用wbs

- 工作分解结构(work breakdown structure)
  - 把一个项目进行分解，得到一些子项目
  - 针对每个子项目，确定实现该子项目的一组活动
- wbs始终处于计划的中心，是制定计划的基础
  - 进度计划
  - 资源需求
  - 成本预算
  - 采购计划与风险管理计划
- wbs是做项目计划、合理分工、协作完成项目的基础

2. WBS分解方法W型

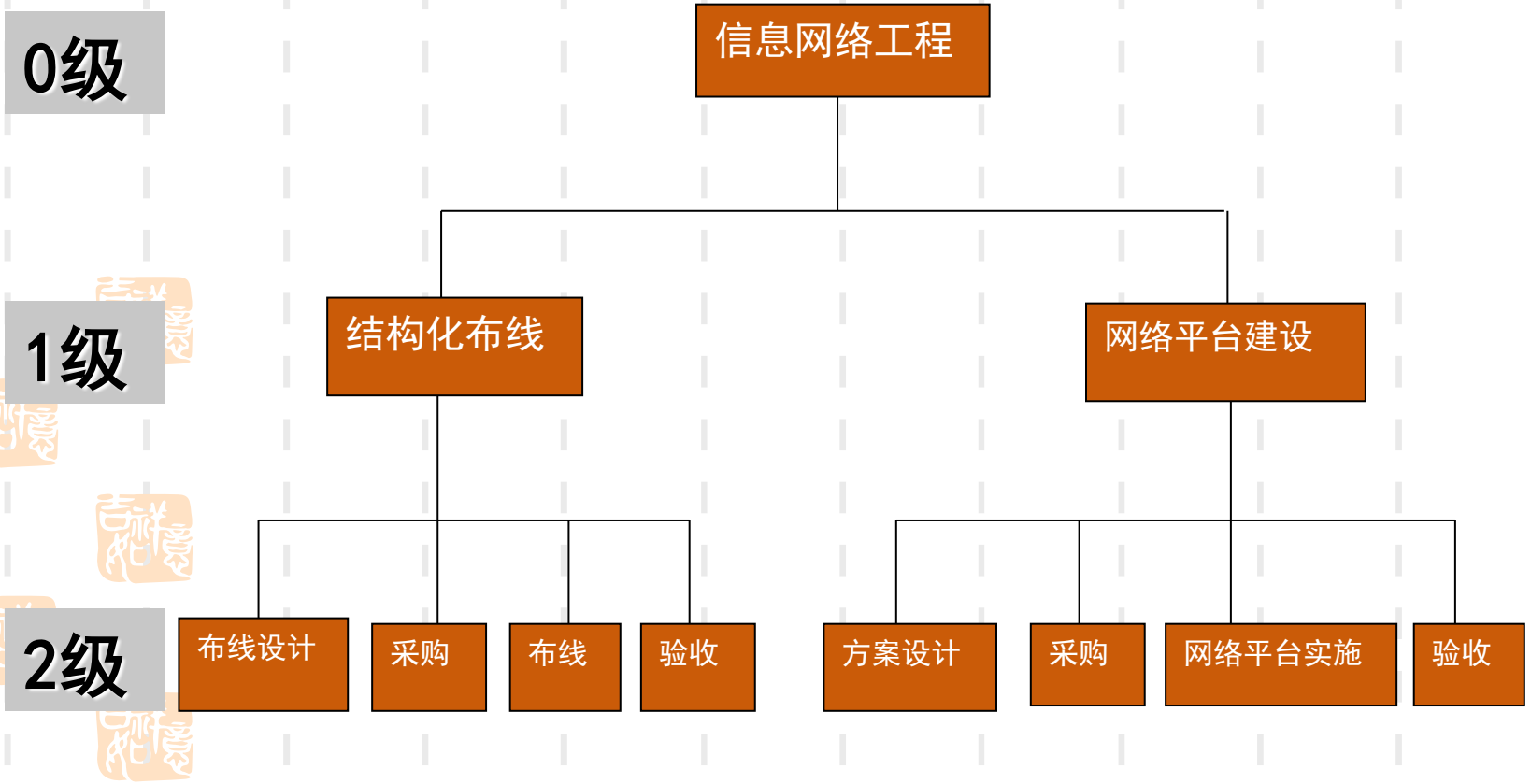
- 按产品或项目的功能分解
- 按照实施过程分解
- 按产品的物理结构分解
- 按项目的各个目标分解
- 按部门分解

