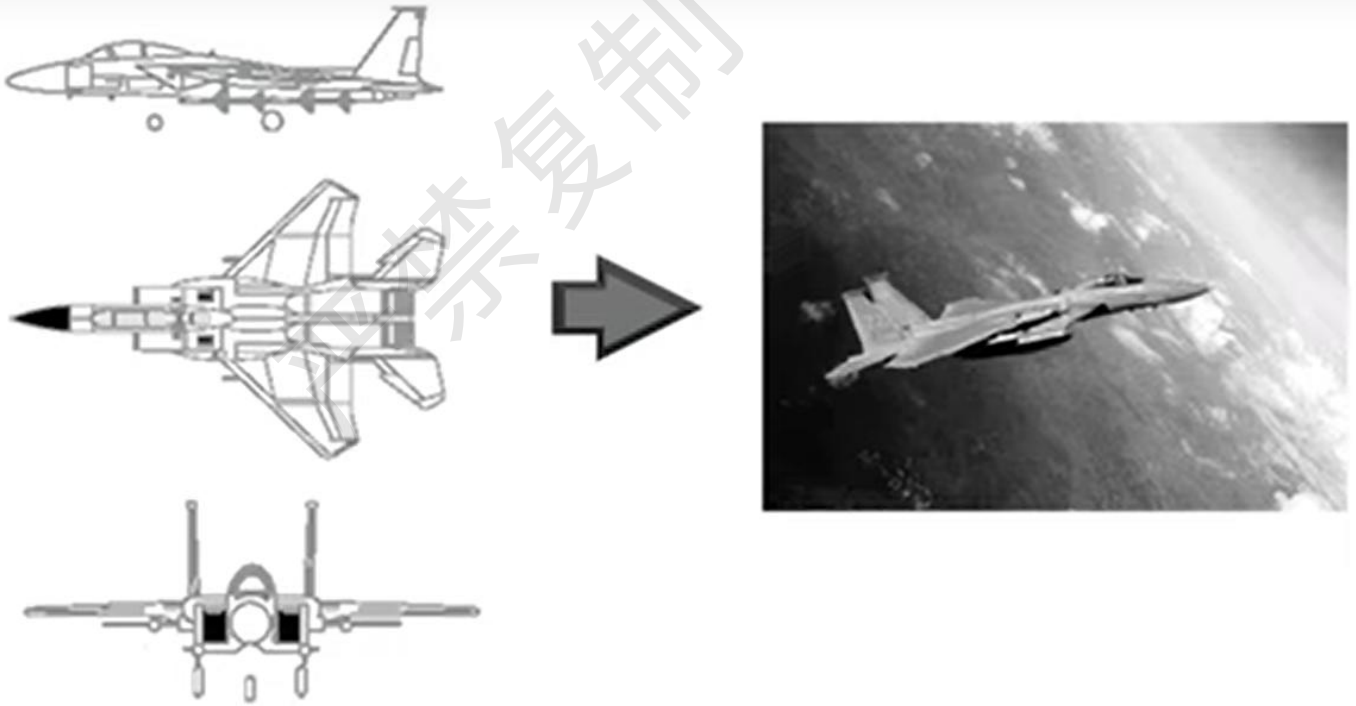
软件体系结构原理、方法与实践

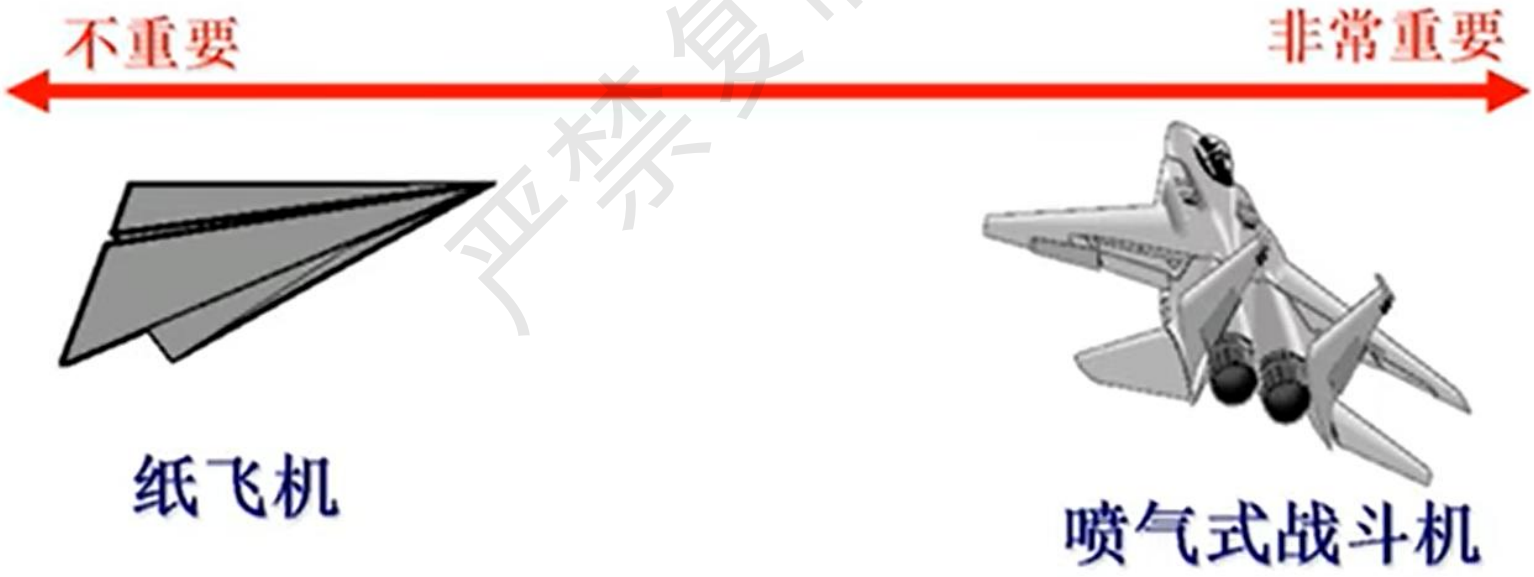
统一建模语言

什么是模型？

- 模型是现实世界的简化



模型的重要性



建模的目的

- 建模的根本目的是为了更好地理解待开发的系统
  - 模型有助于按照所需的样式可视化(Visualize)目标系统
  - 模型能够描述(Specify)系统的结构和行为
  - 模型提供构造(Construct)系统的模板
  - 模型可以文档化(Document)设计决策

