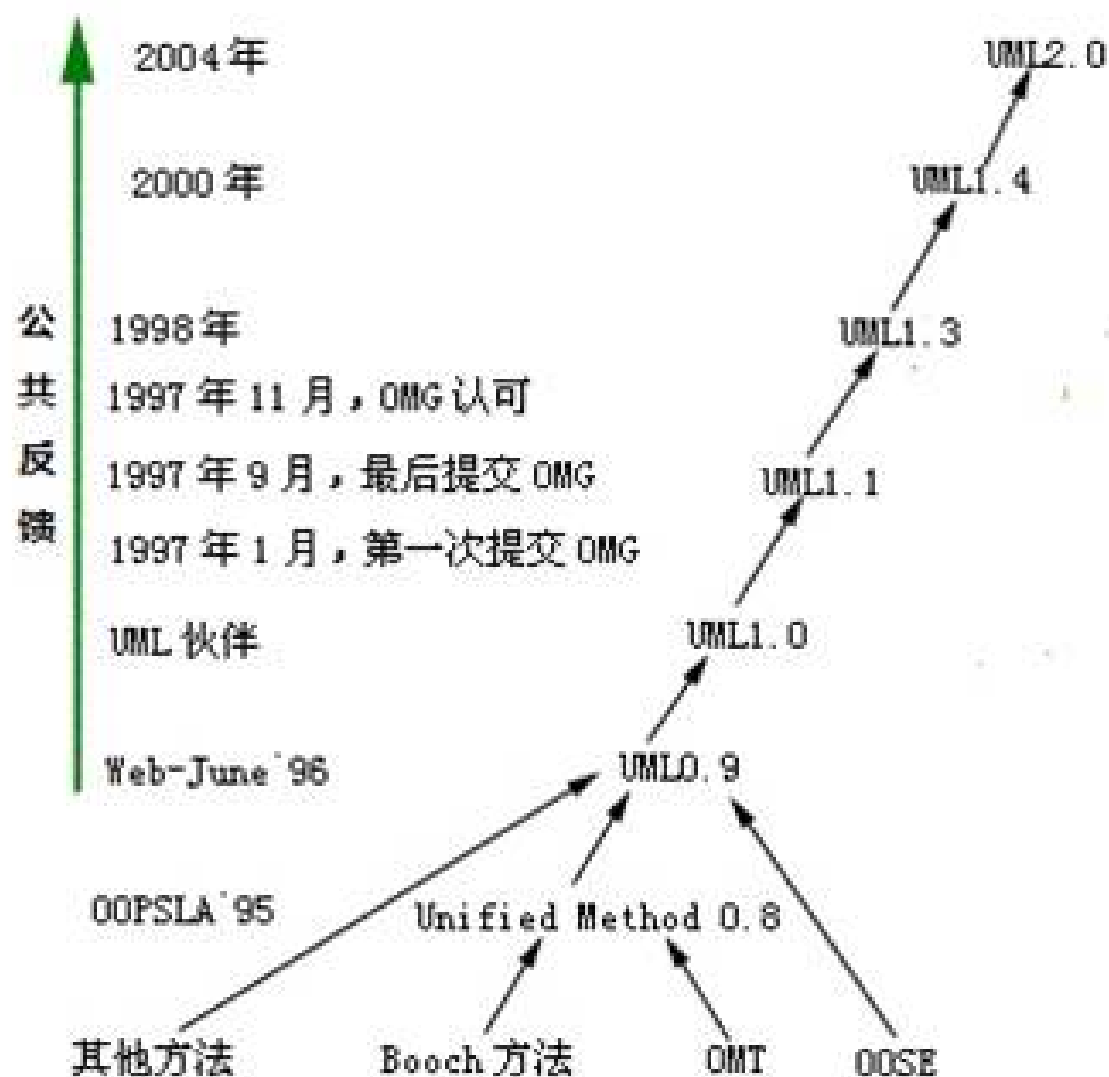


UML (Unified Modeling Language) :