3. 如何使用WBS W型

- 拿到项目首先思考需要交付的成果是什么
- 为了完成这些成果，需要哪些步骤去实现？

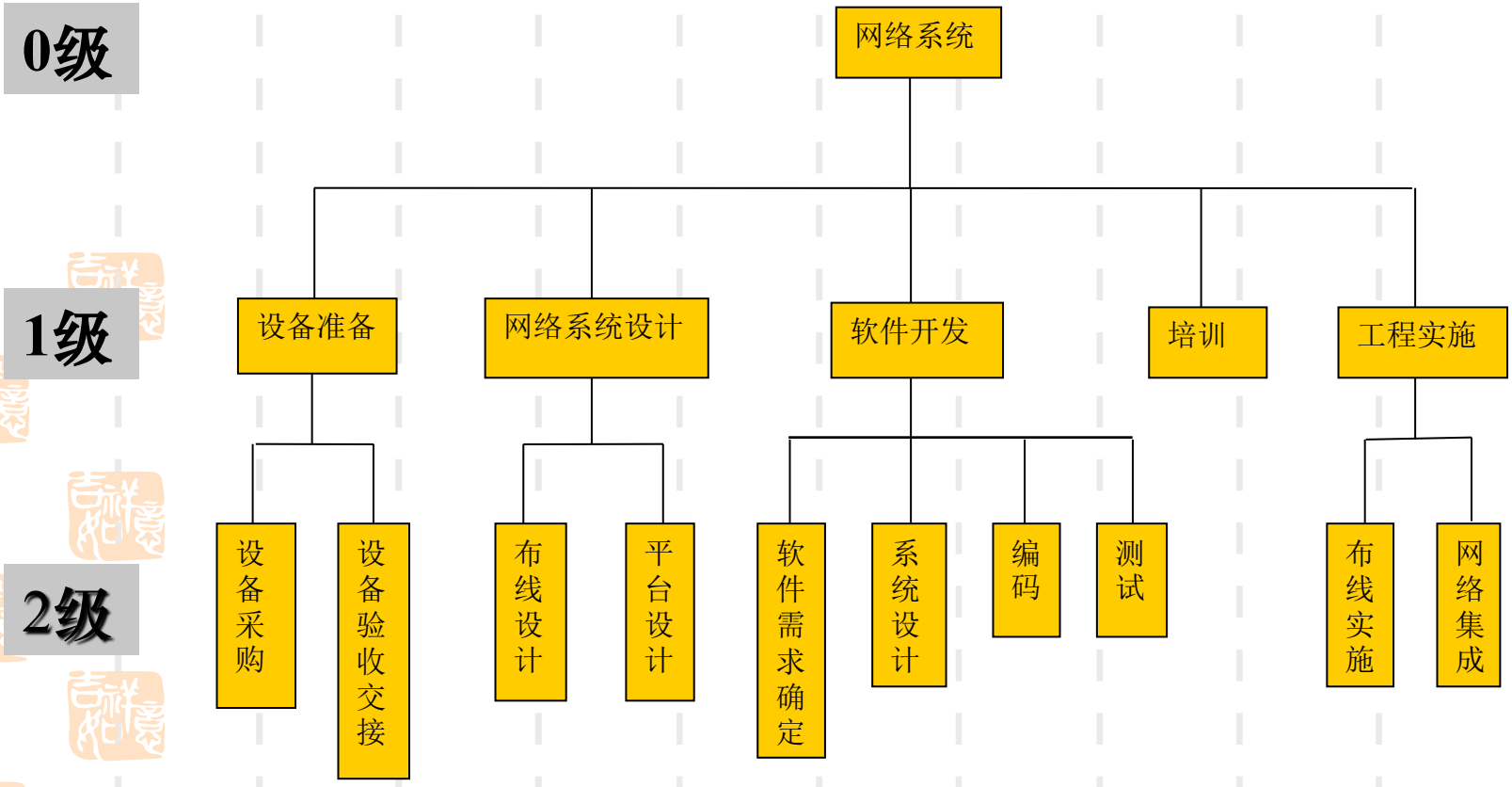
3. 如何使用WBS W型

- 基于可交付成果的划分



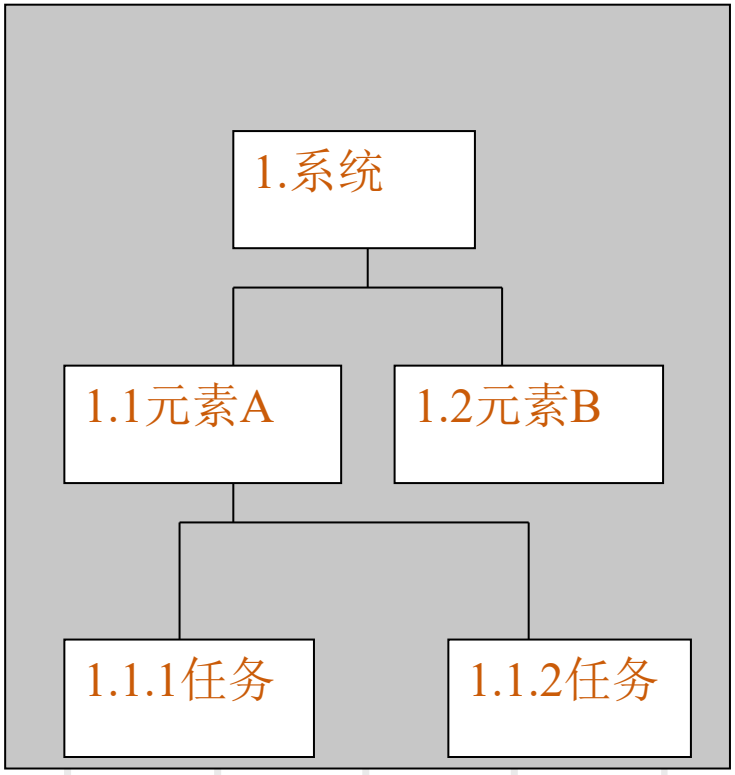
3. 如何使用WBS W型

基于工作过程的划分



3. 如何使用WBS

wbs序号:无论按产品还是按工作过程



1 系统
- 1.1元素A
  - 1.1.1任务
  - 1.1.2任务
- 1.2元素B

4. WBS举例：车库项目

一级	二级	三级	四级
车库项目	美化场地	车道	
		美化	
	车库	材料	
		地基	
		墙体	墙面
			窗户
			车库门
			检修门
			组装
		屋顶	构架
			遮盖物
			排水槽
		公用工程	电
			水
	项目管理	施工计划	
		许可证	
		检验	
		筹资	
		分包	

软件项目分解的实例

阶段和项目标准过程	ID	任务	工作成果名称
项目策划阶段《项目策划管理规范》	1	起草项目任务书	《项目任务书》
	2	审批项目任务书	已批准的《项目任务书》
	3	策划准备	《项目实施计划》
	4	启动项目策划	产品的功能结构图、WBS工作任务分解
	5	项目估计和成果列表	《项目实施计划》：工作量估计，进度计划，人力资源计划，软/硬件、工具要求，风险管理计划，培训计划，沟通计划，交付工作产品清单等
	6	制订项目计划	《项目实施计划》（有些客户需要《质量保证计划（方案）》、《配置管理计划（方案）》等相关计划）
	7	项目计划评审	按照《项目评审管理规范》的规定，QA组织对《项目实施计划》组织评审，直到通过评审
	8	审批项目计划	《项目实施计划》获得相关领导的审批

软件项目分解的实例

需求分析阶段《需求开发与管理规范》	9	需求调研	开始按照《需求调研计划》，采取《需求调研记录表》进行调研，完成《系统需求分析说明书》初稿
	10	需求分析	如果客户需求不清晰需要密切跟踪，要完成《需求调研记录跟踪矩阵》、《需求不一致项列表》
	11	需求不一致项协商处理	相关修订文档，可能包括《系统需求分析说明书》和《需求不一致项列表》等文件
	12	需求规格说明书完善	《系统需求分析说明书》正式稿、《需求跟踪管理表》
	13	需求验证	需求同级评审相关记录。 验证后的《系统需求分析说明书》、《需求跟踪管理表》
	14	需求分析阶段评审	按照《项目评审管理规范》的规定，QA组织对《需求分析说明书的评审》
	15	里程碑评审（可选）	完成《项目里程碑报告》并组织评审

软件项目分解的实例

分析设计阶段《分析设计管理规范》	16	概要设计	概要设计相关技术资料
	17	设计文档编写	《概要设计说明书》
	18	概要设计评审（可选）	《概要设计说明书》的评审（建议详细设计或概要设计必须做一个正式评审）
	19	详细设计	详细设计相关工具和技术资料
	20	文档编写	《详细设计说明书》
	21	用户界面设计	《用户界面设计说明书》
	22	数据库设计	《数据库设计说明书》
	23	详细设计评审	设计评审记录《项目评审报告》
	24	里程碑评审（可选）	完成《项目里程碑报告》并组织评审
实现开发阶段	25	编程	源代码

软件项目分解的实例

《产品实现管理规范》	26	代码走查	《代码走查检查单》
	27	单元测试	《单元测试报告》
	28	初步完成三大手册	初步完成《系统安装手册》《用户操作手册》《项目维护手册》
测试阶段《项目测试管理规范》	29	集成测试	测试bug清单
	30	测试文档	项目《测试计划》、《测试用例》、《测试报告》
部署运行《系统部署管理规范》	31	部署安装使用	《系统部署用户确认书》需要用户确认
	32	客户培训	《客户培训签到表》《客户培训效果调查表》
验收《项目验收管理规范》	32	内部验收	在正式部署之前完成。《项目内部验收评审报告》
	33	客户验收	《客户验收计划》、《客户验收报告》
结项阶段《项目结项管理规范》	34	结项申请	《结项申请表》
	35	结项总结	《结项总结报告》
	36	总结会议	结项总结
维护阶段《项目运行维护管理规范》	37	维护计划审批	维护工作启动制定《项目维护计划》并通过审批
	38	维护报告	项目结束维护，完成《项目维护总结报告》

本课程的活动定义

阶段名	编号	任务
启动	1	1、组建团队 2、产品范围 3、评审 4、监管
计划	2	1、进度计划 2、质量计划 3、评审 4、监管
需求	3	1、调研 2、需求分析 3、需求报告 4、评审 5、监管
设计	4	1、概要设计 2、详细设计 3、测试计划 4、评审 5、监管
编程	5	1、编程 2、测试 3、监管
交付	6	1、各类文档的编写 2、系统的配置和调试 3、培训

软件协同设计课程之

5.3 工作量估算

2025年1月

上海理工大学
UNIVERSITY OF SHANGHAI FOR SCIENCE AND TECHNOLOGY

1. 软件项目的估算 (1/4)

- 估算：时间、成本、大小
- 方法：
  - 功能点、特性点扩展方法
    - **功能点：MIS适合**
    - **特性点：其它类型的软件。如计算分析软件、嵌入式软件**
  - Cocomo 构建性成本模型
    - **成本驱动因子属性：产品属性、硬件属性、人员属性、项目属性。**
  - 宽带Delphi方法
    - **几个有经验的人估计同样的任务**
    - **需要细致的分解工作**
    - **基于一致意见的迭代过程**

功能点计算举例，适合MIS （2/4）

			权重				
度量	计数	*	简单	平均	复杂	=	
用户输入数	3		3	4	6		9
用户输出数	2		4	5	7		8
用户查询数	2		3	4	6		6
文件数	1		7	10	15		7
外部接口数	4		5	7	10		20
总功能点							50

如果每个功能点100LOC, 则共有
50*100LOC=5kloc

2. COCOMO模型

- **构造性成本模型**Constructive Cost Model
- Boehm1981**年提出，被不断研究改进，是一种参数化的项目估算方法**
- COCOMO**有3套模型**：
  - **基本型**：用估计出的代码行LOC为参数，估算项目工作量
  - **中间型**：考虑产品、硬件、人员、项目等参数的影响，调整项目工作量
  - 详细型：以上全部特征，增加软件工程中分析、设计过程等影响

CoCoMo模型

- 将软件项目类型划分为三类：

项目类型	定　义
组织型	相对较小、较简单的软件项目，对需求不苛刻，开发人员对开发目标理解充分，相关的工作经验丰富，对使用环境熟悉，受硬件约束较少，程序规模不大（<5万行），如多数应用软件、早期的操作系统和编译程序等
嵌入型	软件在紧密联系的硬件、其他软件和操作的限制条件下运行，通常与硬件设备紧密结合在一起，对接口、数据结构、算法要求较高，软件规模任意。如大而复杂的事务处理系统、大型/超大型操作系统、航天用控制系统、大型指挥系统等
半独立型	介于组织型和嵌入型之间，软件规模和复杂性属中等以上，最大可达30万行。如多数事务处理系统、操作系统、数据库管理系统、大型库存/生产控制系统、简单的指挥系统等

Cocomo 构建性成本模型举例 (3/4)
(基本型，只考虑KLOC)

- 项目的人月

$$E=a*KLOC^{b}$$

- 项目开发时间

$$D=c*E^{d}$$

项目复杂度	a	b	c	d
组织性	2.4	1.05	2.5	0.38
半独立性	3.0	1.12	2.5	0.35
嵌入型	3.6	1.2	2.5	0.32

Cocomo 构建性成本模型举例 (4/4)

- 如果KLOC=5， 组织型的项目

$$E=a*KLOC^{b}$$

$$=2.4(5)^{1.05}=13.0\text{（人月）}$$

$$D=c*E^{d}$$

$$=2.5(13)^{0.35}=2.5*2.45=6.13\text{（月）}$$

- 人数

$$N=E/D=13/6.13=2.12$$

3. 中间CoCoMo模型

- 在基本CoCoMo模型基础上考虑了15种影响软件工作量的因素
- 通过工作量调节因子(EAF)修正对工作量的估算，从而使估算更合理
- 人月计算公式如下：$E=a(KLOC)^{b}EAF$
  - 其中：KLOC是软件产品的目标代码行数，单位是千行代码数，a、b是常数，取值如下表所示

项目类型	a	b
组织型	3.2	1.05
半独立型	3.0	1.12
嵌入型	2.8	1.20

工作量调节因子的计算

- 每个调节因子F_i的取值分为很低、低、正常、高、很高、极高六级，正常情况下$F_i=1$
- 当15个Fi选定，可得： $EAF=\prod_{i=1}^{15} F_i$

