1、软件工程的产生起源于软件危机

(1)软件危机有哪些主要的表现？

* + 软件开发计划难以制定和实施
  + 软件开发费用和进度失控
  + 软件的质量无法让用户满意
  + 软件无法维护
  + 软件没有适当的文档资料
  + 软件开发成本逐年上升

(2)软件开发引入了工程的概念：时间限制、成本控制和质量因素

(3)软件工程至今仍未完全解决软件危机的所有问题

2、软件工程为软件开发制定了规范的软件开发过程

(1)软件生命周期模型应运而生

(2)瀑布模型作为第一个软件生命周期模型规定了6项软件开发的基本活动

(3)为了解决瀑布模型不灵活的缺陷，后期对其进行一系列的改动，进而产生很多变种

①为了解决需求不清楚的问题，诞生了演化模型；

②为了尽快解决核心功能的问题，诞生了基于优先级的增量模型；

③为了解决需求不清且用户又需要看到结果的情况，诞生了迭代（喷泉）模型；

④为了提高软件质量和尽早开展测试活动，诞生了V和W模型；

⑤为了解决大型项目的风险问题，诞生了螺旋模型；

⑥为了解决时间紧任务重且人力充沛时，诞生了RAD模型；

⑦为了尽可能提高代码复用问题，诞生了构件组装模型；

⑧为了在各个模型中解决需求不清、选择最优方案和解决技术难题的情况，诞生了原型方法；