用例图、类图

UML历史

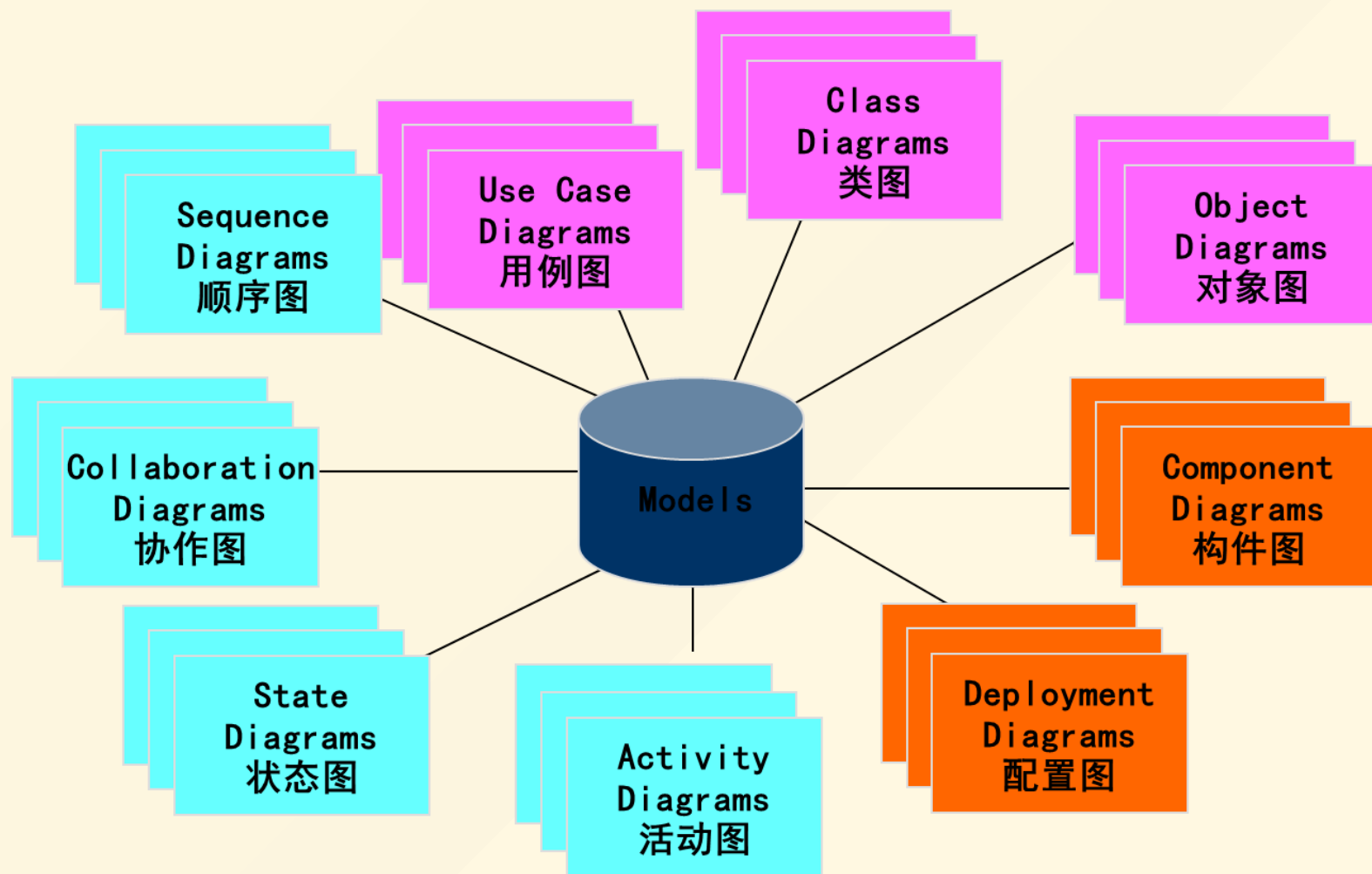
- Unification of "Three Amigos" work
 - Grady Booch: the Booch Method
 - James Rumbaugh: the Object Modeling Technique
 - Ivar Jacobson: Objectory
- 1994, UML 1.0
- 2005, UML 2.0



