


软件工程

大连理工大学软件学院



第1章 软件工程概述

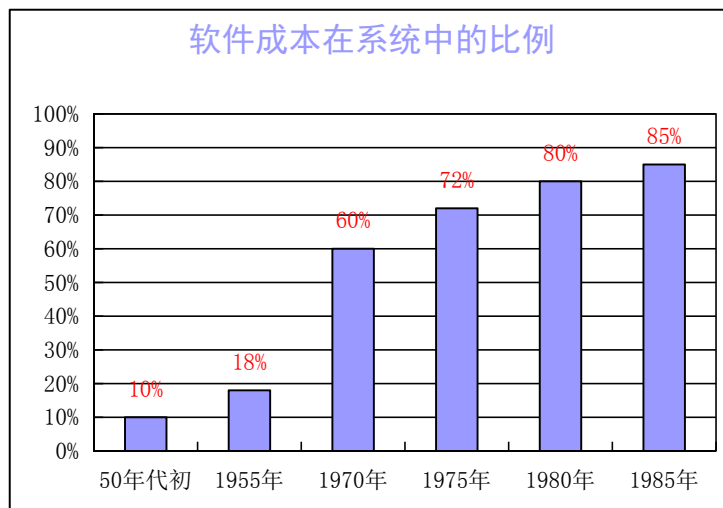
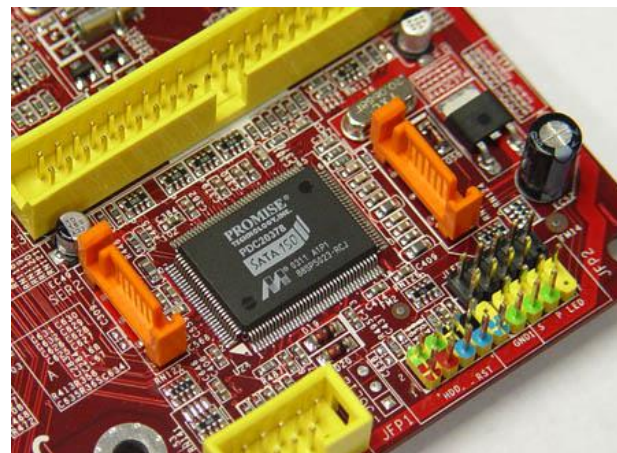


本章主要介绍软件危机的产生以及软件工程的由来、软件工程包括的主要内容以及软件开发的主要方法及技术。

软件危机

• 软件危机的介绍

- 硬件和软件发展的不平衡，硬件性能的发展极其迅速，给软件提出了更高的要求
- 软件开发和维护成本越来越大，令人吃惊地高
- 失败的软件开发项目屡见不鲜



• 什么是软件危机

- 软件危机是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

软件危机

- 软件危机的表现
 - 软件成本日益增长
 - 开发进度难以控制
 - 软件质量差
 - 开发过程无法有效介入和管理
 - 代码难以维护等
- 软件危机的原因
 - 技术原因
 - 软件规模越来越大
 - 软件复杂度越来越高
 - 管理原因
 - 软件开发缺乏正确的理论指导，过分依靠个人技巧和创造性
 - 对用户需求没有完整准确的认识
- 如何克服软件危机： 软件工程