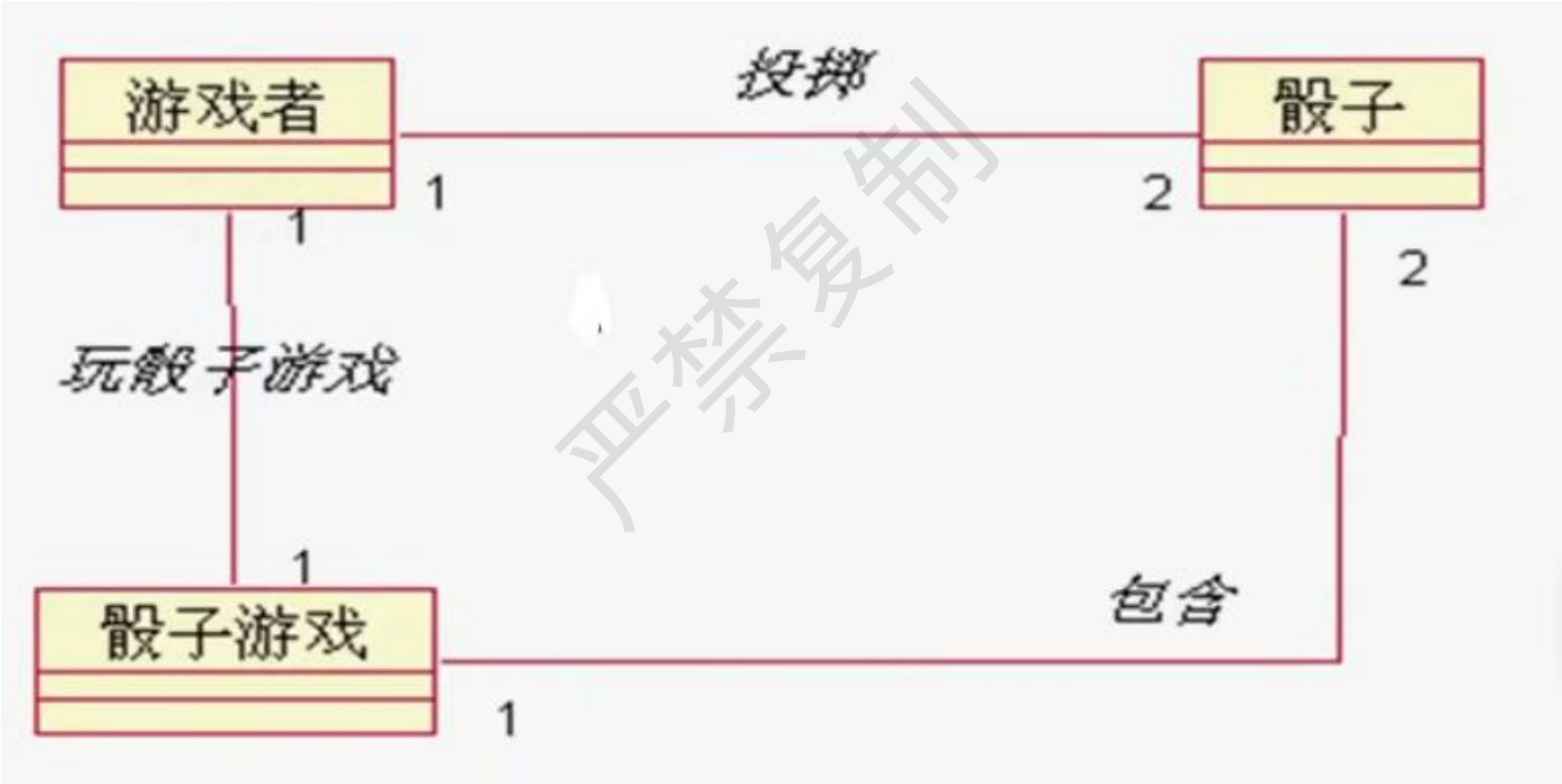
建模的基本原则

- 建模过程需要遵循的原则
  - 选择合适的模型：所创建的模型对解决方案的形成具有重要的影响
  - 模型具有不同的精确程度：面向不同的用户提供不同抽象层次的模型
  - 好的模型是与现实相联系的：简化不能掩盖掉任何重要的细节
  - 单一的模型是不够的：需要从多个视角创建不同的模型

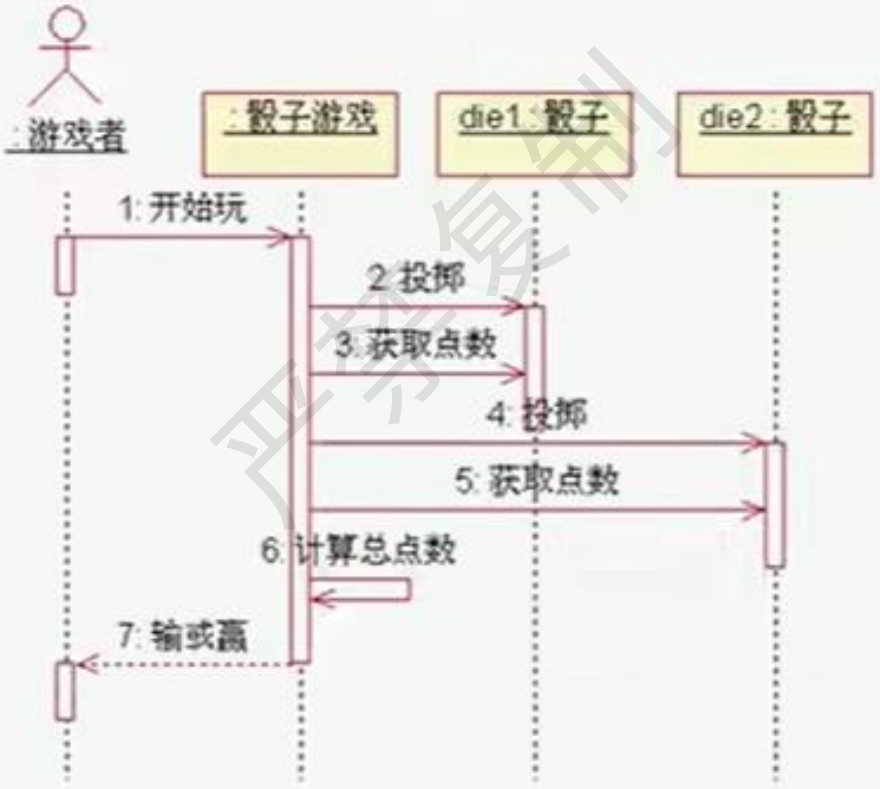
快速开始的实例

- 骰子游戏：软件模拟游戏者投掷两个骰子，如果总点数是7则赢得游戏，否则为输
- 过程：定义用例->定义领域模型->定义交互图->定义设计类图
- 定义用例(用例是需求分析的一种工具，它是一些情节的描述)
  - 骰子游戏：
    - 1、游戏者请求骰子
    - 2、系统展示结果：如果骰子的总点数是7，则游戏者赢；否则游戏者输
- 定义领域模型（OOA） - 识别问题中的**概念**，它是对真实世界领域中的概念和想像可视化，与具体实现的软件技术无关（比如java或C#）
  - 游戏者
  - 骰子
  - 骰子游戏

骰子游戏的领域模型



分配对象职责并绘制交互图(动态建模)



定义设计类图(静态建模)

- 从领域模型以及交互图中获得启示，定义软件类，包括属性、方法等等
- 骰子游戏的局部设计类图示例如下：

UML概述*

标准定义:统一建模语言UML(Unified Modeling Language),描述，构造和文档化系统制品的可视化语言。

- UML是一种建模语言
- UML是一种可视化语言
- UML是一种可用于详细描述的语言
- UML是一种构造语言
- UML是一种文档化语言

统一建模语言（UML）

- UML — You Must Learn
- UML — Unified Modeling Language
- UML是一种标准的图形化建模语言，是面向对象分析与设计的标准表示，它：
  - 不是一种程序设计语言，而是一种可视化的建模语言(用于分析设计)
  - 不是工具或知识库的规格说明，而是一种建模语言规格说明，是一种模型表示的标准
  - 不是过程，也不是方法，但允许任何一种过程和方法使用它