UML特点

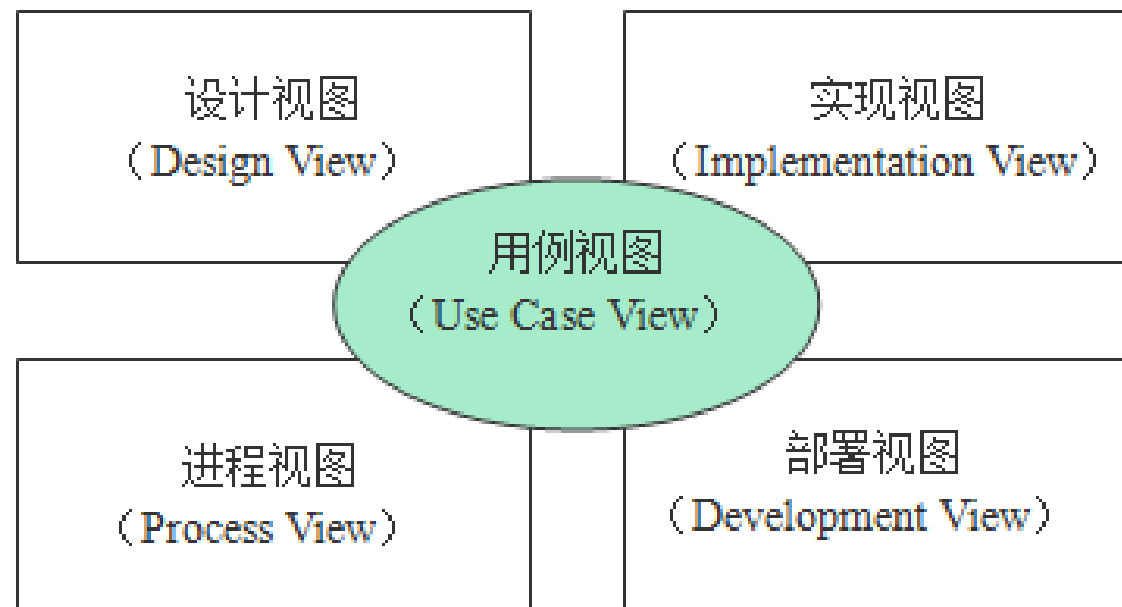
- UML是图示化说明如何构造一个软件系统的标准语言。
- UML独立于开发过程，可与大多数面向对象开发过程配合使用。
- UML独立于程序设计语言，可用C++、Java等任何一种面向对象程序设计语言实现。
- UML可应用于多种领域：软件开发、生产制造等。

UML的图



UML视图

- 不同的视图突出特定的参与群体所关心的系统的不同方面，通过合并所有五个视图中得到的信息就可以形成系统的完整描述。



UML视图

1. 用例视图

定义了系统的外部行为，为最终用户、分析人员和测试人员所关心。该视图定义了系统的需求，因此约束了描述系统设计和构造的某些方面的所有其他视图。

2. 设计视图

描述的是支持用例视图中规定的功能需求的逻辑结构。它由程序组件的定义，主要是类、类所包含的数据、类的行为以及类之间交互的说明组成。

UML视图

3. 实现视图

描述构造系统的物理组件，这些组件包括如可执行文件、代码库和数据库等内容。这个视图中包含的信息与配置管理和系统集成这类活动有关。

4. 进程视图

进程视图包括形成并发和同步机制的进程和线程。

5. 部署视图

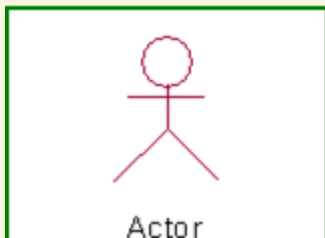
部署视图描述物理组件如何在系统运行的实际环境(如计算机网络)中分布。

UML用例图（Usecase Diagram）

- 用例图描述外部执行者（actor）与系统的交互，表达系统功能，即系统提供的服务。
- 主要元素：用例、执行者
- 用例：执行者与计算机一次典型交互，代表系统某一完整功能。