客户是这样解释的



项目经理是这样理解的



架构师是这样设计的



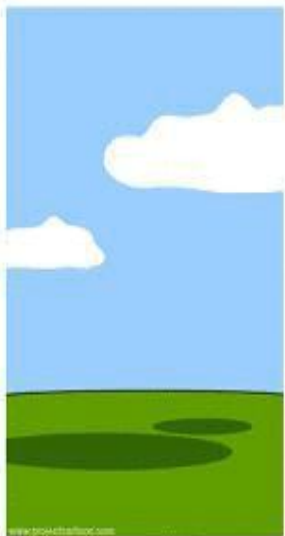
程序员是这样编写的



测试人员收到的是这样的



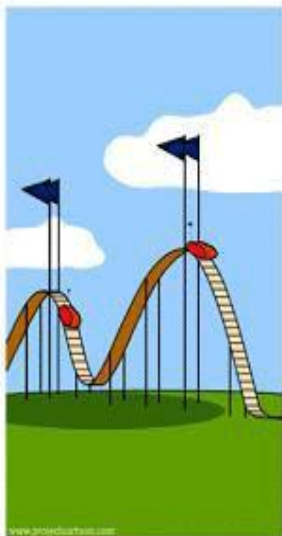
商务顾问是这样描述的



项目文档是这样编写的



软件是这样安装的



客户是这样付款的



技术支持是这样服务的



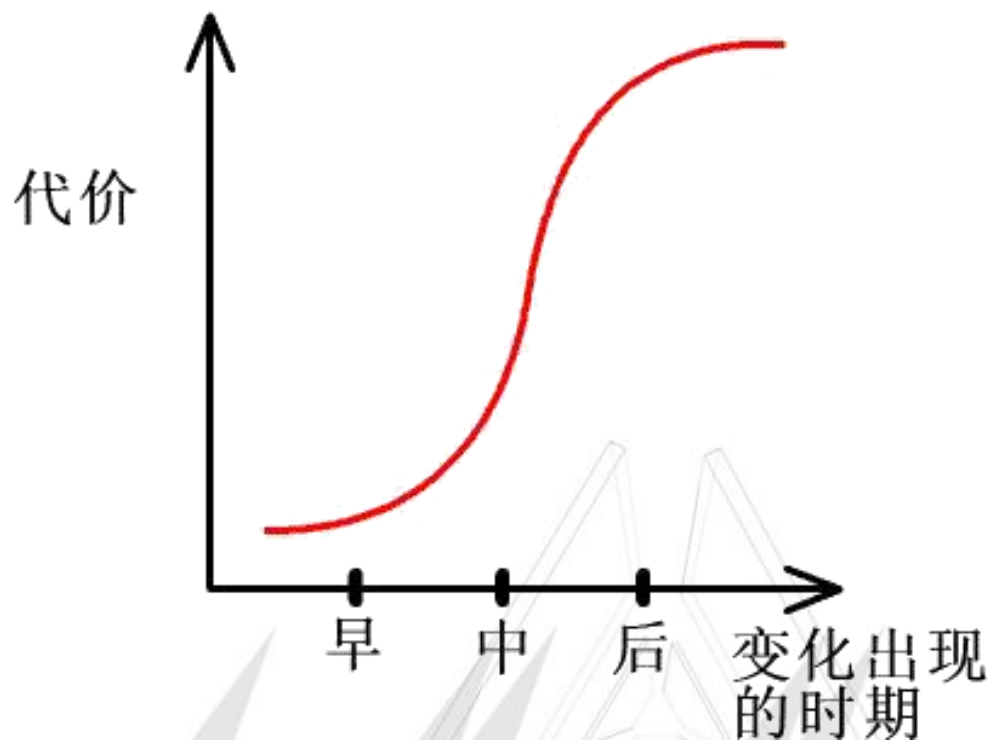
市场是这样推广的



客户真正的需求是这样的

问题在哪里？

- 软件开发链条的“放大”作用。(规范每个环节)
- 只有早期发现问题，才会尽量减少损失。(失之毫厘，谬以千里)
- 但客观规律：用户的牙膏不会一下子挤完。(静态开发方法“天生”会延迟问题的发现)



引入同一变动付出的代价随时间变化的趋势

消除软件危机的途径

- 对计算机软件正确认识。
- 推广使用开发软件成功的**技术和方法**，研究探索更好更有效**的技术和方法**，消除错误概念和做法。
- 开发和使用更好的**软件工具**。
- 对于时间、人员、资源等需要引入更加合理的**管理措施**。
- 软件工程正是从**技术和管理**两方面研究如何更好地开发和维护计算机软件的一门新兴学科。

无章法（个人英雄主义）  工程项目管理模式（团队合作开发）

软件工程知识体系

- 软件工程（IEEE）

- 1968年秋，提出软件工程

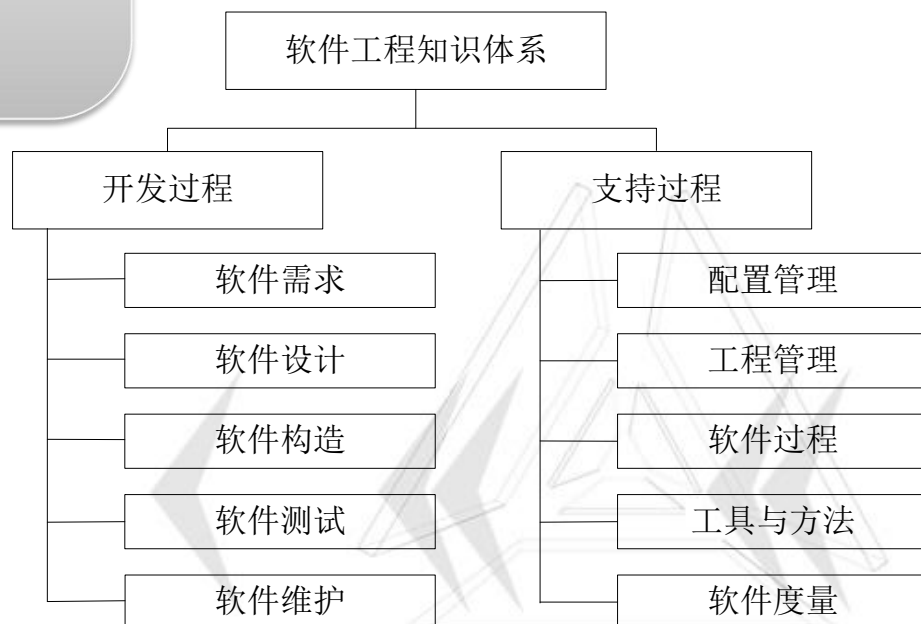
- 1) 将系统化、规范化、可量化的工程原则和方法，应用于软件的开发、运行和维护；