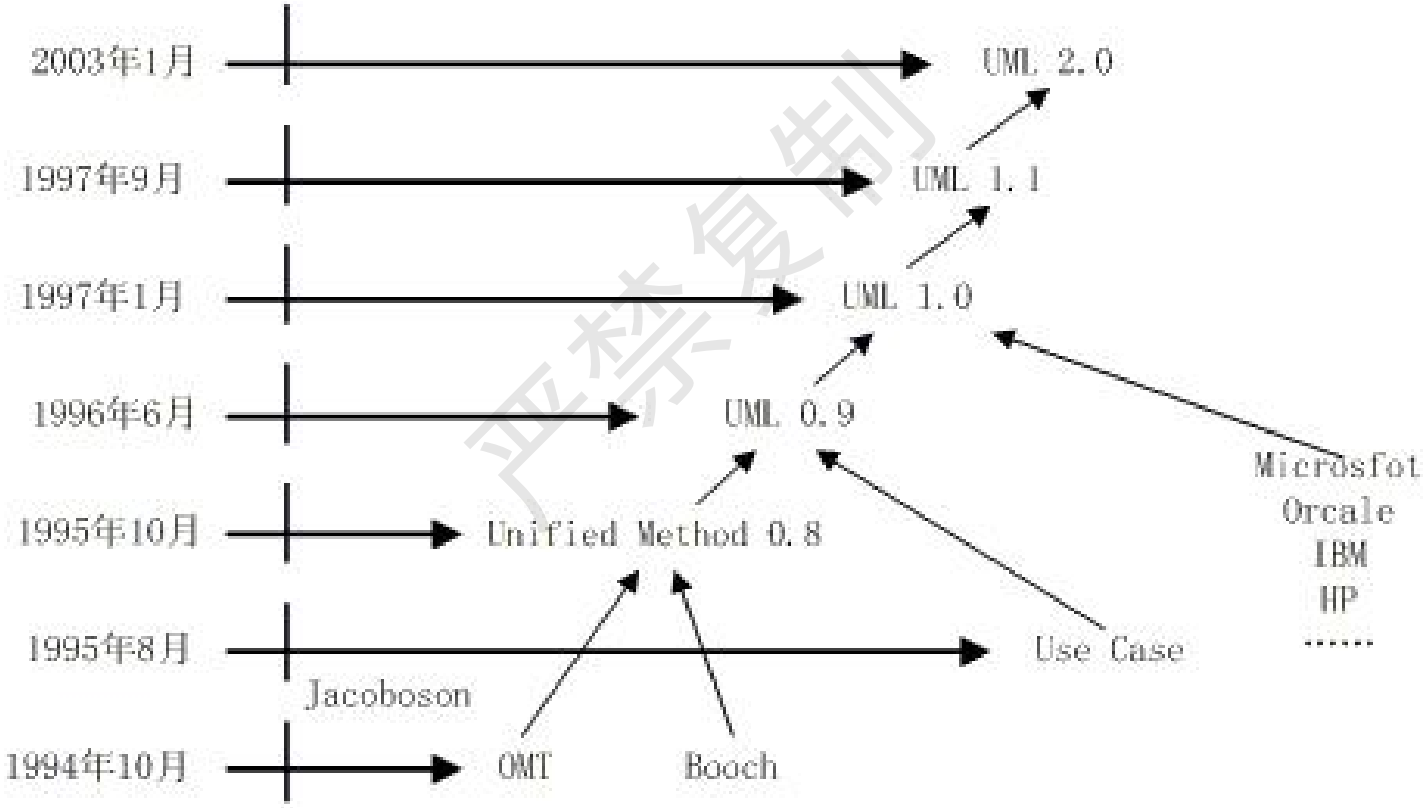
选择UML

- 很多情况下，推荐使用UML：
  - 1）OO方法是项目决定采用的方法论，是整个项目或产品成功的关键
  - 2）开发人员感觉用源码说明不了真正的问题，希望提高交流效率
  - 3）系统的规模和设计都比较复杂，需要用图形抽象地表达复杂概念，降低开发风险
  - 4）组织希望记录已成功项目、产品的设计方案，在开发新项目时可以参考、复用过去的设计
  - 5）有必要采用一套通用的图形语言和符号体系描述组织的业务流程和软件需求，促进业务人员、开发人员之间一致、高效的交流

不选择UML

- UML不是万能，有些场合并不适合
  - 1）传统的做法已完全适用，对面向对象技术的要求也不高，项目非常成功，无任何改进的必要
  - 2）开发的系统比较简单，直接用源码配上少量的文字就能解决问题，软件开发文档也无需添加图形来辅助说明
  - 3）开发的系统本身不属于OO方法、UML适用的范围

UML的发展历史



新的开始，新的起点，让我们一起为梦想而努力。

UML的应用领域

- UML是一种建模语言而不是一种方法，其中并不包括过程的概念，其本身是独立于过程的，可以在**任何过程中使用它**。
- UML能够用面向对象的方法描述任何类型的系统，并对系统开发从需求调研到测试和维护的**各个阶段进行有效的支持**

UML的结构

- 规格说明：对构造块的语法，语义进行文字性说明，是模型的核心。
- 结构事物：在模型中属于静态部分，代表该年少或物理上的元素；总共有七种结构事物
- 行为事物：是UML模型中的动态部分，代表时间和空间上的动作，主要有两种行为事物，分别是交互(消息)和状态机。

1、交互(内部活动)：由一组对象之间在特定上下文中，为达到特定目的而进行的一系列消息交互动作。

2、状态机：由一系列对象状态组成，状态的变化，如审核流程。

- 分组事物：是UML中的组织部分，可以把它看成一个盒子，模型可在其中进行分解，如包，子系统。
- 注释事物：模型的解释，说明，描述的元素，如注释。

- 公共分类
- 扩展机制

UML2.0新增

- 构件图
- 部署图
- 包图
- 组合结构图
- 制品图

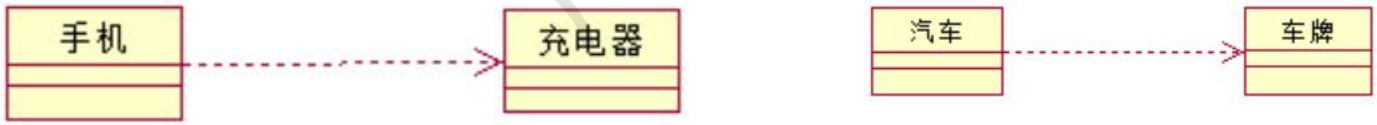
- 顺序图/序列图
- 通信图/协作图
- 状态图
- 活动图
- 定时图
- 交互概览图