- 执行者：描述与系统交互的人或物，代表外部实体（如用户、硬件设备或其它软件系统）。



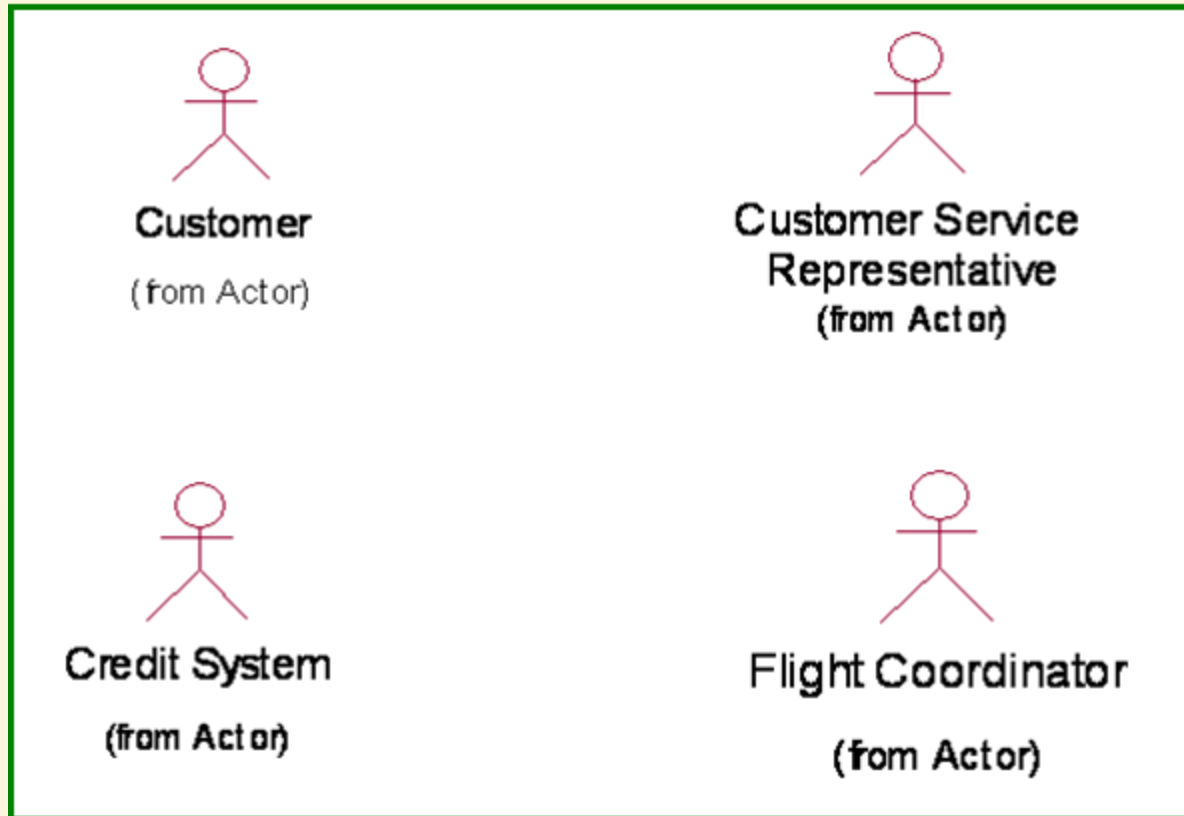
UML用例图举例

例：建立一航空公司的机票预订系统，让客户通过电话或网络买票、改变订票、取消订票、预定旅馆、租车等等。

用例建模步骤一：发现执行者

- 谁使用该系统；
- 谁改变系统的数据；
- 谁从系统获取信息；
- 谁需要系统的支持以完成日常工作任务；
- 谁负责维护、管理并保持系统正常运行；
- 系统需要应付那些硬件设备；
- 系统需要和那些外部系统交互；
- 谁对系统运行产生的结果感兴趣。

机票预订系统中的执行者



用例建模步骤二：获取用例

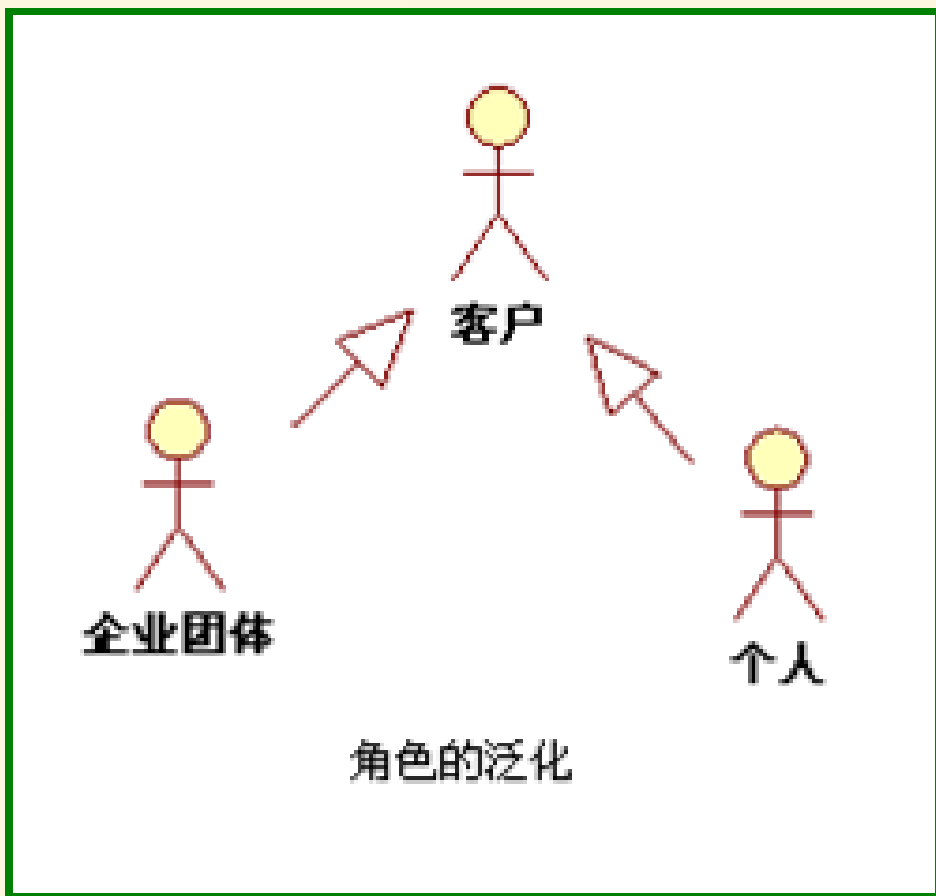
- 向执行者提出问题获取用例：
 - 执行者需获取何种功能，需要作什么；
 - 执行者需读取、产生、删除、修改或存储系统中某种信息；
 - 系统发生事件和执行者间是否需要通信。