- 2) 对其中方法的理论研究。


- 主要目标：高效开发高质量软件，降低开发成本。

- 软件工程知识体系指南

- SWEBOK指南的目的是为软件工程学科的范围提供一致的确认。



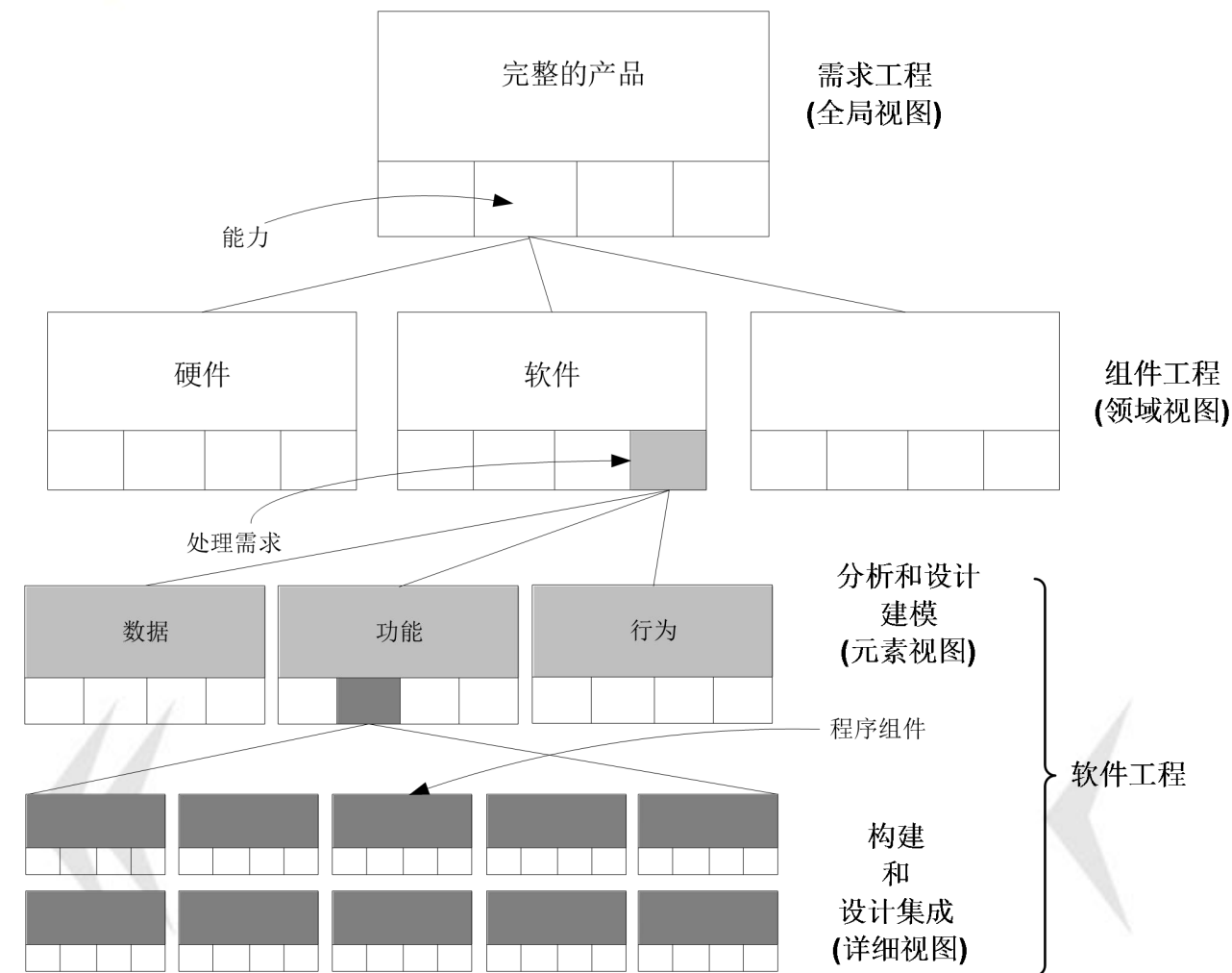
• 软件工程基本原理 (开发与维护的指导)