工作量因素F_i		很低	低	正常	高	很高	极高
产品因素	软件可靠性	0.75	0.88	1.00	1.15	1.40	
	数据库规模		0.94	1.00	1.08	1.16	
	产品复杂性	0.70	0.85	1.00	1.15	1.30	1.65
计算机因素	执行时间限制			1.00	1.11	1.30	1.66
	存储限制			1.00	1.06	1.21	1.56
	虚拟机易变性		0.87	1.00	1.15	1.30	
	环境周转时间		0.87	1.00	1.07	1.15	
人员的因素	分析员能力		1.46	1.00	0.86		
	应用领域实际经验	1.29	1.13	1.00	0.91	0.71	
	程序员能力（软硬件结合）	1.42	1.17	1.00	0.86	0.82	
	虚拟机使用经验	1.21	1.10	1.00	0.90	0.70	
	程序语言使用经验	1.41	1.07	1.00	0.95		
项目因素	现代程序设计技术	1.24	1.10	1.00	0.91	0.82	
	软件工具的使用	1.24	1.10	1.00	0.91	0.83	
	开发进度限制	1.23	1.08	1.00	1.04	1.10	

4. 项目进度管理

- 目标：确保软件项目在规定的时间内完成
- 项目进度管理任务
  - 定义所有的项目任务以及它们之间的依赖关系
  - 规划每个任务所需的工作量和持续时间
  - 制订项目的进度安排
  - 在项目开发过程中不断跟踪项目的执行情况，发现那些未按计划进度完成的任务对整个项目工期的影响，并及时进行调整

4. 项目进度管理

- **制定进度计划的两种情况**
  - 客户方都**规定了明确的交付日期**，此时进度安排必须在此约束下进行
  - **只规定了大致的时间界限**，最终的交付日期由开发组织确定，此时的进度安排可以比较灵活，工作量的分布可考虑对资源的充分利用

5 如何制定软件项目进度

- 任务划分
  - 项目划分成若干可以管理的活动和任务，为了实现项目的划分，对产品和过程都需要进行分解
- 相互依赖性分析
  - 确定各个被划分的活动或任务之间的相互关系，有些任务必须是串行的，有些可能是并行的
  - 采用网络图分析的方法（数据结构中有介绍）

5 如何制定软件项目进度

- 时间分配
  - 必须为每个被调度的任务分配一定数量的时间，此外还必须为每个任务制定开始和结束日期，这些日期是相互依赖的
- 工作量确认
  - 确保在任意时段中分配给任务的人员数量不会超过项目组中的人员数量

5 如何制定软件项目进度

- 定义责任
  - 每个被调度的任务都应该指定某个特定的小组成员来负责
- 定义结果
  - 每个被调度的任务都应该有一个确定的输出结果
- 定义里程碑
  - 每个任务或任务组都应该与一个项目里程碑相关联(当一个或多个工作产品经过质量评审并且得到认可时，标志着一个里程碑的完成)

6.1 系统平台与模式设计

1 系统平台

2 软件平台的设计

3 软件计算模式的设计

4 软件结构的设计

1 系统平台

系统平台是系统开发和运行的环境，包括网络、计算机、相关设备、支撑软件和系统软件等。平台设计需要根据系统设计要求，通过对技术和市场的综合分析，确定出网络结构、设备选型和软件平台方案。

2 软件平台设计

软件平台是系统开发和运行所需的集成软件系统。设计和选择高效、实用、方便、功能齐全的软件平台，对系统开发有着十分重要的意义。

1). 操作系统

操作系统是计算机系统中最重要的系统软件。

Windows server版、unix，linux等

2). 支撑软件

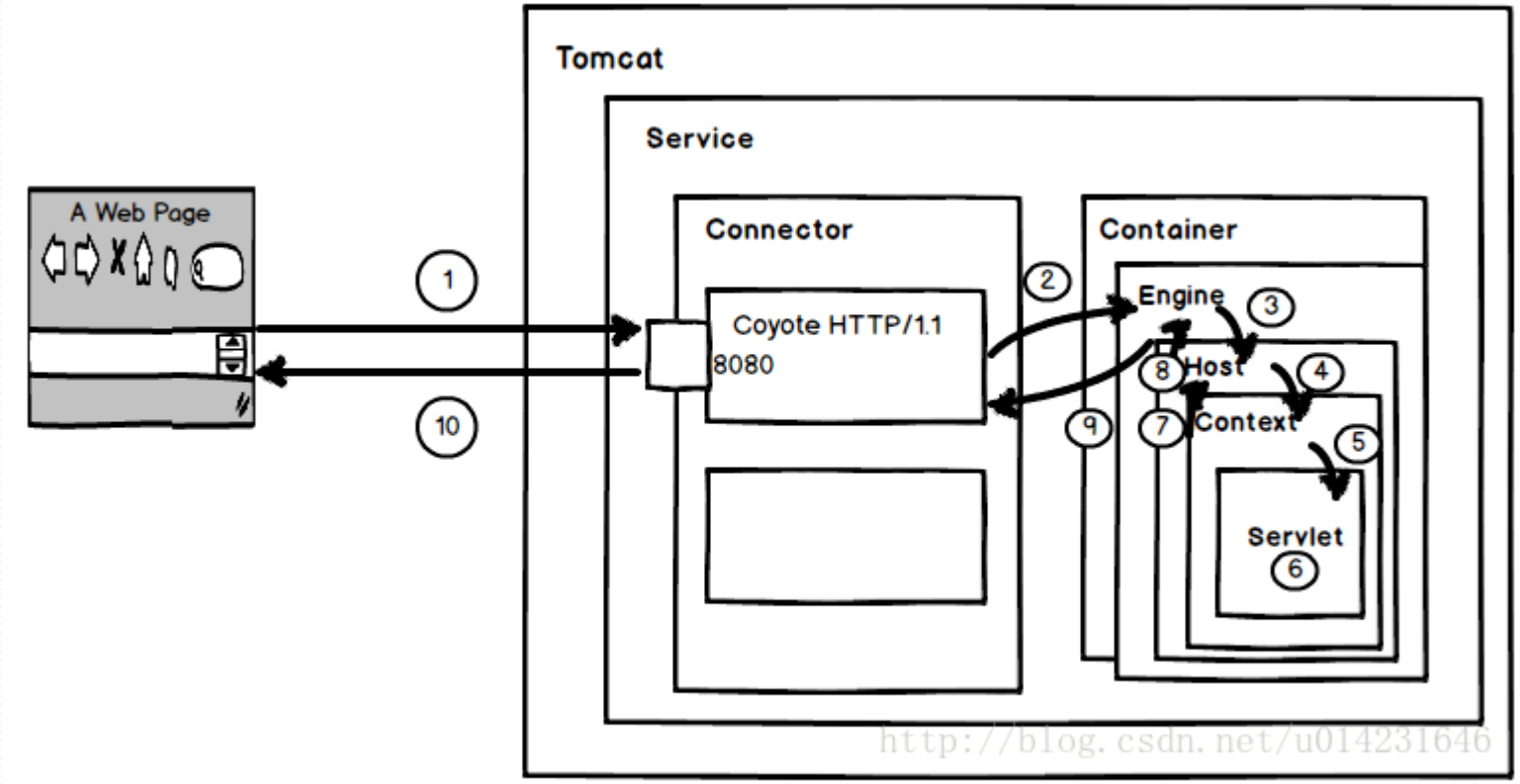
支撑软件是协助人们开发和维护软件的工具和环境软件。编辑程序、数据库系统、集成开发环境等都属于支撑型软件，支撑软件主要包括以下几方面：

（1） 数据库管理系统DBMS

在数据库服务器上的DBMS对数据库实施集中管理，可以并发地处理多个客户机发来的数据处理请求。常见的数据库管理系统有SQL-Server、Oralce、Sybase、Informix、DB2等，系统分析员可以根据自己的需要进行选择。

(2) 应用服务器

- Tomcat（运行JSP 页面和Servlet容器）



(2) 应用服务器

- IIS（Internet Information Server的缩写）是IIS是一种Web（网页）服务组件，其中包括Web服务器、FTP服务器、NNTP服务器和SMTP服务器