机票预订系统中的用例



用例建模步骤三：明确执行者间关联

泛化关系

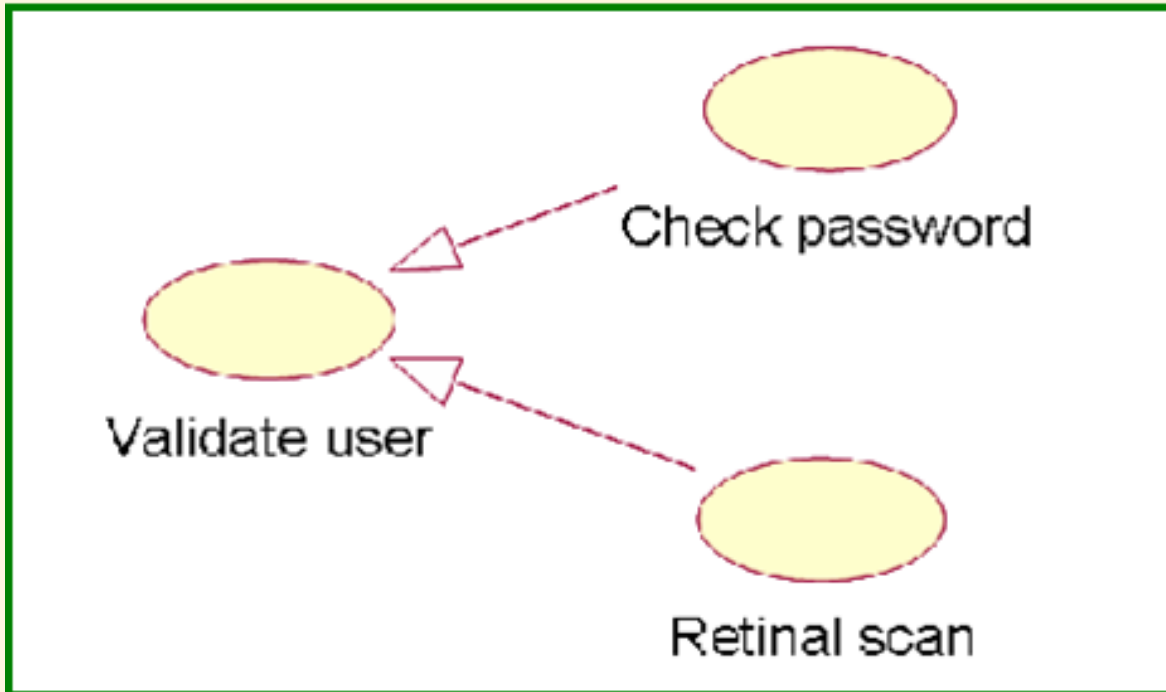


用例建模步骤四：明确用例间关联

- 泛化关系（一般与特殊关系）
- 扩展关系：允许一个用例扩展另一用例提供的功能。
- 包含关系：一个基本用例行为包含另一个用例行为。

用例间的泛化关系

子用例有父用例的行为，可出现在父用例出现的任何地方。
子用例可添加自己行为（前者检查文本密码，后者检查用户视网膜）。

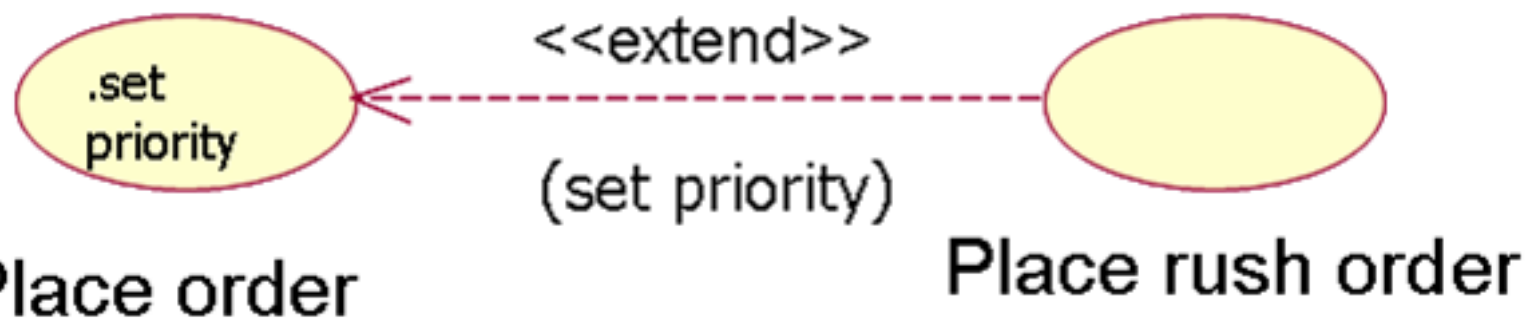


用例间的扩展关系

允许一个用例扩展另一用例提供的功能，与泛化关联类似，但有更多规则限制：

基本用例必须声明若干“扩展点”，扩展用例只能在扩展点上增加新行为。

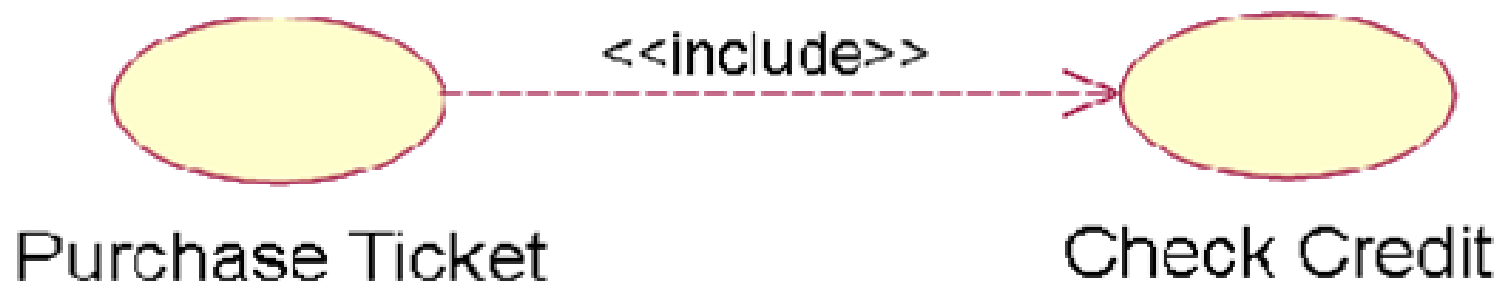
例：