- 
1. 用分阶段的生命周期计划严格管理
 2. 坚持进行阶段评审
 3. 实行严格的产品控制
 4. 采用现代程序设计技术
 5. 结果应能清楚地审查
 6. 开发小组的人员应该少而精
 7. 承认不断改进软件工程实践的必要性

系统工程

- 系统工程是为了更好地达到系统目标，对系统的构成要素、组织结构、信息流动和控制机构等进行分析与设计的技术。
- 针对不同的领域，系统工程有着不同的实现方法，如商业过程工程（Business Process Engineering）、产品工程（Product Engineering）等。
- 系统工程的目的是使得人们能够确保在正确的时间使用了正确的方法在做正确的事情。

系统分析方法



系统分析的常用方法是层次分析方法，将问题分解为不同的组成因素，并按照因素间的相互关联影响以及隶属关系将因素按不同层次聚集组合，形成一个多层次的分析结构模型。

统一建模语言

- 统一建模语言（UML）提供了一整套对系统建模的基础设施，包括模型的表示及建模的方法等，可以适用不同的系统层次。
- 统一建模语言顾名思义它是一种语言，或者说是一种工具，而不是一种方法。
- 统一建模语言将软件开发中的语言表示与过程进行了分离，具有重要的功能：可视化（Visualization）、规格说明（Specification）、构造（Constructing）和文档化（Documenting）。

UML、代码与自然语言

- UML是提供给用户高层建模的方法，是针对用户需求的高层抽象，具体表现为描述组件的构成及联系，给人一种高屋建瓴的效果。
- 代码也能够描述一种模型，但是具体的、底层的，如果其他人想要了解设计思想，必须要读懂代码，甚至可能要先去学习一门新的语言。
- 另一种表示方法是使用自然语言的叙述，但自然语言具有歧义及含糊不清的弱点。

UML的构成及视图

- 用例图
- 活动图
- 类图
- 对象图
- 顺序图
- 通信图
- 时序图
- 交互概况图
- 组成结构图
- 组件图
- 包图
- 状态机图
- 部署图

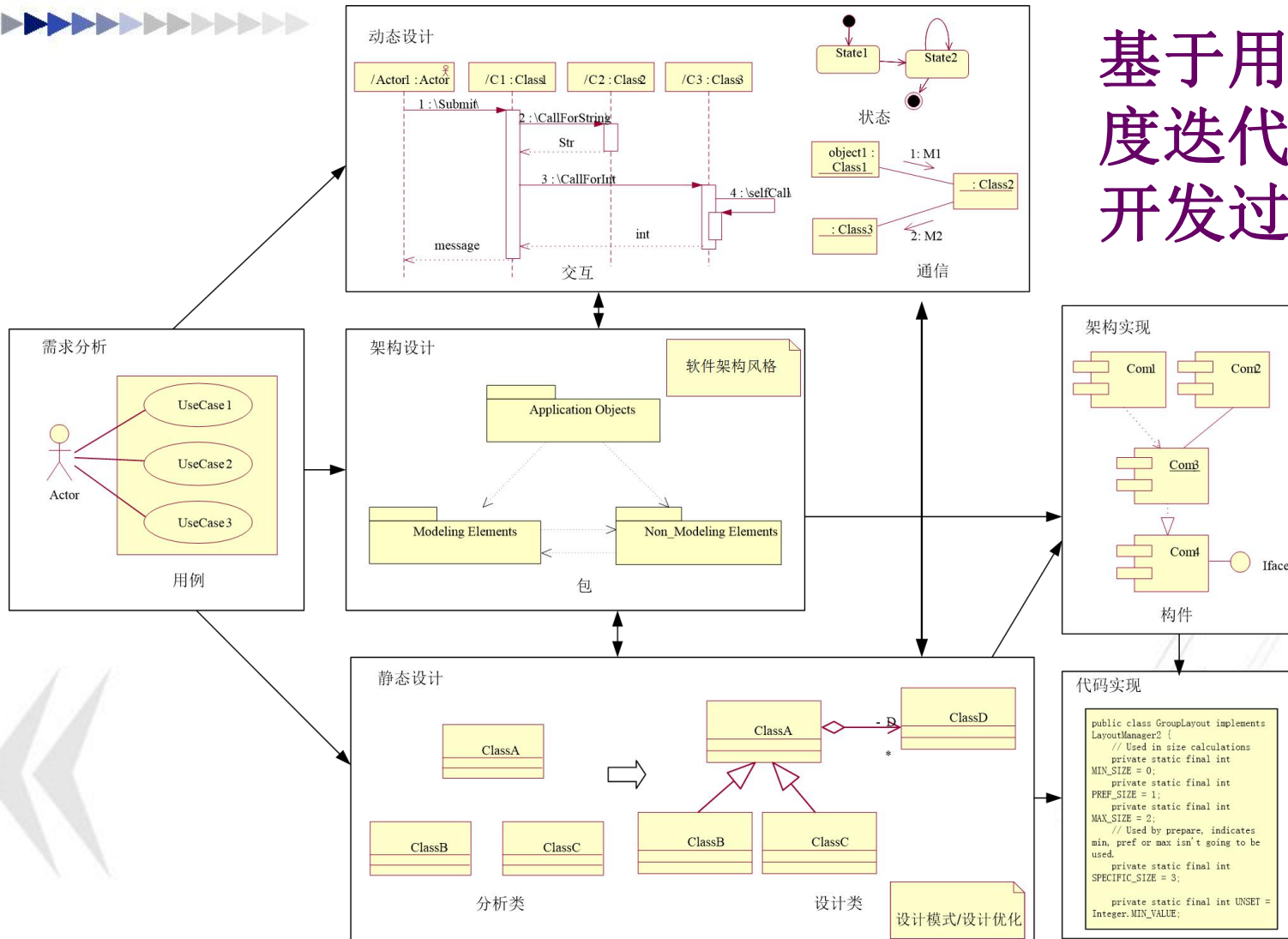
描述系统的所有元素的集合及元素之间的关系，构成了模型。图是通往模型的窗口，特定的图会显示模型的某些部分，但不必显示模型的全部，构成了UML的视图，每个视图捕获系统的一个方面。