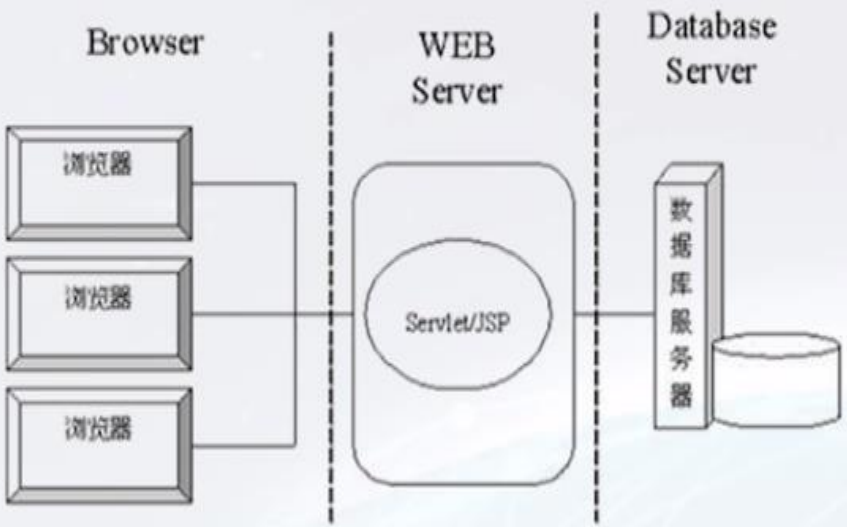
(3) 客户端开发软件

客户端开发软件十分丰富，系统开发人员可以根据设计需要进行选择，选择客户端开发软件要考虑继承性。常见的客户端开发软件有Java等。

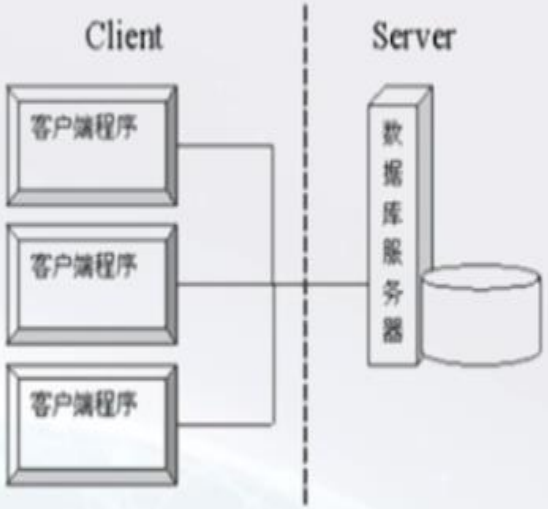
3、系统计算模式设计

系统的计算模式是指通过网络计算机处理信息的方式。常用的模式有：基于客户机/服务器（C/S）、浏览器/服务器(B/S).

B/S系统架构



C/S系统架构



4 软件结构设计

1）概述

软件的软件结构是由软件的各子系统按照确定的关系构成的结构框架。子系统是对软件分解的一种中间形式，也是组织和描述软件的一种方法。由多个子系统构成系统软件，每一个子系统又包括多个用例设计、设计类和接口。

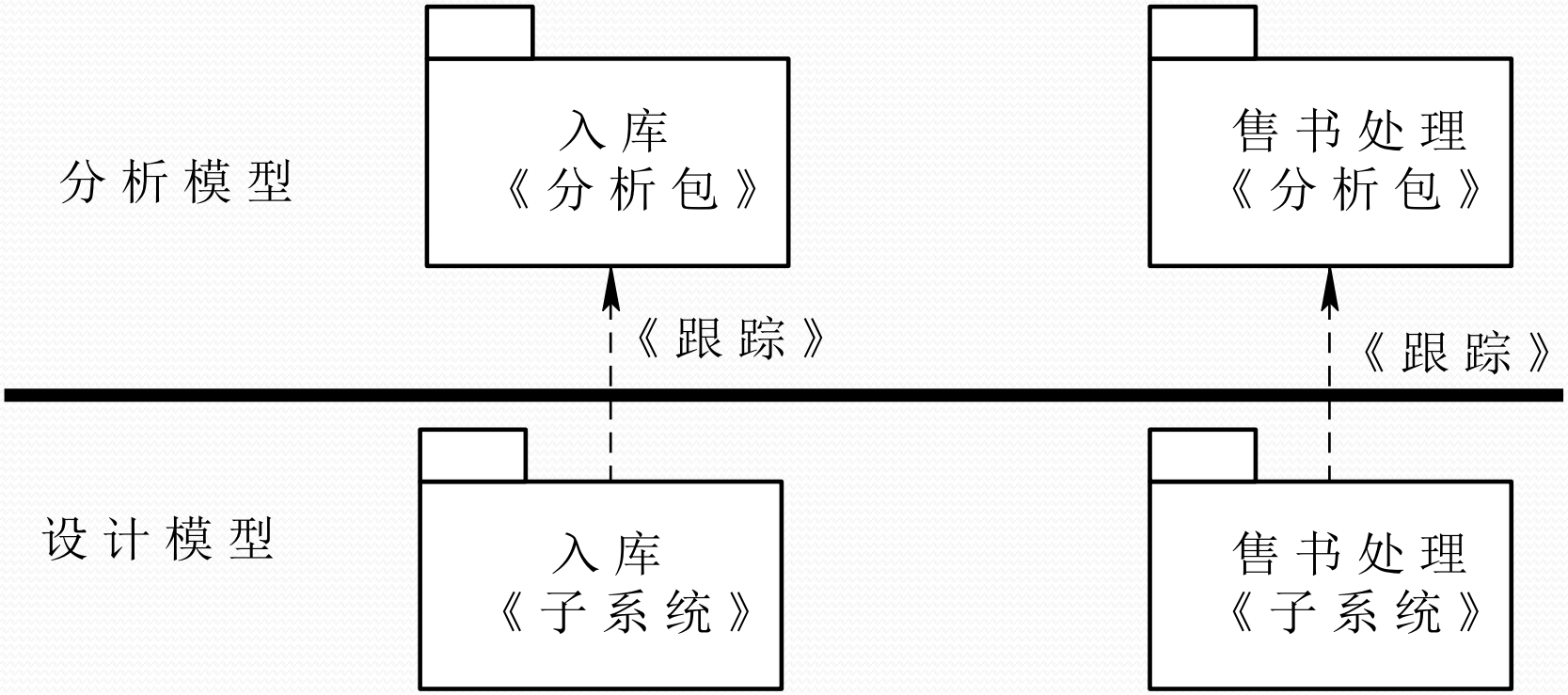
软件结构设计是把软件分解成为多个子系统，并确定出由各子系统及其接口构成的软件结构。

2） 应用子系统设计

（1）识别应用子系统

应用子系统的原型是系统逻辑结构中的分析包。把分析包作为初步的应用子系统，然后，再对各子系统进行分析和优化，以确定应用子系统。

案例：书店分析模型和设计模型



案例：“售书处理”应用子系统的优化：

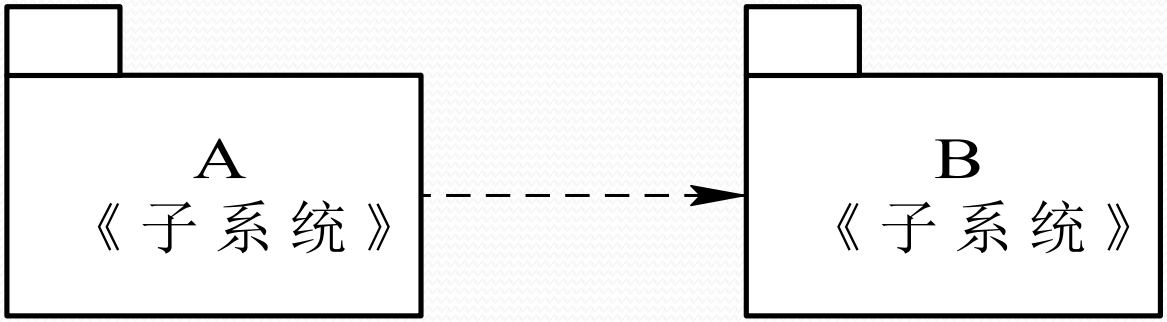
◆第一，规模分析。“售书处理”分析包对应着“售书处理”一个用例，但由于该分析包规模过于复杂，所以需要进行分解以减少其复杂性。可以分解成为四个应用子系统，“售书处理”、“开书单”、“收书款”和“出售图书”。

◆第二，应用分析。“售书处理”、“开书单”、“收书款”和“出售图书”四个子系统均要访问“书目”、“架存图书”、“售出图书”和“职工”四个数据表， 因此，可以再设置“书目管理”、“架存图书管理”、“售出图书管理”和“职工管理”四个子系统。

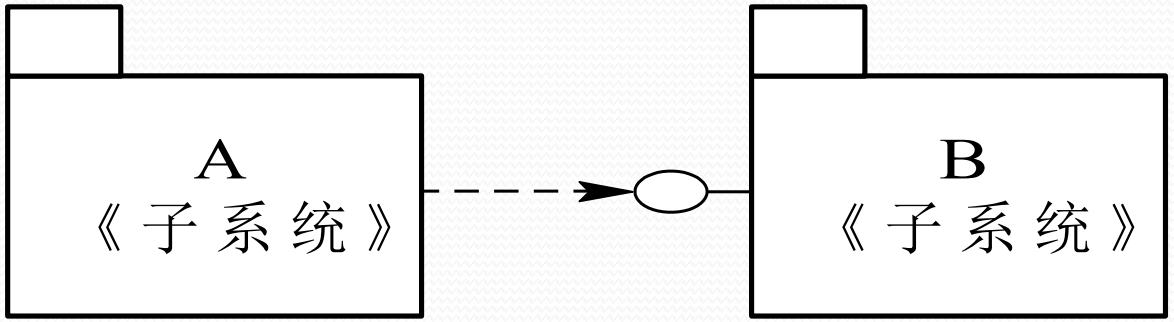
（3） 确定子系统间的接口

当子系统之间存在依赖关系时，子系统之间就存在确定接口。子系统接口定义了外部子系统对本子系统可进行的访问操作集。这些操作由子系统内部的类来提供，或着由子系统中的其它子系统提供。