用例间的包含关系

一个基本用例行为包含另一个用例行为。

例：



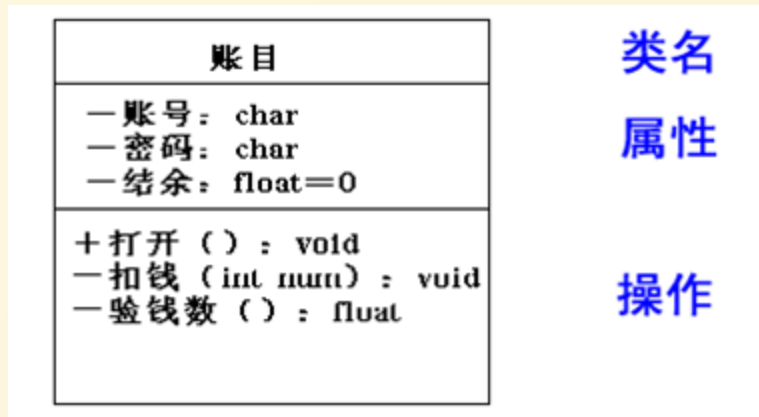
Check Credit检查输入的信用卡号是否有效，有足够资金。
处理Purchase Ticket用例的时候，总运行Check Credit用例。

用例的泛化、扩展、包含

- A是一个完整的用例，在业务流程中可能会发生一种异常，该异常在B用例中处理。（扩展）
- A是一个一般性的用例（完整或不完整都有可能），B用例是其特殊的一种情况。（泛化）
- A是一个不完整的用例，其业务流程中有一部分在B中处理，通常其它用例也会用到B的处理。（包含）

UML类图（ Class Diagram ）

类图是面向对象建模最常用的图，描述类与类间的静态关系。



属性的语法：[可见性] 属性名[：类型][=初值]

操作的语法：[可见性]操作名[（参数列表）][：返回类型]