Kruchten的4+1视图：

1. 逻辑视图
2. 进程视图
3. 开发视图
4. 物理视图
5. 用例视图

系统开发的解空间

基于用例的高度迭代的软件开发过程



基于UML的系统开发过程

- 需求分析
 - 使用参与者和用例描述系统功能性需求，每个用例要开发一个叙述性描述的用例规约
 - 用户的输入和主动的参与是必不可少的
- 分析和设计
 - 迭代式开发系统的静态和动态模型
 - 静态模型定义了问题域类之间的结构、关系，这些类及其关系描绘在类图中
 - 开发动态模型来实现来自需求模型的用例，以显示每个用例中参与的对象以及对象间是如何交互的，对象和它们之间的交互描绘在通信图或者顺序图中。

基于UML的系统开发过程

- 软件架构建模
 - 设计系统的软件体系结构。
 - 问题域的分析模型被映射到设计模型的解空间中，提供包和子系统的组织准则来将系统组织为子系统。子系统被视为聚合或者复合对象。
 - 设计每个子系统。对于顺序系统，重点放在信息隐藏、类和继承的面向对象概念。对于并发系统的设计，如实时、客户端/服务器或分布式应用，除了考虑面向对象概念，还需要考虑并发任务的概念。
- 实现
 - 将软件架构模型映射到可部署运行实现的模型解空间中。在此阶段中，包和子系统映射生成为组件，而其中的类则使用代码生成方法和优化策略生成可运行和部署的面向对象结构的代码。

软件开发方法

– 软件工程三个要素：方法、工具和过程。

- 方法是完成软件开发各项任务的技术，回答“如何做”；
- 工具是为方法的运用提供自动或半自动软件支撑环境，回答“用什么做”；
- 过程是为获得高质量的软件要完成的一系列任务的框架，规定完成各项任务步骤，回答“如何控制、协调、保证质量”。