UML2.0新增

种图，
1.0的
基础上新增了
5种图

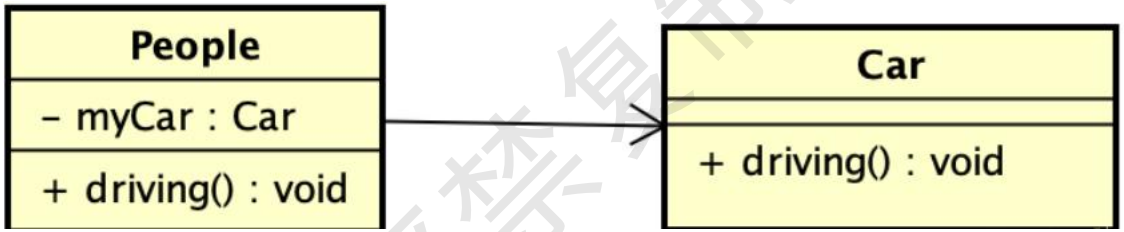
UML的结构-构造块中的关系-依赖关系

- **依赖关系:**也是类与类之间的连接. 表示一个类依赖于另一个类的定义. 依赖关系总是单向的 。可以简单的理解，就是一个类A使用到了另一个类B，而这种使用关系是具有偶然性的、临时性的、非常弱的，但是B类的变化会影响到A，如:参数或变量；也就是说一个事务变化影响另外一个事务,
- **箭线表示：**带箭头的虚线，指向被使用者。
- **例如：**手机依赖于充电器，汽车依赖于车牌。

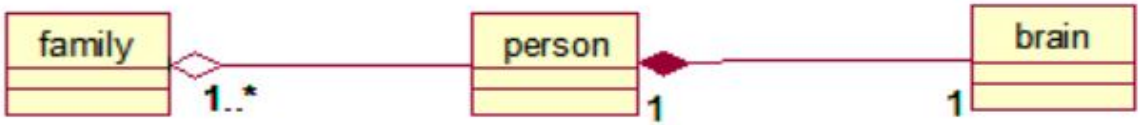


UML的结构-构造块中的关系-关联关系

- **关联关系:**关联关系式一种结构化的关系，是指一种对象和另一种对象有联系。给定关联的两个类。能够从当中的一个类的对象访问到另外一个类的相关对象
  箭线表示：带普通箭头的实线，指向被拥有者。双向的关联可以有两个箭头，或者没有箭头，单向的关联有一个箭头。如图：人是人，车是车，没有整体与部分的关系。

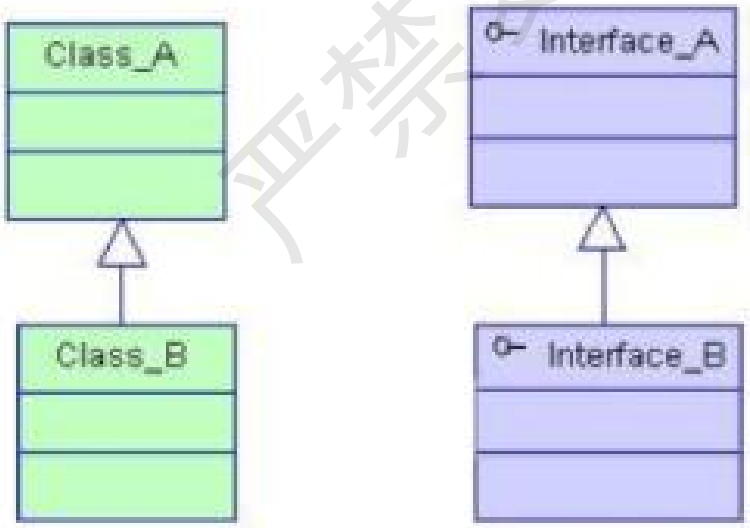


- **聚合关系:**总体和部分的关系，Has-a关系。总体和部分能够分离，总体与部分生命周期不同；如：计算机和主板，家庭和孩子
- **组合关系：**是一种包含关系，Is-a 关系。总体与部分不可分割，总体与部分的生命周期相同；如：桌子与桌腿的关系，人和大脑



UML的结构-构造块中的关系-泛化关系

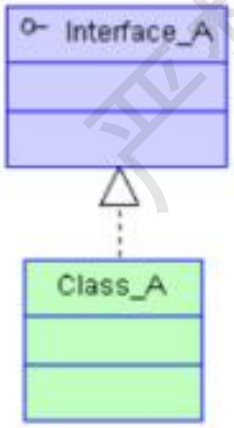
- **泛化关系:**一个类/接口继承另外一个类/接口的关系，子类还能添加自己的功能；子类与父类的关系，也称为特殊和一般的关系，例如：动物类与老虎类
- **箭线表示:**用实线空心箭头表示
- **如图：**B类继承A类，B接口继承A接口均为泛化关系