可见性：公有（+）、私有（-）、保护（#）

类之间的关联关系（ Association ）

- 普通关联：双向，用实线连接两个类。



- 导航关联：关联是单向的，用实线箭头连接两个类。



用重数（ multiplicity ）表示关联中的数量关系。

类之间的聚合关系（ Aggregation ）

类与类间关系是"has-a"，整体与部分关系，较弱情况。

菱形端代表整体事物类；代表部分事物类可属于整体事物类。

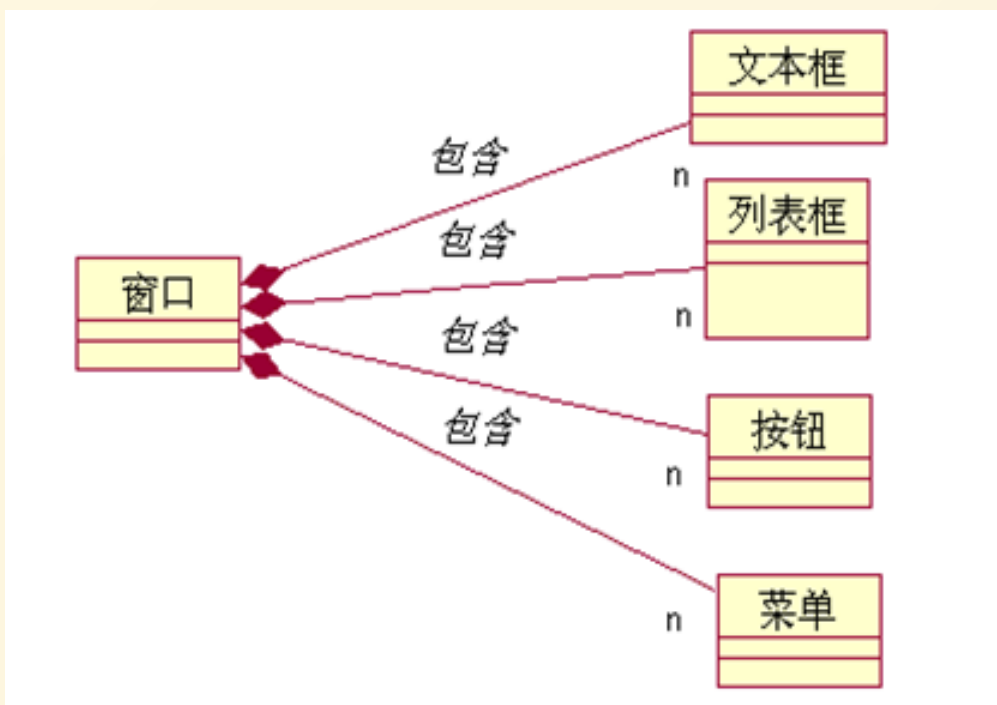
聚合关系中代表部分事物对象与代表聚合事物对象生存期无关，删除聚合对象不一定删除代表部分事物对象。



类之间的组合关系（Composition）

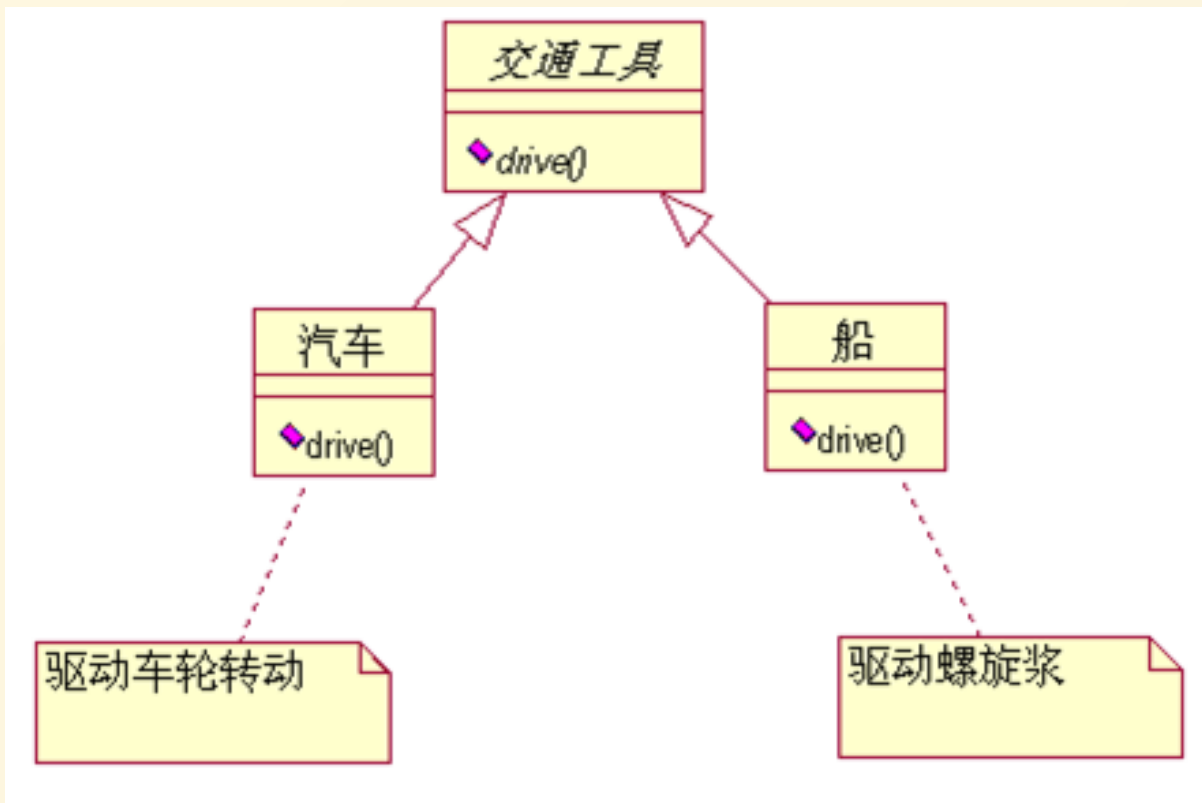
组合是“contains-a”关系，是整体与部分较强关系，部分类完全隶属于整体类。