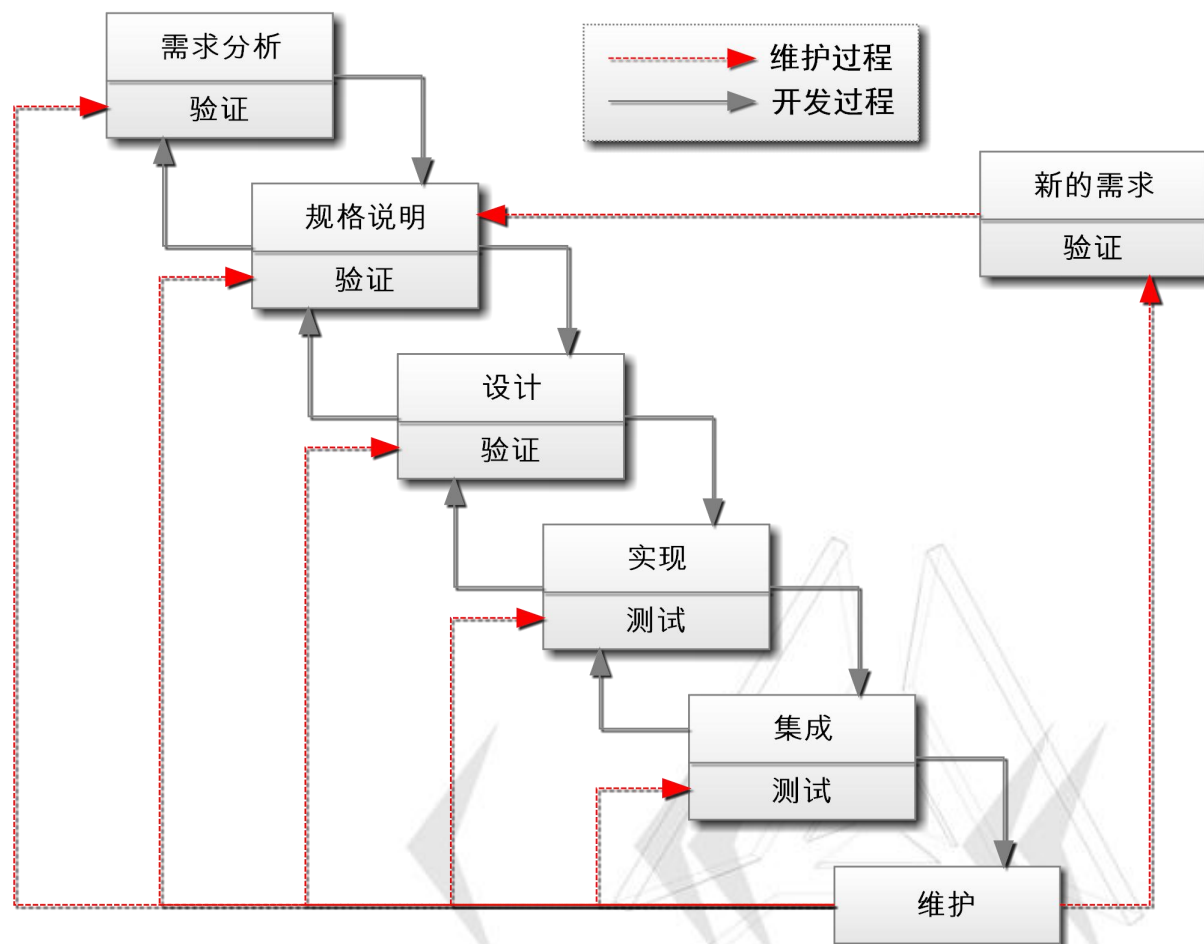
随着软件工程的发展及技术的不断进步，软件开发方法从人们分析问题角度和方式的不同，可以笼统的分为传统的和面向对象的两种。

传统方法

- 传统开发方法又称为结构化方法，是一种静态的思想。
 - 将软件开发过程划分成若干个阶段，并规定每个阶段必须完成的任务，各阶段之间具有某种顺序性；
 - 体现出对于复杂问题“分而治之”的策略，但主要问题是缺少灵活性，缺少应对各种未预料变化的能力，这些变化却是在实际开发中无法避免的；
 - 当软件规模比较大，尤其是开发的早期需求比较模糊或者经常变化的时候，这种方法往往会导致软件开发的不成功或维护困难。

传统方法的特点

- 生命周期模型
- 软件过程划分为若干个阶段
- 每个阶段有各自的任務
- 阶段之间有某种顺序性



Everything is
Object.

面向对象方法

1. 对象作为融合数据及在数据之上的操作行为的统一的软件构件。
2. 把所有对象都划分成**类(Class)**。每个类都定义了一组**数据**和一组**操作**。

数据：静态
操作：动态
3. 按照父类(或称为基类)与子类(或称为派生类)的关系，把若干个相关类组成一个**层次结构**的系统(也称为类等级)。在类等级中，下层派生类自动拥有上层基类中定义的数据和操作，称为**继承**。
4. 对象彼此间仅能通过发送消息互相联系—**封装性**。