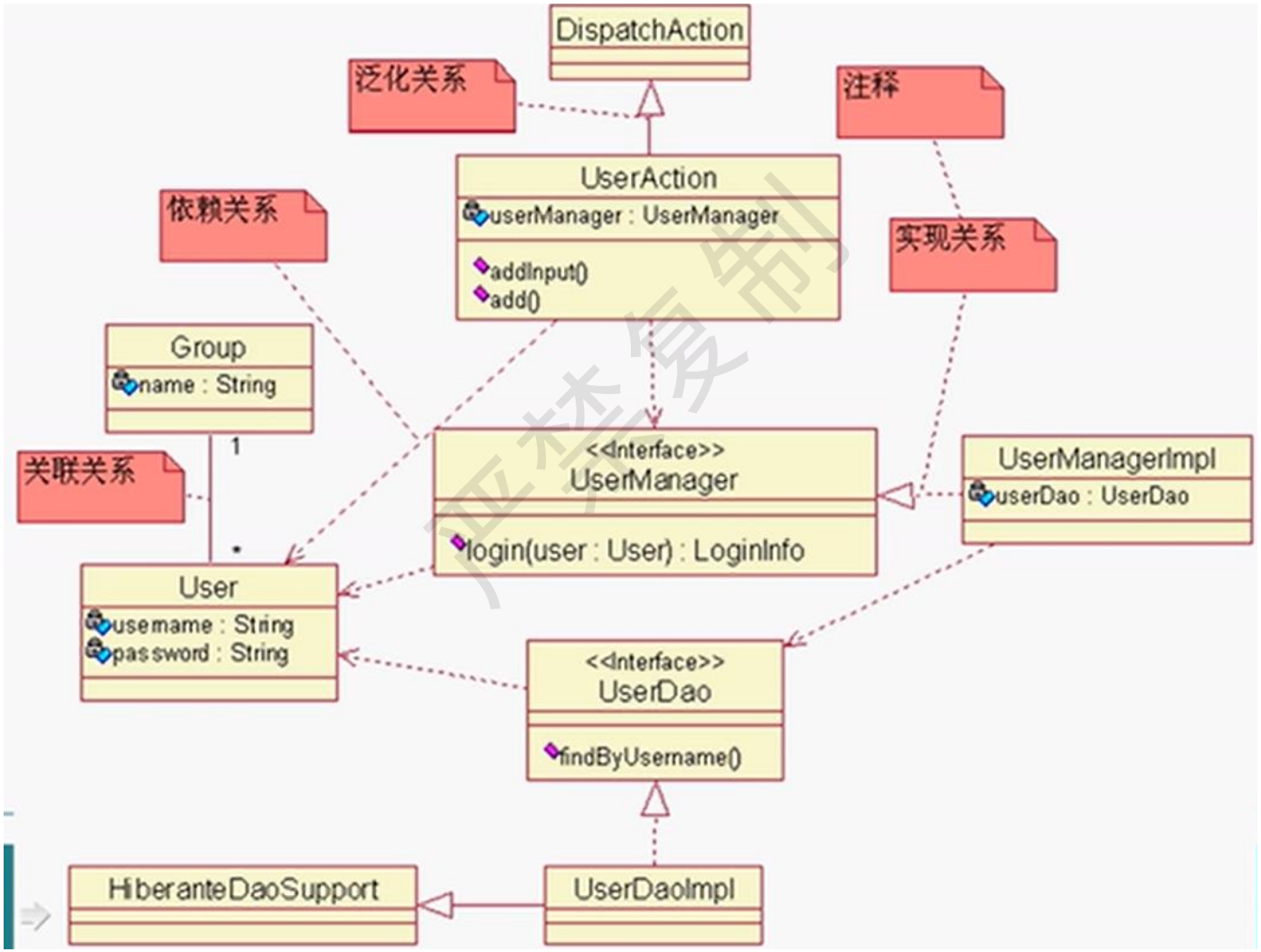
UML的结构-构造块中的关系-实现关系

- **实现关系:**指一个类实现接口的关系。动物类实现移动的接口
- **箭线表示:**虚线空心箭头表示

如图：A类实现A接口均为泛化关系



UML的结构-关系图形表示实例



UML的结构-五个系统视图

逻辑视图:用于描述用例视图中提出的系统功能实现，关注系统内部，既描述系统的静态结构(类，对象以及他们之间的关系)，也描述系统内部的动态协作关系。
组成:静态(类图和对象图)，动态(状态图，序列图，协作图，活动图)。
使用者：系统分析人员，设计人员
干系人关注点:类和对象

实现视图：组件是不同类型的代码模块，构造应用的软件单元，该视图描述系统的实现模块和他们之间的依赖关系。
组成：构件图

使用者：开发人员
干系人关注点:物理代码文件和组件

用例视图:描述系统应具备的功能，用例是系统的一个功能单元，可以被描述为系统与参与者的一次交互，是情景描述。
组成:用例图，使用者：用户
关系人关注:需求分析模型

进程视图：考虑资源的有效利用，代码的并行执行以及系统环境中异步事件的处理

组成：状态图，协作图，活动图

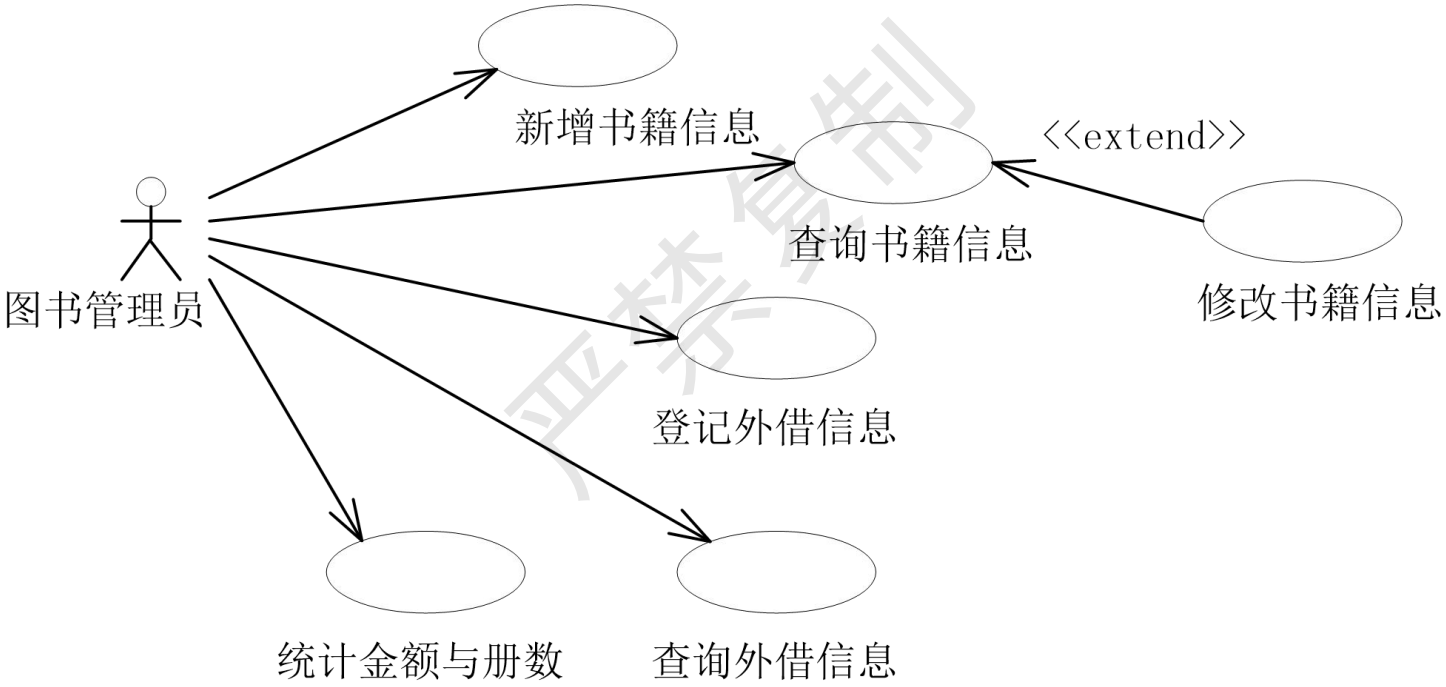
使用者：开发人员，系统集成人员
干系人关注点:线程，进程，并发

物理视图:显示系统的物理部署，描述节点上运行实例的部署情况。

组成：部署图

使用者：开发人员，系统集成人员，测试
干系人关注点:软件到硬件的映射

用例图



类图

类图:一切面向对象的核心建模工具,描述了系统中对象的类型以及他们之间存在的各种静态关系。

类图中的各种关系:

泛化关系:继承关系。

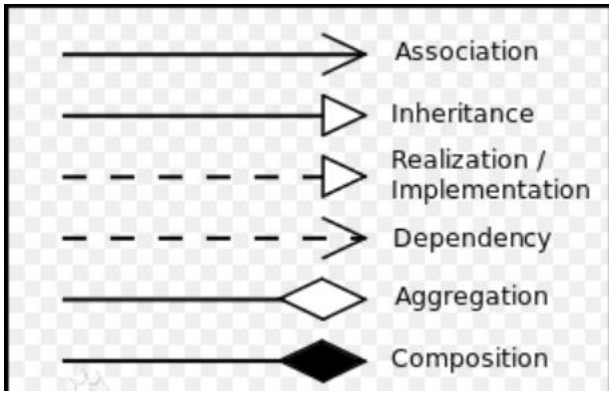
实现关系:接口与实现类关系

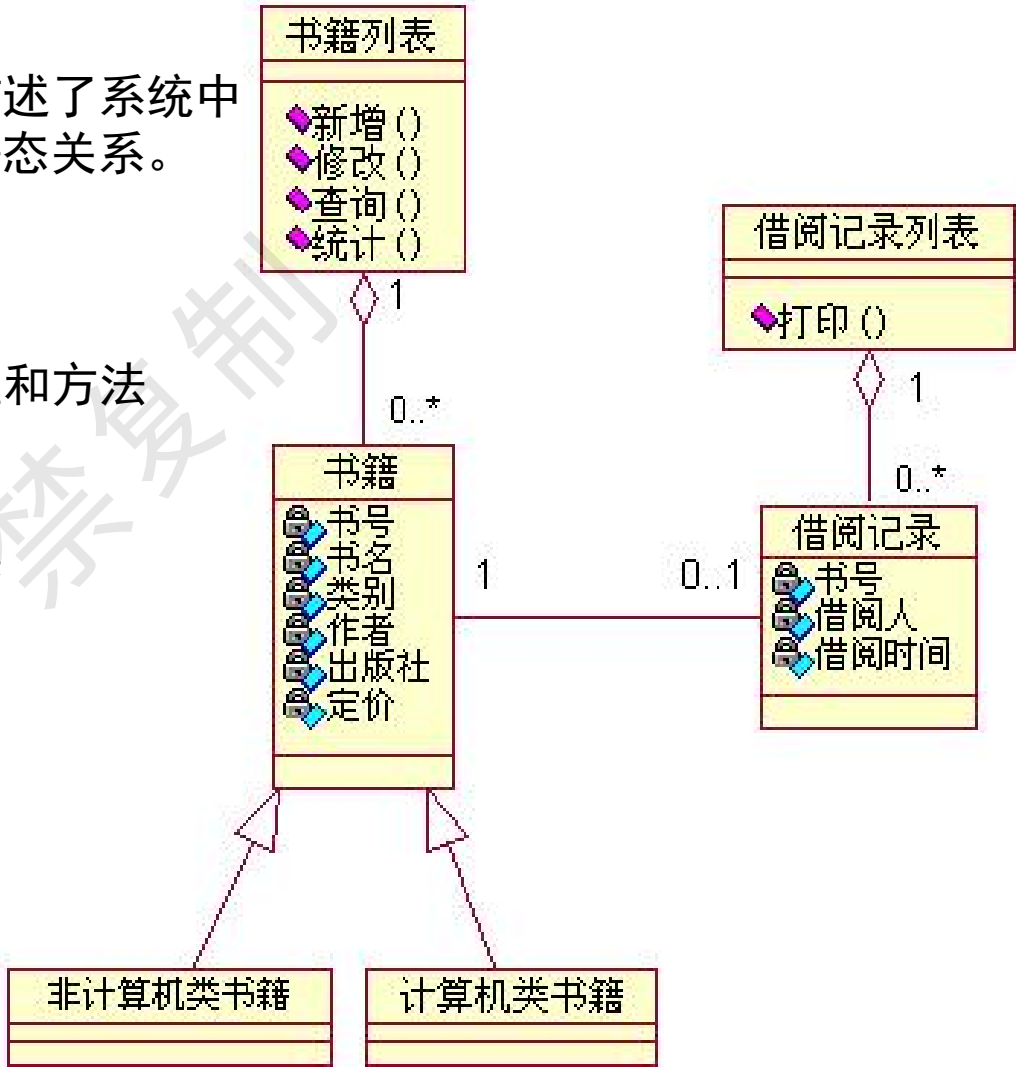
关联关系:一个类知道另一个类的属性和方法

聚合关系:整体与部分生命周期不同

组合关系:整体与部分生命周期相同

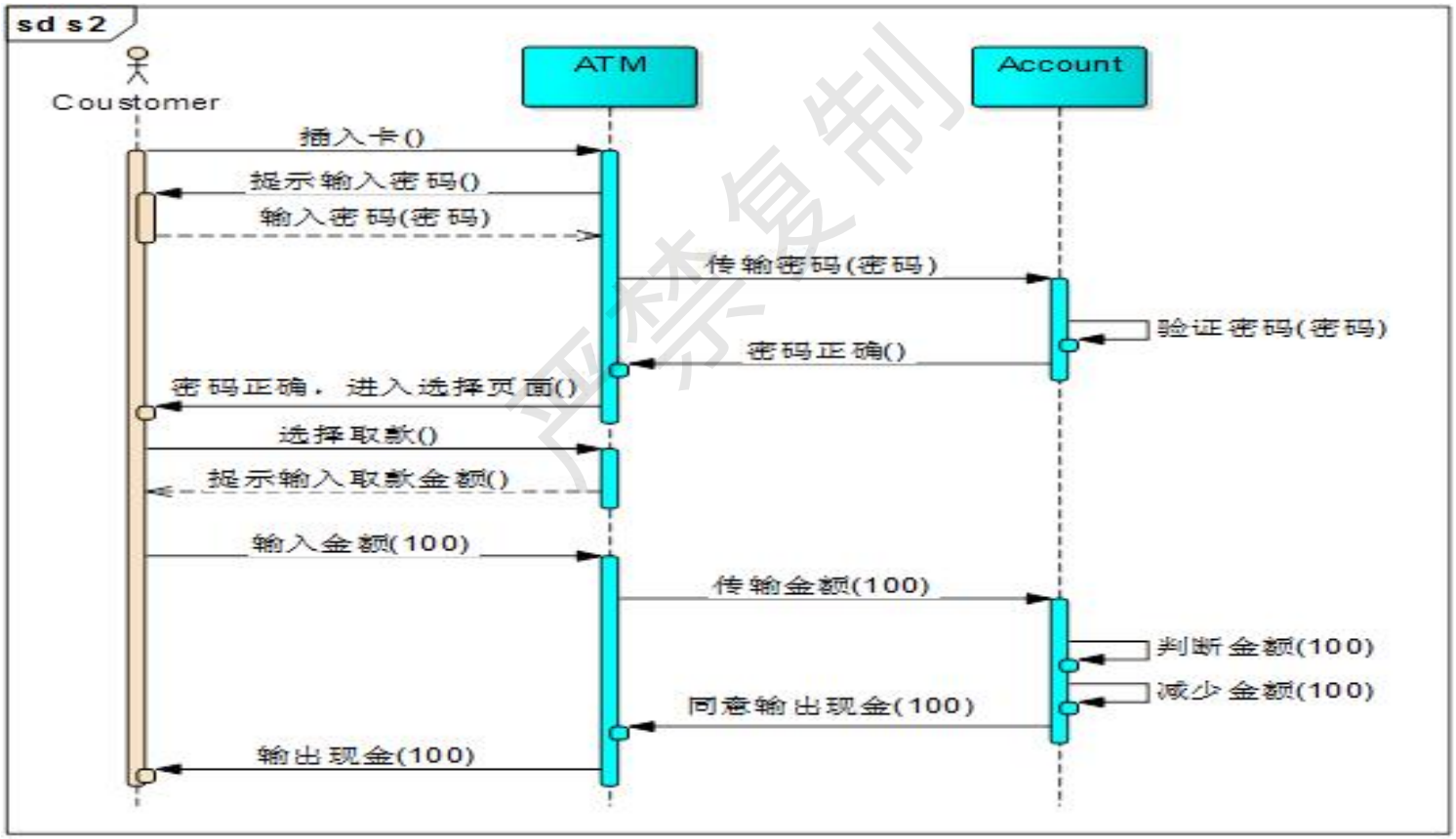
依赖关系:一个类变化影响另外一个类





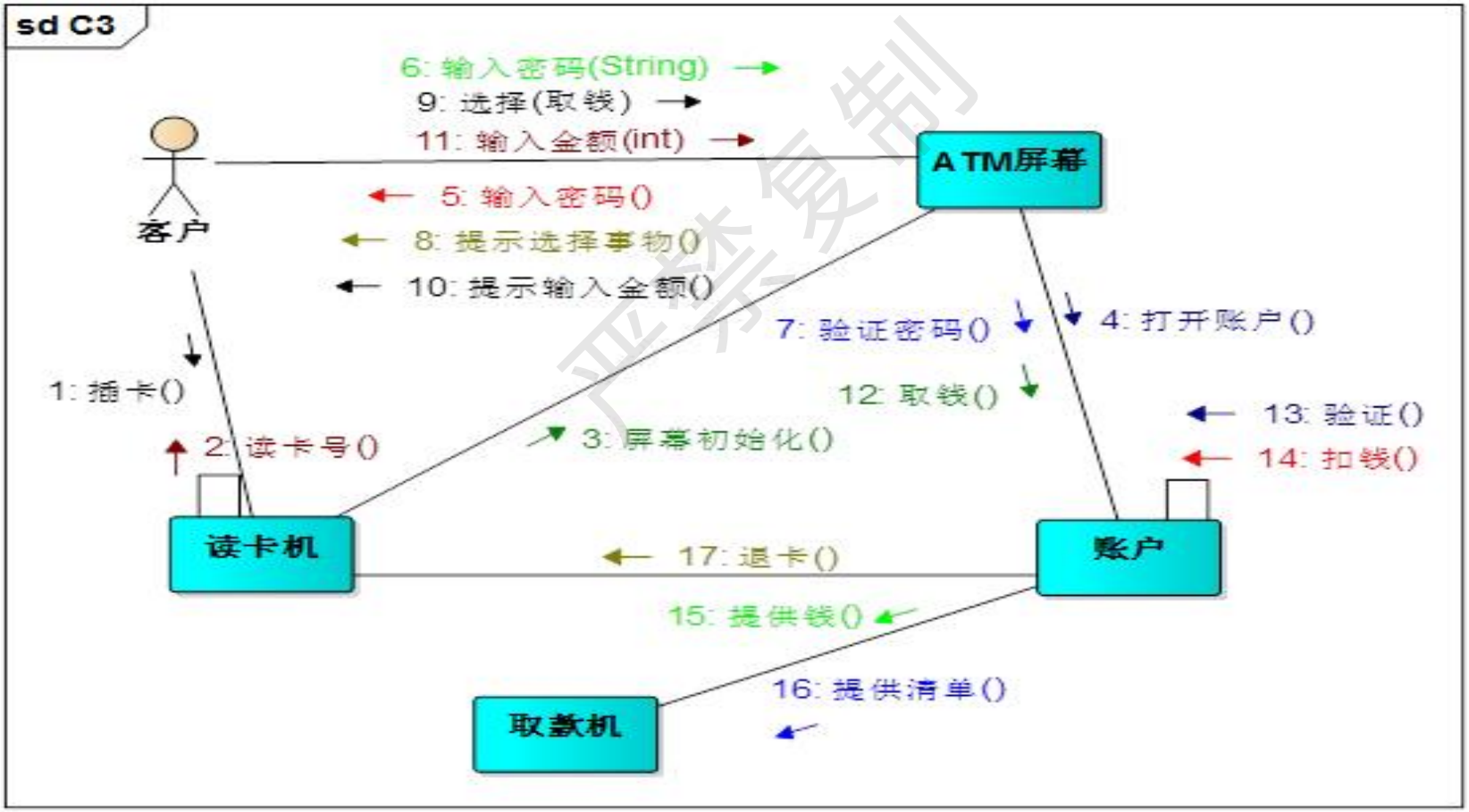