⑨为了解决面向对象建模和应用的诞生了UP模型；⑩为了解决快速开发和快速响应客户多变需求的情况，诞生了诸如极限编程方法的敏捷思想

3、为了明确软件要做什么的目标，就必须要进行软件的需求分析

(1)软件需求的来源是使用软件的客户和用户；

(2)为此就需要了解客户和用户的业务背景：需求调研，客观描述业务场景或业务流程；

(3)进而对调研报告进行整理和抽象建模：物理模型和逻辑模型，确定软件系统要做什么的决策；

(4)为了解决自然语言表达二义性的问题，提出标准化的建模方法：

①结构化的建模方法：数据流图、数据词典、状态迁移图

②面向对象的建模方法：用例模型

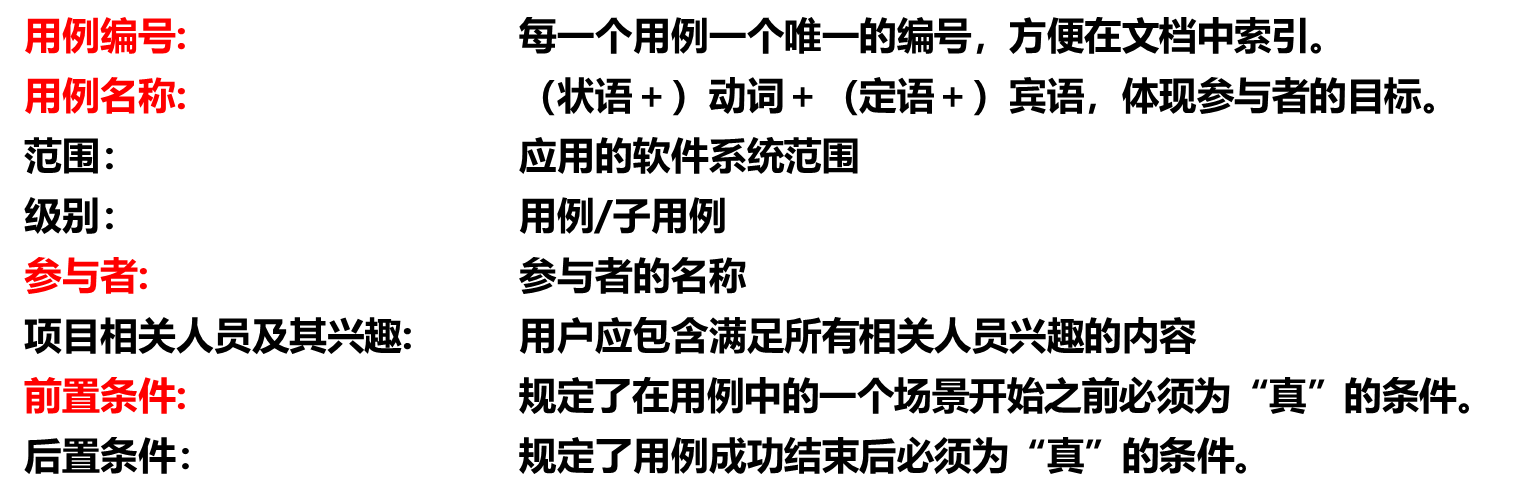
1)用例图

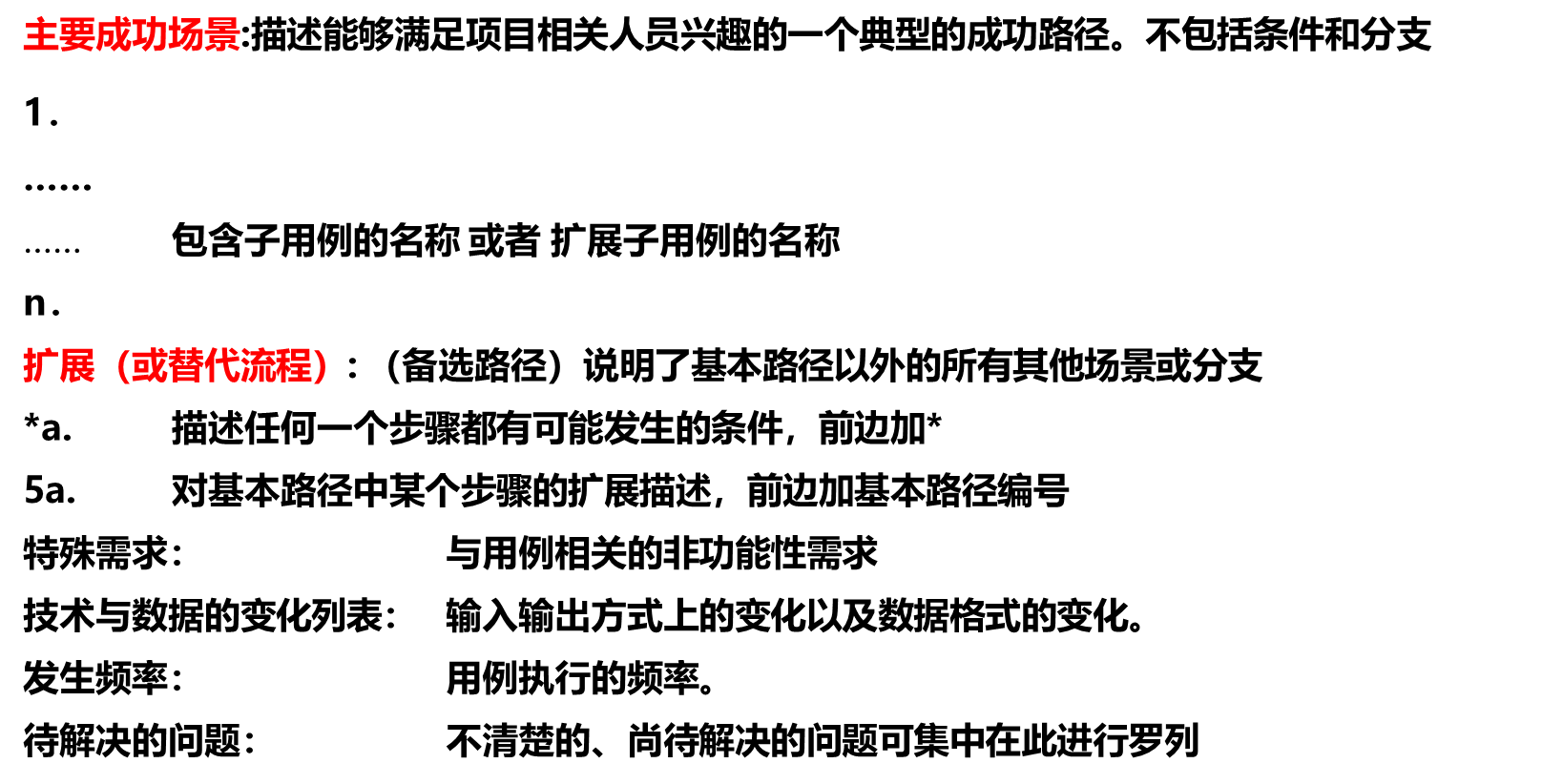
* 用例图由三个基本元素组成
  + Actor：称为角色或者参与者，表示使用系统的对象，代表角色的不一定是人，也可以是组织、系统或设备；
  + Use\_case：称为用例，描述角色如何使用系统功能实现需求目标的一组成功场景和一系列失败场景的集合；
  + Association：表示角色与用例之间的关系，以及用例和子用例之间的关系；