可以通过子系统之间存在的关系来发现子系统之间的接口。如果子系统A依赖子系统B，则子系统B应该向子系统A提供接口。

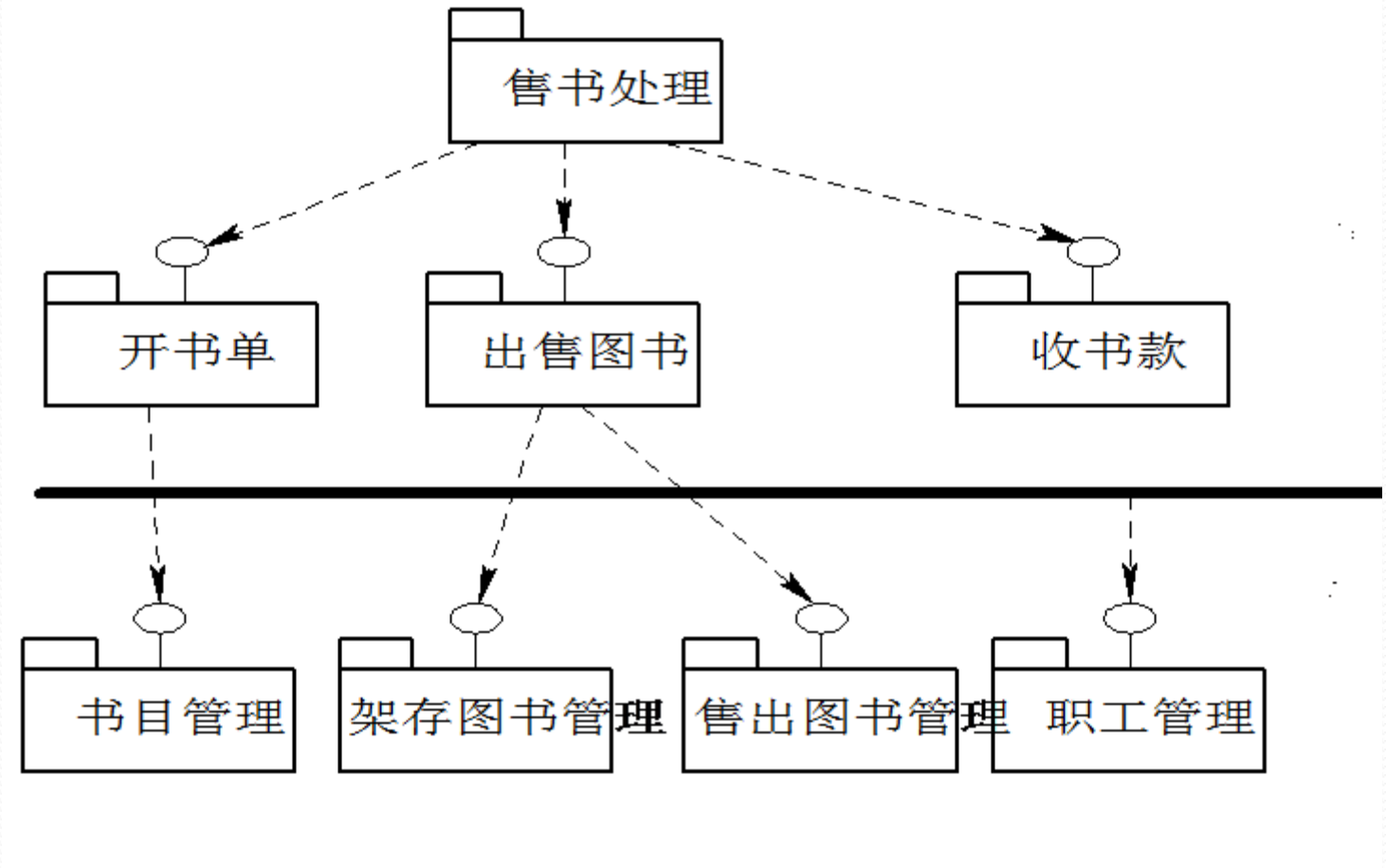


A依赖B



B向A提供接口

根据依赖关系确定接口



书店销售子系统的软件结构图

软件协同设计课程之

6.2 问题域设计

2025年1月

上海理工大学
University of Shanghai for Science and Technology

1 软件总体设计

- 设计方法
  - SA/SD
  - OOA/OOD
- 最富创造性地阶段：
  - 构造问题的解
  - 形成产品的结构 Architecture
- 总体设计/高层设计/概念设计
  - 系统由部件组成
  - 部件之间通信（参数传递/接口）

1 软件总体设计内容

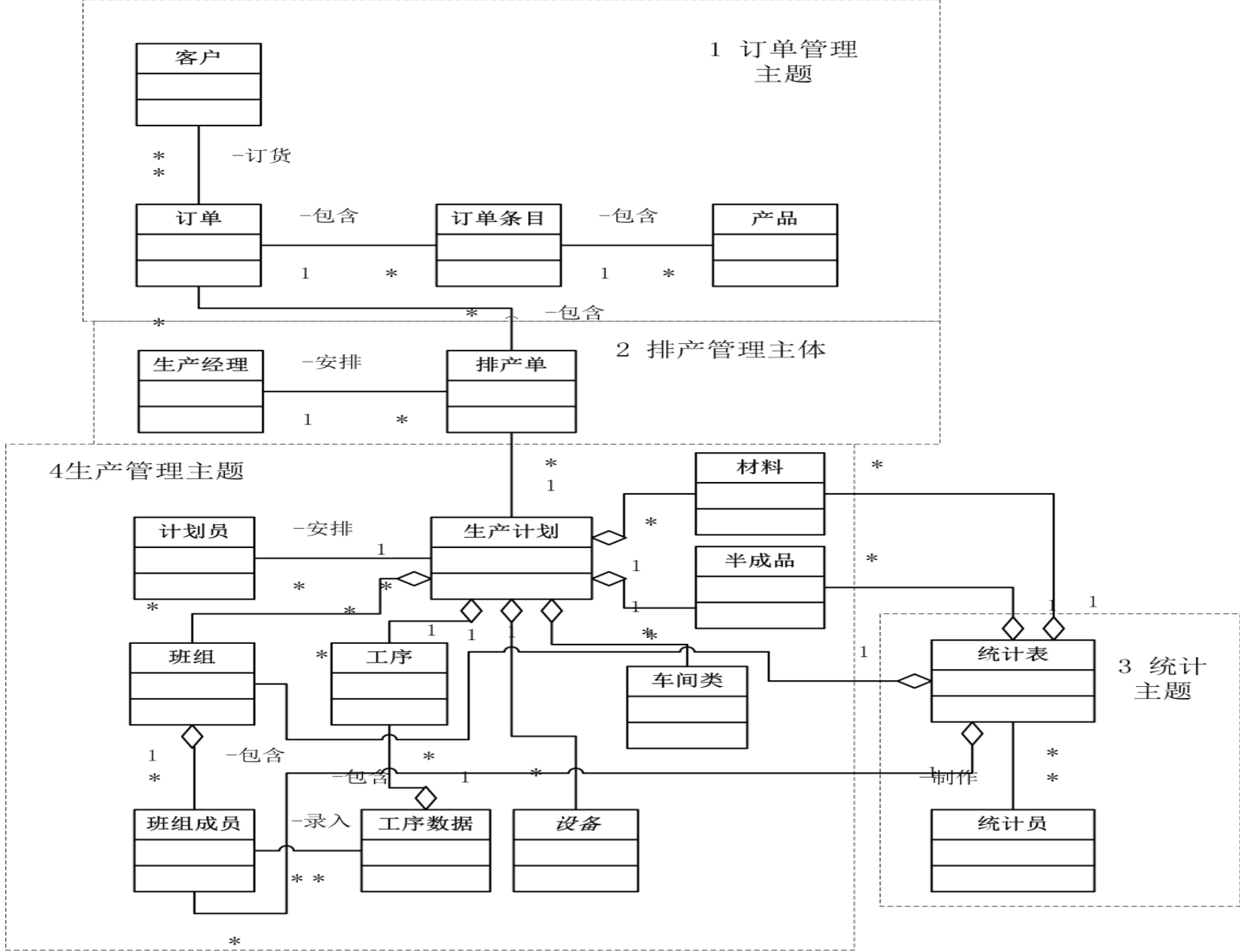
- 硬件、OS、网络
- 软件体系结构：总体架构
- 问题域的设计：功能模块结构图/主题划分的类图
- 数据管理：（文件、数据库？）、类设计
- 软件运行驱动设计
- 用户界面、界面类的设计
- 编程语言
- 可复用组件的设计
- 质量与安全设计

2 传统的结构化设计

- 结构化概要设计
  - 对软件产品进行模块化分解，产生具有期望功能的模块结构
  - 基于DFD图进行设计
- 结构化详细设计
  - 模块的数据结构与算法

3 面向对象的设计

- 问题域类图设计（概要设计）
  - 对需求进行分析理出问题域的类
  - 基于对业务逻辑分析，设计整体类图
    - **生产信息系统主题划分与类设计，见下页**
- 详细设计
  - 对类的设计进一步完善
    - **类的属性**
    - **类的方法：数据结构与算法**


1 订单管理主题
客户
订货
订单
包含
订单条目
包含
产品
包含
2 排产管理主体
生产经理
安排
排产单
4生产管理主题
材料
计划员
安排
生产计划
半成品
班组
工序
车间类
统计表
3 统计主题
包含
包含
制作
班组成员
录入
工序数据
设备
统计员