交互图-顺序图

顺序图:描述按时间的先后顺序对象之间的交互过程，纵向是时间轴，时间沿竖线向下延伸。顺序图对象，消息，生命线(激活)构成。



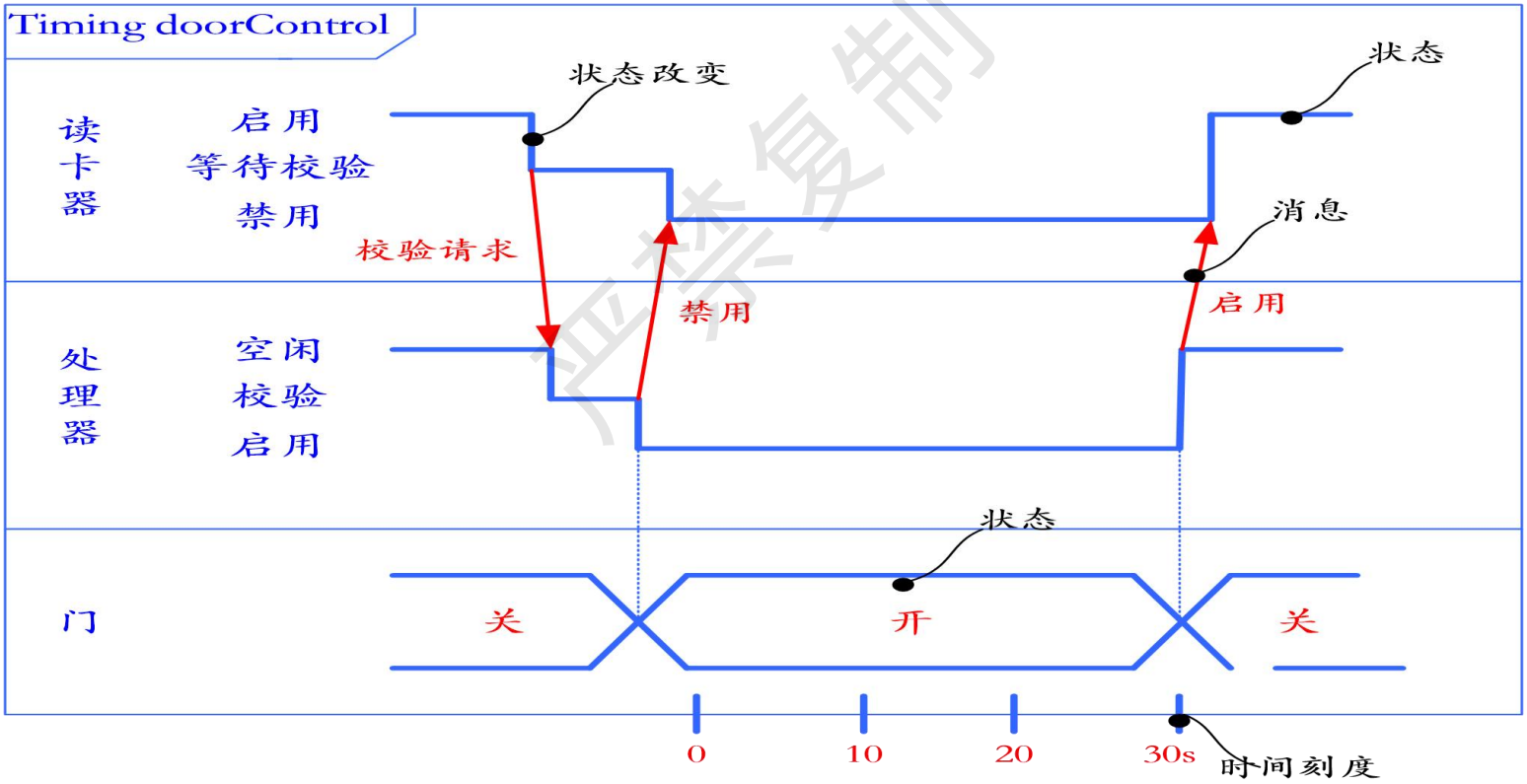
交互图-通信图

又称协作图，是表示对象交互关系的图，表示多个对象在达到共同目标的过程中互相通信的情况。强调收发消息的对象或参与者的结构组织关系。



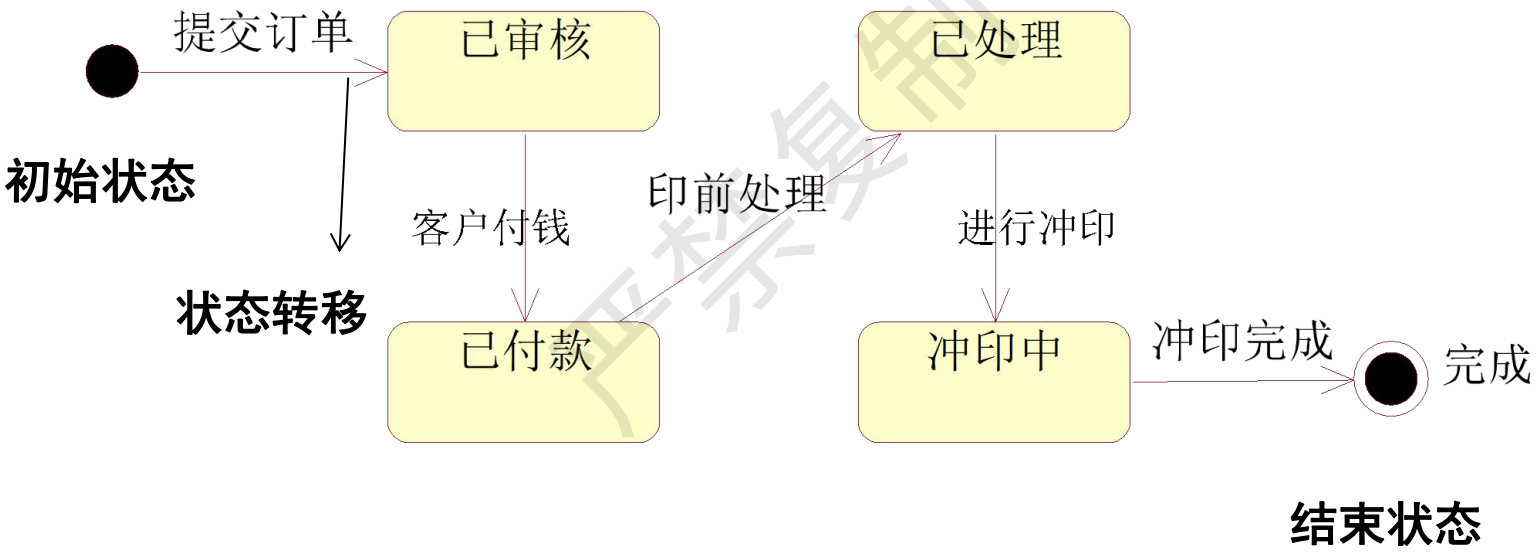
交互图-定时图