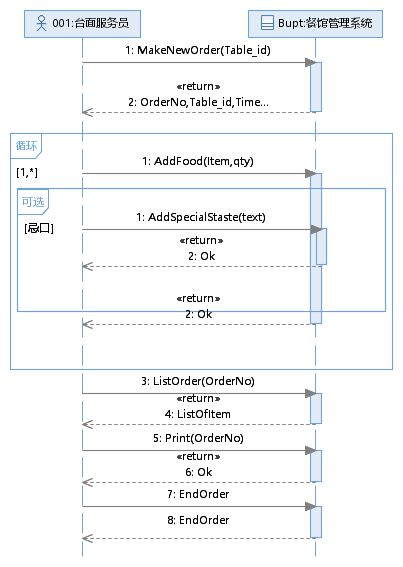
2)用例说明

* 基于已经找到的用例和子用例，并参考之前的需求定义以及场景描述的内容，将用例交互的成功场景和失败场景以标准的格式归纳描述。

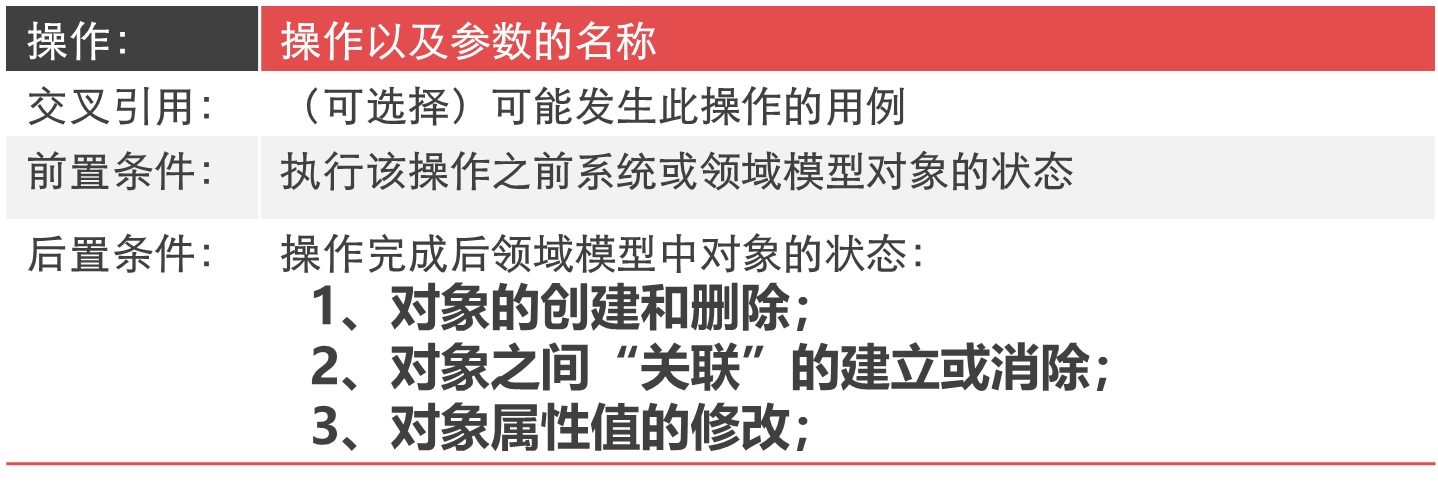




3)SSD



4)操作契约



(5)最终将建模的结果整合为需求分析阶段的里程碑成果：需求分析规格说明书，作为软件设计、软件实现（编码）和软件测试的依据；

4、为了说明软件系统的由哪些功能构成，就需要在编码之前进行软件系统的结构设计和数据设计：软件的概要设计

(1)软件概要设计的前提：需求规格说明书或部分的已知需求

(2)软件概要设计的主要活动：

①软件的框架结构的设计与选择；

②软件的结构设计

1)结构化建模方法：基于数据流图转换对应的功能结构图

2)面向对象建模方法：基于用例模型的SSD和操作契约，寻找和确定软件对象并为其分配功能；

③软件设计过程为了提高软件质量，就需要参考和应用软件设计原则

1)模块的独立性：模块的内聚性和模块之间的耦合性

2)面向对象的7个设计原则，也是尽可能降低类之间的耦合度

3)模块的控制范围和模块的作用范围也是降低耦合度必须要考虑的问题

④其中，软件系统的界面设计以及数据库表结构的设计也应包含在概要设计阶段；

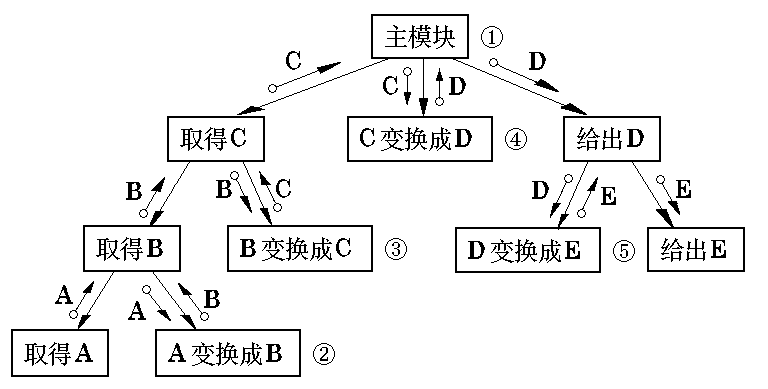
⑤如果存在一个软件类具有多个不同用例功能的情况，就需要应用状态迁移图表示其各个功能可被调用的上下文