4. 如何确定类

① 标识候选的对象

② 筛选对象，确定最终对象的类

③ 检查和调整

④ 标识类

4. 如何确定类

（1）标识候选的对象

需求陈述中的名词或名词短语是可能的候选对象，它们以不同的形式展示出来。

- 外部实体

  如其他系统、设备、人员，它们生产或消费计算机系统所使用的信息

- 物件、构造物

  如成绩单、报表、报告、显示、信函、信号，它们是问题信息域的一部分

- 发生的事情或事件

  如性能改变或完成一组机器人移动动作，它们出现在系统运行的环境中

- 角色

  如管理者、工程师、销售员，它们由与系统交互的人扮演

- 组织单位

  如部门、小组、小队，它们与一个应用有关

- 场所

  如制造场所、装载码头，它们建立问题和系统所有功能的环境

- 潜在的类
  - 是否有存储、转换、分析与处理
  - 是否有外部系统

4. 如何确定类

（2）筛选对象，确定最终对象的类

可以用以下选择特征来确定最终的对象：

- 保留的信息

仅当必须记住这些候选对象的信息，系统才能运作时，则该对象可定为候选对象

- 需要的服务

候选对象必须拥有一组可标识的操作，它们可以按某种方式修改对象属性的值

- 多个属性
  - 在分析阶段，关注点应该是“较大的”信息。
  - 仅具有单个属性的对象，最好把它表示为另一对象的属性

4. 如何确定类

（3）检查和调整

- 类的属性和操作不适合该类的全部对象
  - 重新分类
- 属性和操作相同的类
  - 可能合并，要根据实际情况
- 对同一事物的重复描述
  - 学生证类与学生类

4. 如何确定类

（4）标识类

这里介绍一种类—责任—协作者（class—responsibility—collaborator，简称CRC）技术。CRC实际上是一组表示类的索引卡片，每张卡片分成三部分，它们分别描述类名、类的责任和类的协作者

类名：	
类的类型：（如：设备，角色，场所，……）	
类的特征：（如：确切的事物，原子的，并发的，持久的……）	
责任：	协作者：

CRC卡

4. 如何确定类

- 类的责任：即类的操作，大体可分为三类
  - 以某种方式操纵数据的操作
    - 如：增加、删除、重新格式化、选择等
  - 完成某种计算的操作
  - 为控制事件的发生而监控对象的操作

4. 如何确定类

- 协作者
  - 一个类也可和其他类协作来完成某个责任。
- 标识协作者
  - 两个类具有整体与部分关系
  - 一个类必须从另一个类获取信息
  - 一个类依赖于（depends-upon）另一个类，则它们间往往有协作关系

4. 如何确定类

(5)复审CRC卡：基于用例

在填好所有CRC卡后，应对它进行复审。复审应由客户和软件分析员参加，复审方法如下：

1) 参加复审的人，每人拿CRC卡片的一个子集。注意，有协作关系的卡片要分开，即，没有一个人持有两张有协作关系的卡片

2) 将所有用例/场景分类

3) 复审负责人仔细阅读用例，当读到一个命名的对象时，将令牌（token）传送给持有对应类的卡片的人员

4）收到令牌的类卡片持有者要描述卡片上记录的责任，复审小组将确定该类的一个或多个责任是否满足用例的需求。

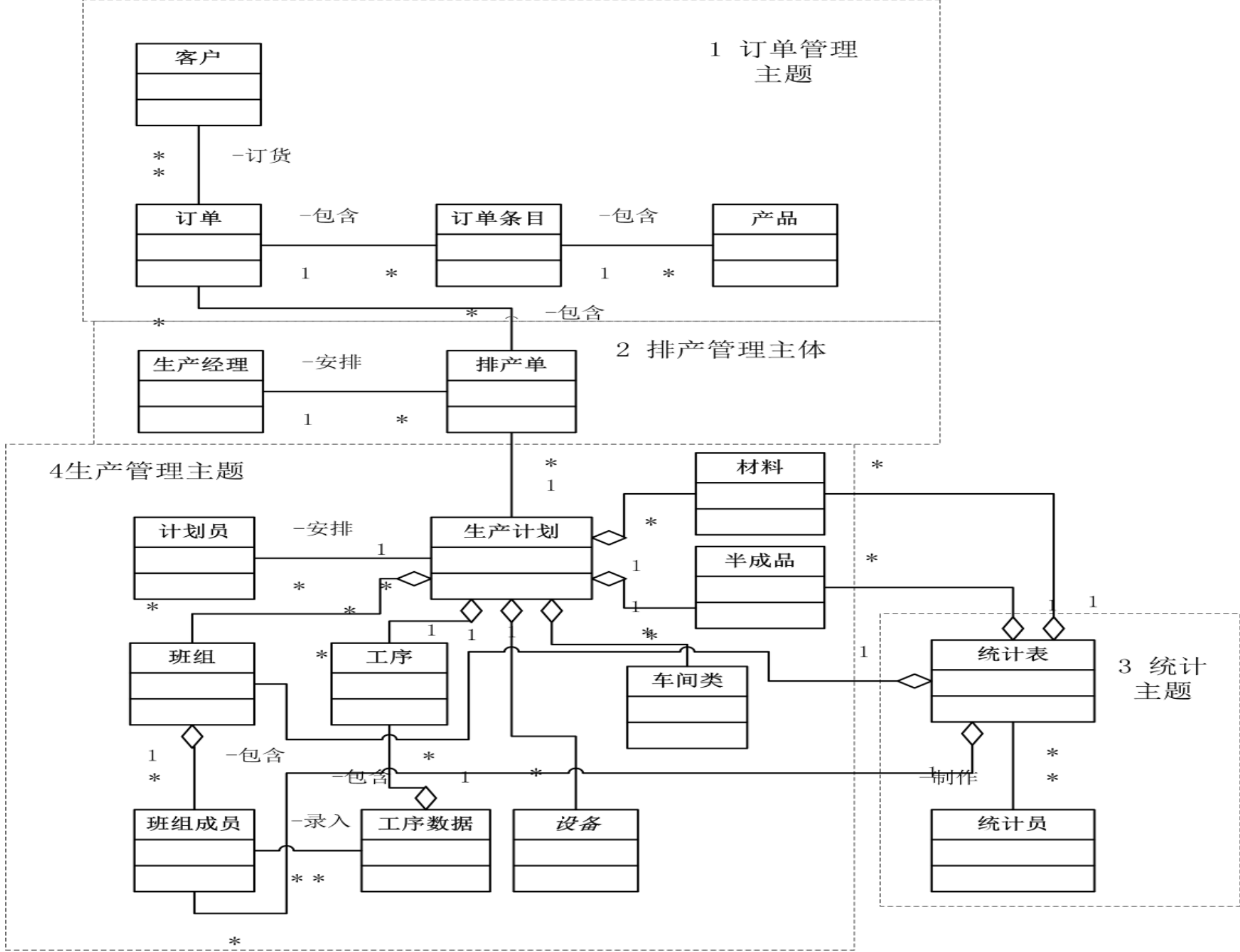
当某个责任需要协作时，将令牌传给协作者，并重复4）

5）如果卡片上的责任和协作不能适应用例，则需对卡片进行修改，这可能导致定义新的类，或在现有的卡片上刻画新的或修正的责任及协作者

这种做法持续至所有的用例都完成为止

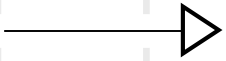
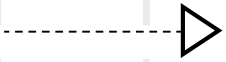
举例

- 项目介绍
- 简化的需求\需求分析0. doc
- 简化的需求\需求分析1. doc
- 简化的需求\需求分析2. doc （介绍）
- 简化的需求\需求分析3. doc