组合中删除组合对象，同时也就删除代表部分事物对象。



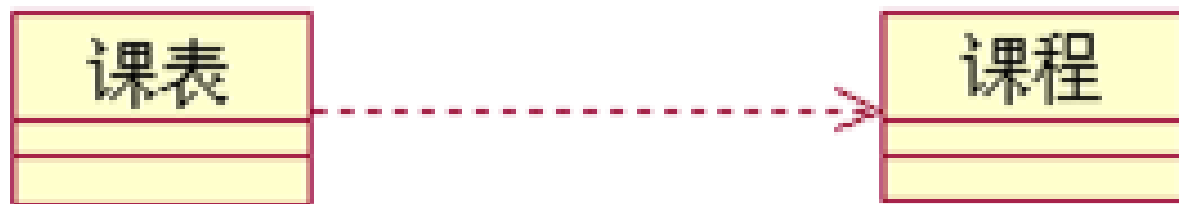
类之间的泛化关系（Generalization）

指类间的“一般-特殊”关系（即继承关系）。



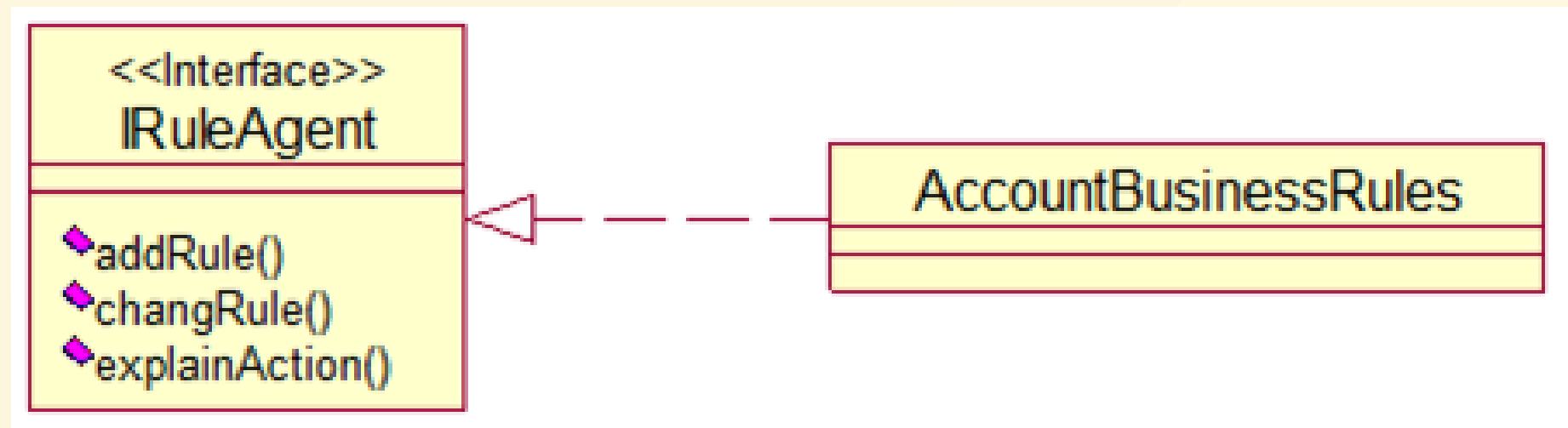
类（或接口）之间的依赖关系（Dependency）

一模型元素变化必影响另一模型元素。



类（或接口）之间的实现关系（Realization）

是指一个类描述了另一个类（通常是抽象类或接口）保证实现的合约。



UML类图

举例