OO特点

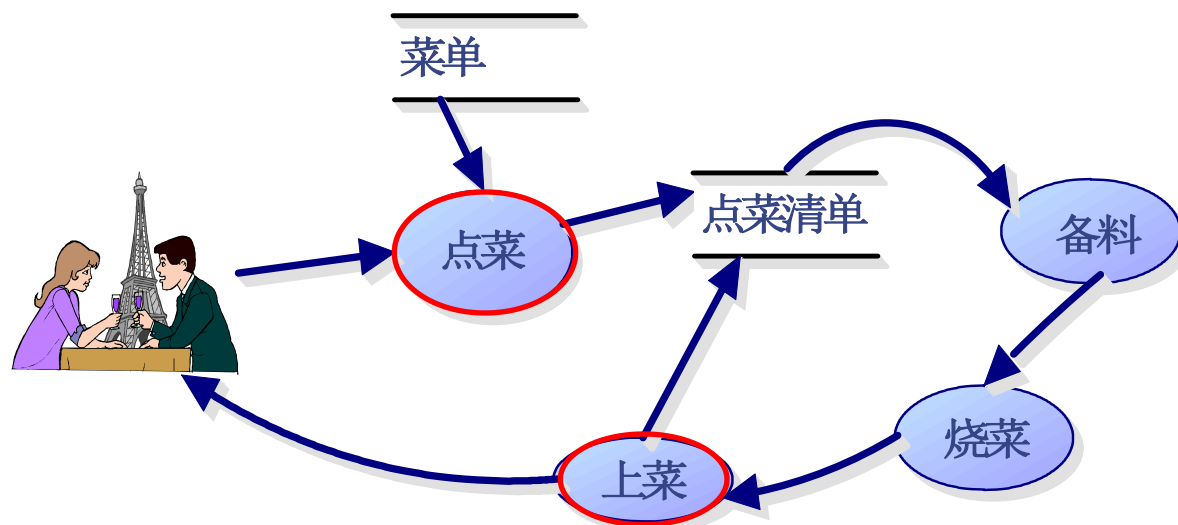
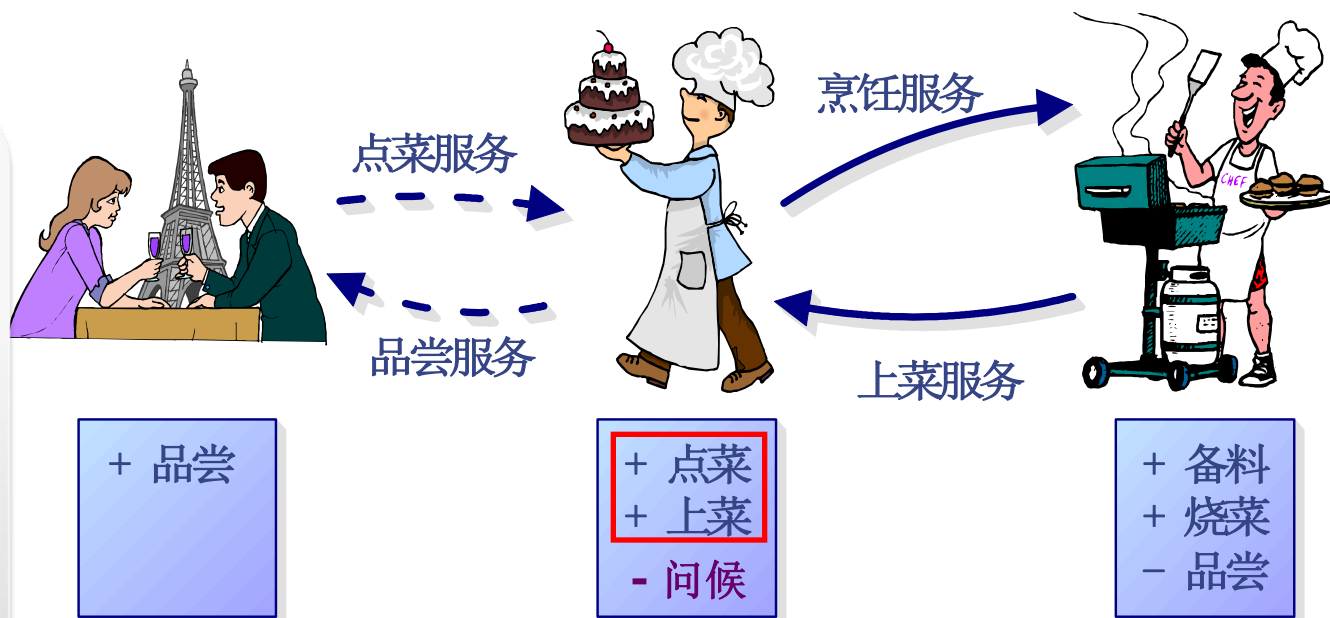