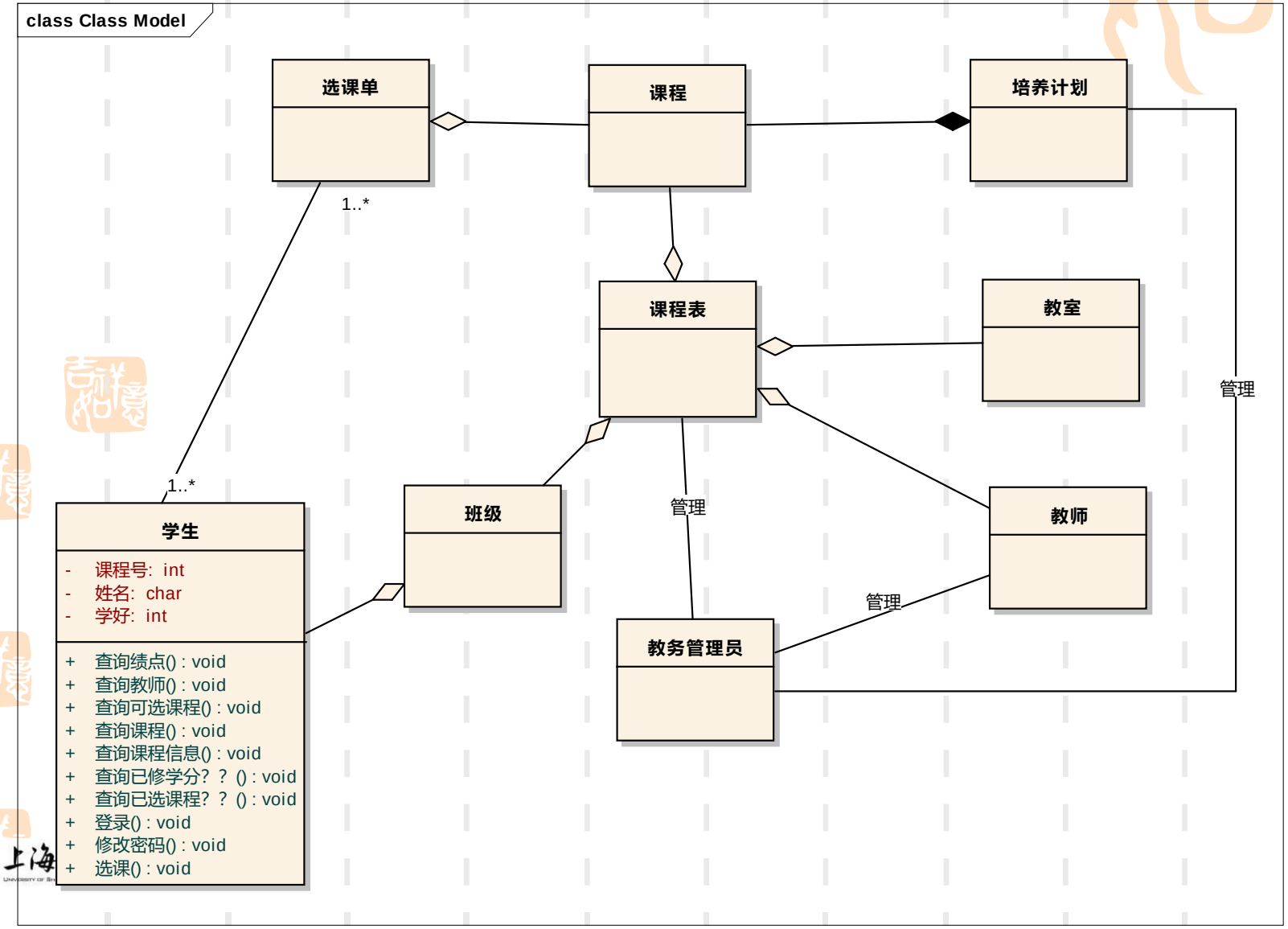

1 订单管理主题
客户
订货
订单
包含
订单条目
包含
产品
包含
2 排产管理主体
生产经理
安排
排产单
4生产管理主题
材料
计划员
安排
生产计划
半成品
班组
工序
车间类
统计表
3 统计主题
包含
包含
制作
班组成员
录入
工序数据
设备
统计员

- **类之间的关系**
  - 依赖关系(using a，一个类中的方法使用另一个类)
  - 关联association: 类之间的关系
    - 一个类的属性类型是另一个类
  - 聚集和组合（has a）
  - 泛化关系（is a）

类之间的关系图形

关 系	功能	符号
关联	类实例间连接的描述	
依赖	二个模型元素之间的一种关系	
泛化	更特殊描述与更一般描述之间的一种关系，用于继承和多态性类型声明	
实现	规约（specification）与它的实现之间的关系	

教务管理系统：选课管理类图



总结

- 面向对象设计：包括若干个侧面
- 问题域设计：基于业务与需求描述，构建问题域类图
- CRC卡片与类的审核

6.3 软件运行驱动设计

1、概述

2、控制类设计

3、时序图

1、概述

软件结构设计完成后，我们就将整个软件分解成子系统，而子系统可以跟踪到分析包，分析包可以跟踪到用例，即一个子系统能够完成它所跟踪的用例的功能，所以接下来设计的第一项工作便是对子系统所跟踪的用例进行软件运行驱动设计。

软件运行驱动设计包括两个含义：一是基于用例设计控制流，二是描述用时序图给出设计的结果。

2. 控制类设计的工作

1) 确定控制类

2) 设计类图

3) 基于用例描述，绘制顺序图

3. 控制类设计过程

1) 问题域的设计已经完成了问题域的类设计，有了静态类模型

2）但程序为了完成一个功能，从哪个对象启动，调用哪些对象的方法，并不清楚，为此需要

- 控制类设计
- 基于用例，绘制用例图

实体类(Entity Class)是信息系统表示客观实体的抽象要素。

边界类(Boundary Class)是描述系统与参与者之间交互的抽象要素。边界类只是对信息系统与参与者之间交互的抽象建模，并不表示交互的具体内容及交互界面的具体形式。

控制类(Control Class)是表示信息系统对其它对象实施协调处理、逻辑运算的抽象要素。（ER图、问题域类体设计）

- **控制流：主动对象的一次执行活动的控制**
- 识别控制流：
  - **从Use case识别**
    - Use case 的每一个功能都可能需要一个控制流来实现
  - **从用户需求出发**根据用户的要求，分析哪些任务可以并行的执行

4 分析用例设计类图

案例：

售书的处理过程：

读者把从书架上选择的图书拿到售书员面前，售书员用条形码扫描仪接收读者所选图书的书号，并启动系统打印出三联书单。读者持书单到收款台交书款，收款员接收书款之后，自己留存一联书单，并在另外两联书单上盖章，交给读者，读者再持交款后的书单到售书员前领取图书。售书员见到盖章后的书单，自己留存一联，然后给图书盖章，并把图书和一联书单交给读者。

4 分析用例设计类图

控制类的构建：售书处理”用例为例来设计类图。

在“售书处理”子系统中，“售书界面”就是运行驱动的发起点（页面也归为类），从该类发起售书操作。该类与“产生待售图书”控制类之间存在关联关系。当“售书界面”类接收一个要出售图书的书号和册数时，就给“产生待售图书”类发送一个消息，启动“产生待售图书”类的“产生待售图书对象”操作，产生一个待售图书对象，记入“待售图书”类中。

5 绘制顺序图

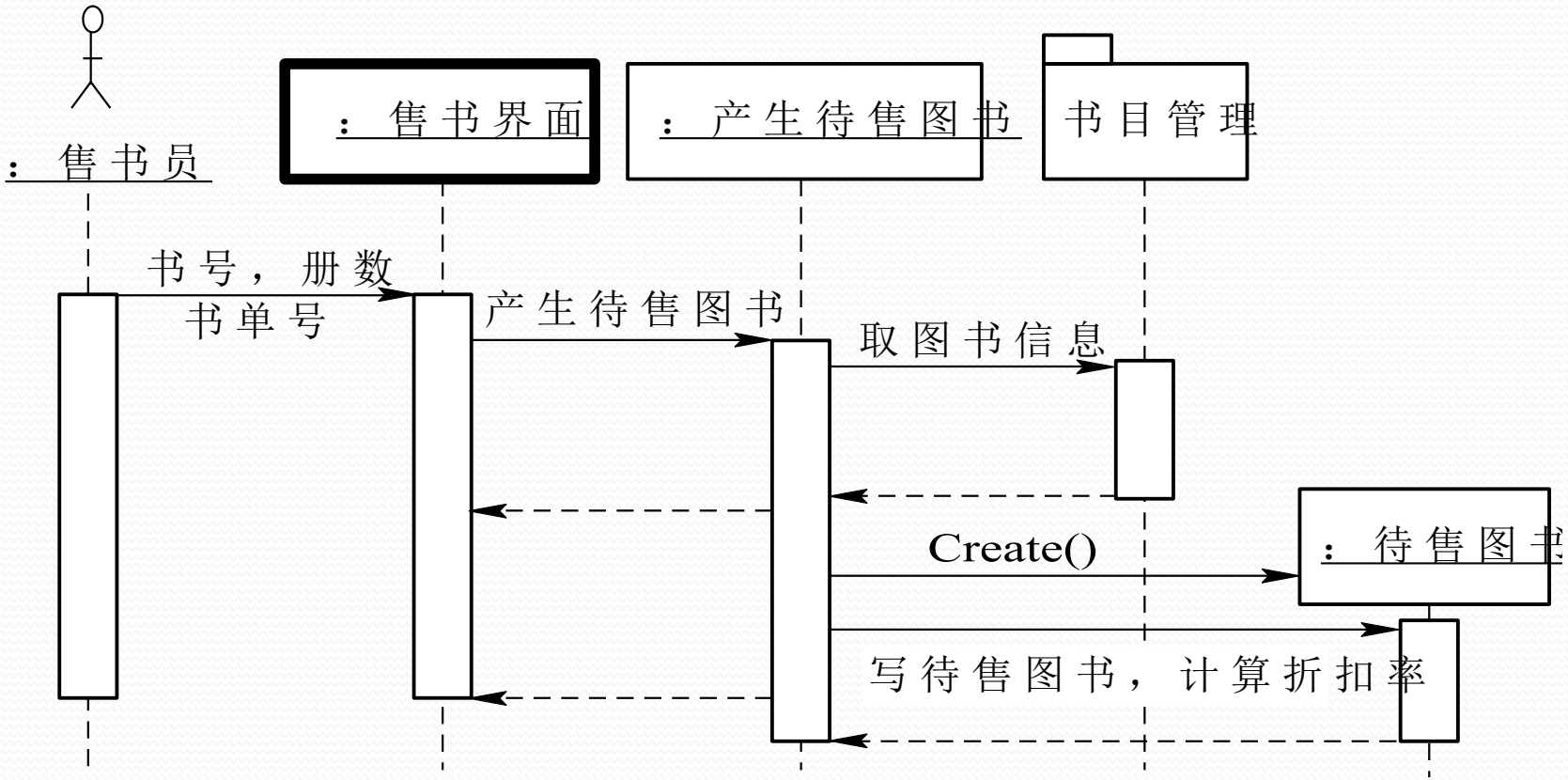
（1）顺序图

顺序图是将交互关系表示为一个二维图。纵向是时间轴，时间沿竖线向下延伸。横向轴代表了在协作中各独立对象。对象用生命线表示。当对象存在时，角色用一条虚线表示，当对象的过程处于激活状态时，生命线是一个双道线。