⑥结构设计==功能建模；表结构设计或者数据结构设计==数据建模；状态迁移图设计或者并发设计==行为建模

(3)最终形成软件设计阶段的里程碑成果：软件概要设计说明书。

5、为了保证或者提高软件质量，还需要对功能模块或者软件类中的方法的内部逻辑结构进行的设计：软件的详细设计

(1)结构化建模：应用程序流程图或者NS图表示；



(2)面向对象建模：程序流程图、NS图或者UML 的活动图

(3)最终形成软件详细设计的里程碑成果：软件的详细设计说明书

6、编码或者软件实现：从软件工程的角度将源代码看成是工程文档

(1)序言性注释

* 位于程序代码之前，说明该模块（类及方法）具体作用
* 主要包括以下内容：
  + 程序标题：模块名称
  + 模块描述：该模块的功能和目的说明
  + 主要算法：（option）说明算法结构
  + 接口说明：说明该模块与其他模块的调用关系
  + 开发简历：
    - 创建者、创建时间；
    - 修改者、修改时间、修改内容；
    - 版本

(2)功能性注释

* + 主要描述一段程序，必要时对某一段进行说明；
  + 修改代码的同时，对应的功能性注释也要进行修改；

(3)要求源代码的可读性和可维护性尽可能满足后期开发人员的要求

7、为了在各个阶段保证软件质量，软件测试是软件开发过程必不可少的活动

(1)软件测试的目的：从使用者和开发者两个角度出发

(2)软件测试的对象：软件各阶段的里程碑成果

(3)软件测试活动：单元测试、集成、确认、系统和验收测试，它们的测试依据、测试环境、测试人员需要了解。

(4)软件测试活动必须具备的三个前提：软件配置、测试配置和测试工具

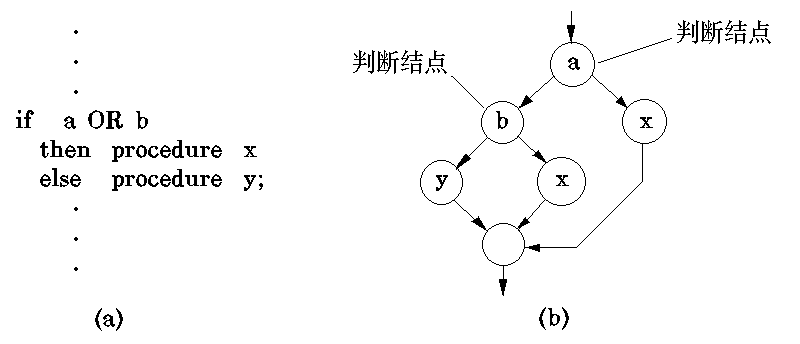
(5)软件测试的结果必须分析和保存，发现的问题必须处理和解决：不断的做测试

(6)软件测试的方法：

①白盒测试：6个逻辑覆盖的方法

1)路径覆盖为基础+5个逻辑覆盖

2)基于程序流程图给出程序的控制流图，获得独立路径数的上界：必须被测试的路径



3)转换控制流图之前必须将多条件的判定结构拆分为单条件的逻辑结构

②黑盒测试：等价类划分、边界值分析和因果图

1)等价类划分的5个划分原则很重要

* 按区间划分：如果某个字段的输入条件属于一个取值范围[x,y]，则可以确立
  + 一个有效等价类;
  + 两个无效等价类。
* 按数值集合划分：如果输入条件规定了输入数据的集合，则可划分
  + 一个有效等价类：所有符合输入条件的数据集合，
  + 一个无效等价类：所有不允许输入的数据集合。
* 如果输入条件是一个布尔量，则可以确定
  + 一个有效等价类，即取真；
  + 一个无效等价类，即取假。
* 按数值划分：如果规定了输入数据的一组值，而且程序要对每个输入值分别进行处理，这时可以为
  + 每一个输入值确立一个有效等价类;
  + 一个无效等价类，包含所有不允许输入的数值
* 按限制条件或规则划分：如果规定了输入数据必须遵守的规则或限制条件，则可以确立
  + 一个有效等价类，即各方面均符合规则要求;
  + 若干个无效等价类，每个无效等价类从不同角度违反输入规则。

2)等价类划分表的结构：等价类条件、有效等价类和无效等价类



8、一旦软件测试的结果被用户所接受，则软件系统将移交给用户，则软件就进入到软件的维护周期

(1)软件的维护种类：改正型、适应型、完善型和预防型

(2)软件维护周期的软件开发活动也非常重要，是延长软件生命周期的重要阶段

9、为了尽可能将软件开发各阶段的活动做到精准控制，保证工程的时间和成本以及软件质量还需要进行必须要软件项目管理。