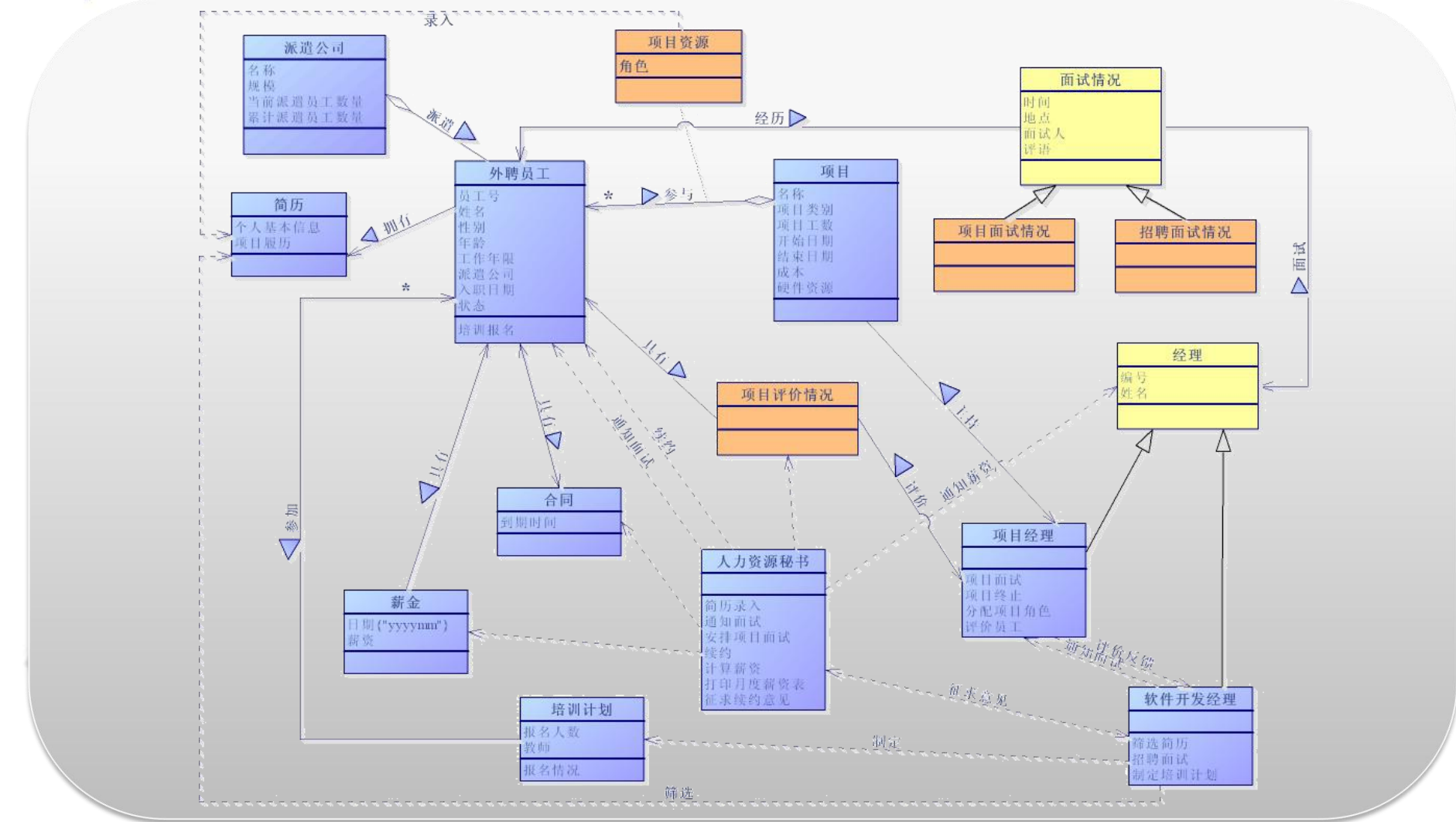
- 面向对象方法学的出发点和基本原则，是尽可能**模拟人类习惯的思维方式**。
- 用面向对象方法学开发软件的过程，是一个**主动地多次反复迭代**的演化过程。
- 概念和表示方法上的一**致性**，阶段间平滑（无缝）过渡。
- 特殊到一般的**归纳**思维过程；一般到特殊的**演绎**思维过程。（**继承的思想**）

OO特点 (2)

- 最终产品中的对象与现实世界中的实体相对应，降低了**复杂性**，提高了**可理解性**，简化了软件的开发和维护工作。
- 对象是相对独立的实体，容易在软件产品中重复使用，促进了**软件重用**。
- 面向对象方法特有的继承性，也进一步提高了面向对象软件的可重用性。

- 模拟人类思维
- 迭代开发
- 设计简单、容易理解
- 当需求变化时：
 - 要求服务员礼貌待客！





作业

- 习题1~3