（2）顺序图的绘制

以“售书处理”用例为设计顺序图

“售书处理”的顺序图。“售书界面”接收售书员输入的读者所要购买图书的书号和册数，同时给该读者产生一个书单号。


：售书员
：售书界面
：产生待售图书
书目管理
书号，册数
书单号
产生待售图书
取图书信息
Create()
：待售图书
写待售图书，计算折扣率

"产生待售图书"顺序图

7.1 测试的意义

1、什么是软件测试

2、为什么要进行软件测试？

3、软件测试工作范畴

1、软件测试的定义

软件测试是由“验证（Verification）”和“有效性确认（Validation）”活动构成的整体

- “验证”是检验软件是否已正确地实现了产品规格书所定义的系统功能和特性
- “有效性确认”是确认所开发的软件是否满足用户真正需求的活动。

2、为什么要进行软件测试？

(1) 软件总存在缺陷。只有通过测试，才可以发现软件缺陷。也只有发现了缺陷，才可以将软件缺陷从软件产品或软件系统中清理出去。

(2) 软件中存在的缺陷给我们带来的损失是巨大的，这也说明了软件测试的必要性和重要性

悲剧

- 放射性治疗仪Therac-25中的软件存在缺陷，导致几个癌症病人受到非常严重的过量放射性治疗，其中4个人因此死亡
- 当爱国者导弹防御系统的时钟累计运行超过14小时后，系统的跟踪系统就不准确。从而导致拦截伊拉克飞毛腿导弹的几次失败，其中一枚在沙特阿拉伯的多哈爆炸的飞毛腿导弹造成28名美国士兵死亡

(3)测试和质量保证的关系

软件质量保证（Software Quality Assurance，SQA）活动是通过对软件产品有计划的进行评审和审计来验证软件是否合乎标准的系统工程，通过协调、审查和跟踪以获取有用信息，形成分析结果以指导软件过程。

- ✧ 对软件工程各个阶段的进展、完成质量及出现的问题进行评审、跟踪。
- ✧ 审查和验证软件产品是否遵守适用的标准、规程和要求，并最终确保符合标准、满足要求。
- ✧ 建立软件质量要素的度量机制，了解各种指标的量化信息，向管理者提供可视信息。

SQA活动

- 技术方法的应用
- 正式技术评审的实施
- 软件测试
- 标准的执行
- 修改的控制
- 度量
- 质量记录和记录保存

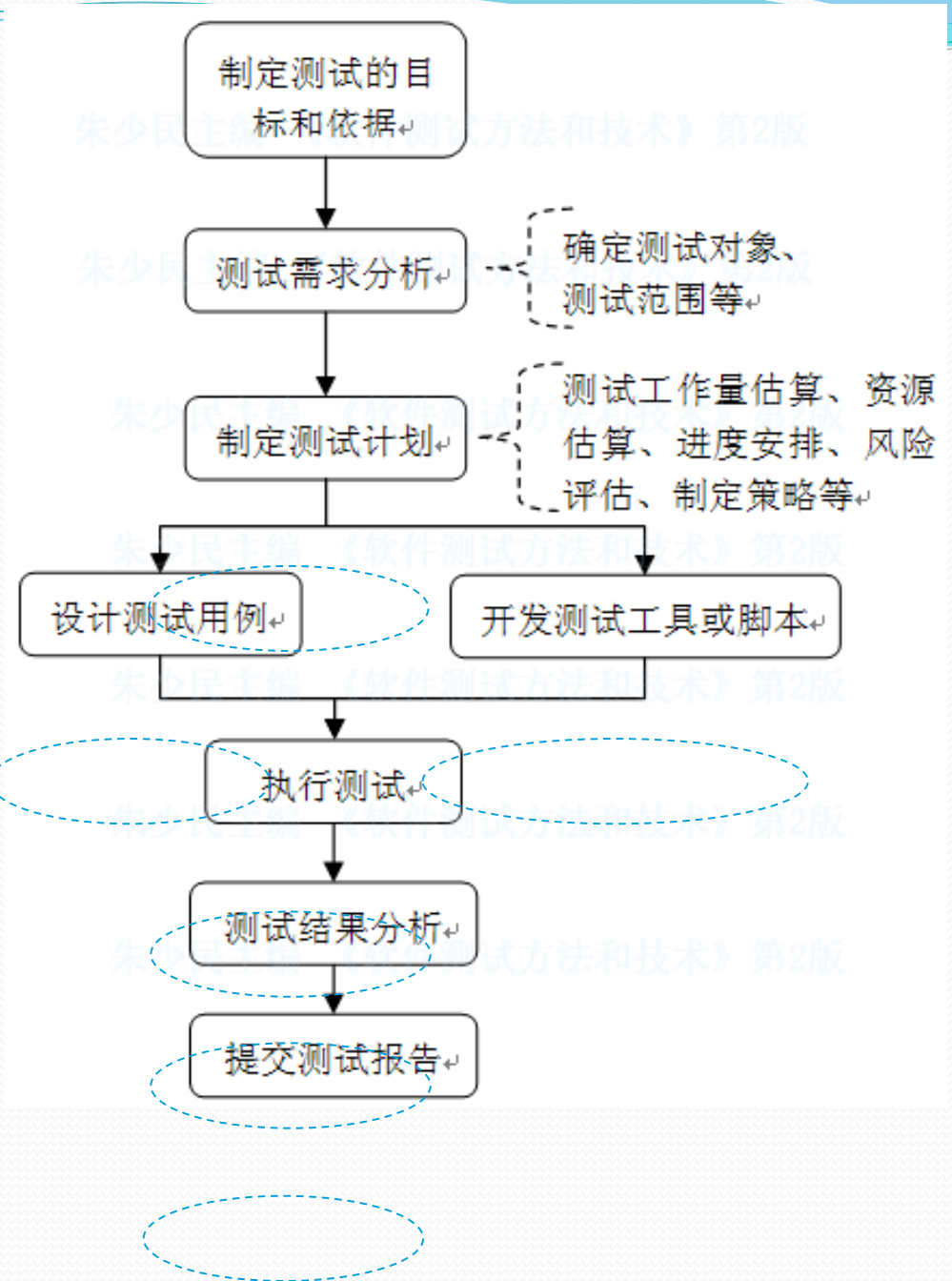
测试 vs. SQA

- SQA指导、监督软件测试的计划和执行，督促测试工作的结果客观、准确和有效，并协助测试流程的改进。
- 软件测试是SQA重要手段之一，为SQA提供所需的数据，作为质量评价的客观依据。

- SQA是一项管理工作，侧重于对流程的评审和监控
- 测试是一项技术性的工作，侧重对产品进行评估和验证

3、软件测试工作范畴

- 软件测试工作的组织：制定测试策略、测试计划，确认所采用的测试方法与规范，控制测试进度，管理测试资源。
- 测试工作的实施：编制符合标准的测试文档，搭建测试环境，开发测试脚本、与开发组织协作实现各阶段的测试活动


制定测试的目标和依据
测试需求分析
确定测试对象、测试范围等
制定测试计划
测试工作量估算、资源估算、进度安排、风险评估、制定策略等
设计测试用例
开发测试工具或脚本
执行测试
测试结果分析
提交测试报告

测试团队

	程序经理	开发工程师	测试工程师	测试/开发
Exchange server 2000	约 25 人	约 140 人	约 350 人	2．5：1
Windows 2000	约 250 人	约 1700 人	约 3200 人	1.9:1

对测试人员的要求

- **技术，编程能力**
- 责任感、耐力
- 沟通能力、理解能力
- 分析问题能力（批判性思维）
- 项目管理能力
- 组织能力
- ……

优秀测试工程师的素质

- 高度的责任感
- 非常好的沟通能力、幽默感
- 技术能力、自信心、耐心
- 怀疑一切的精神、勤奋精神
- 洞察力、适度的好奇心
- 反向思维和发散思维能力、记忆力
- 自我学习能力、创新能力等

看一则招聘（2022）：

【招聘方】上海华为研究所
【岗位名称】软件验证工程师（专业类）
【岗位职责】
1、对产品软件质量进行把关，理解产品设计原理、实现过程；
2、编写软件测试工具，开发测试自动化脚本；执行软件测试，分析测试数据，输出测试报告；
3、对测试中的问题进行分析和定位，与开发人员一起寻求解决方案；
4、对负责的测试对象进行质量评估，对测试结果进行总结和统计分析。
【岗位要求】
1、目标院校：上海海事大学，上海理工大学，东华大学，安徽大学
2、22届毕业生，本硕均可。计算机、软件工程、通信工程、电子信息、自动化等相关工科专业背景；
2、对无线、IP、光传输、数据存储、云计算等某一领域的知识有一定了解，熟悉通信网络基础知识优先；
3、具备基本的软件编程能力；
4、熟悉相关网络协议，如TCP/IP或3GPP等；
5、熟悉测试基础理论者优先，如黑盒和白盒测试方法；
6、具备良好的表达和沟通能力及团队协作能力。
【工作地点&薪酬】上海华为研究所，薪资面谈
【联系方式】19514707743，苗女士，微信号同手机号