定时图:采用一种带数字刻度的时间轴来精确地描述消息的顺序，允许可视化地表示每条生命线的状态变化，适用于实时事件的建模。



状态图

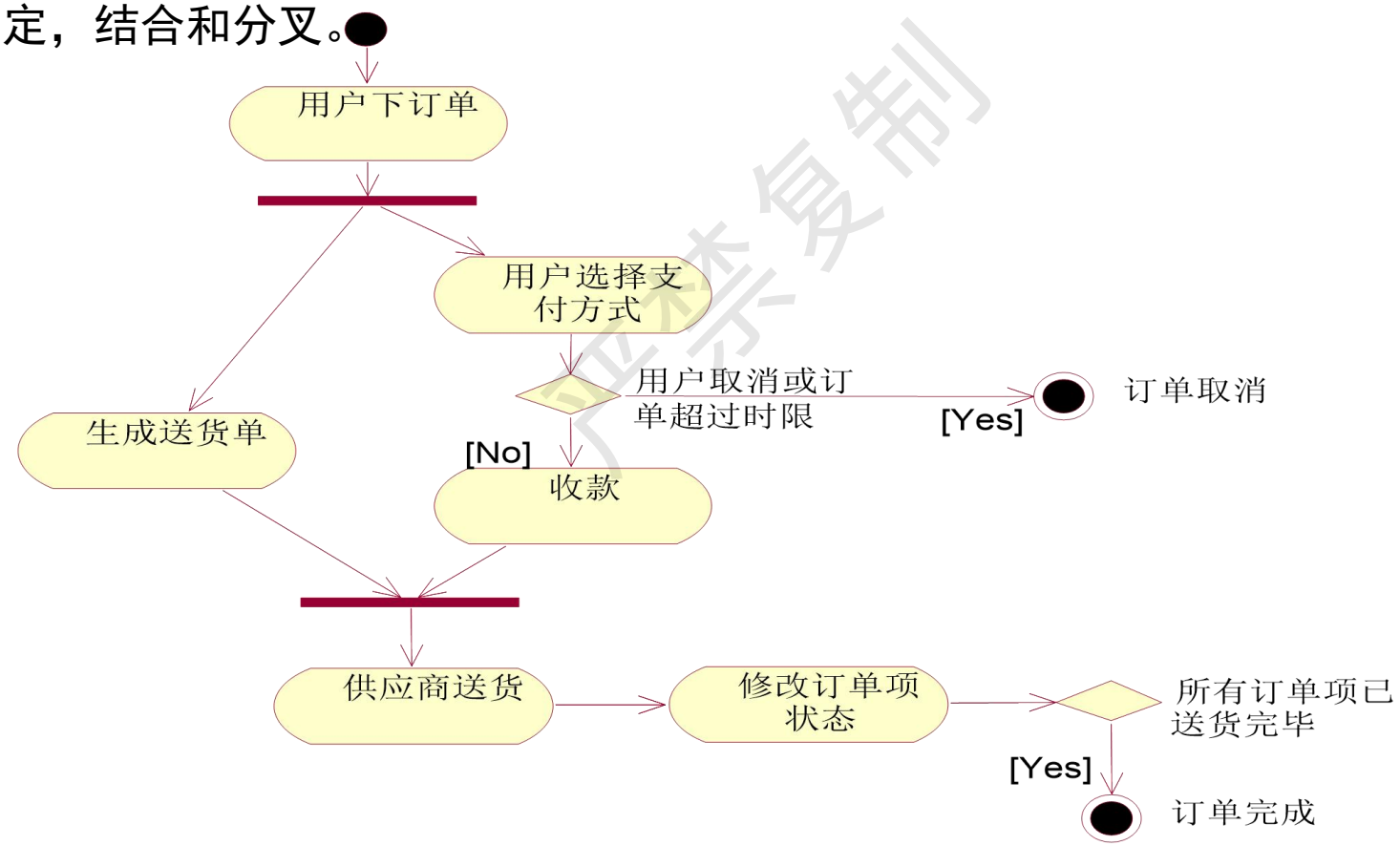
状态图:描述对象在生命周期中的各种状态及状态的转换。
如图：描述一个订单对象的状态变化过程。



状态图适合用于表述在不同用例之间的对象行为，但并不适合于表述包括若干协作的对象行为。通常不会需要对系统中的每一个类绘制相应的状态图，而通常会在业务流程、控制对象、用户界面的设计方面使用状态图。

基本活动图

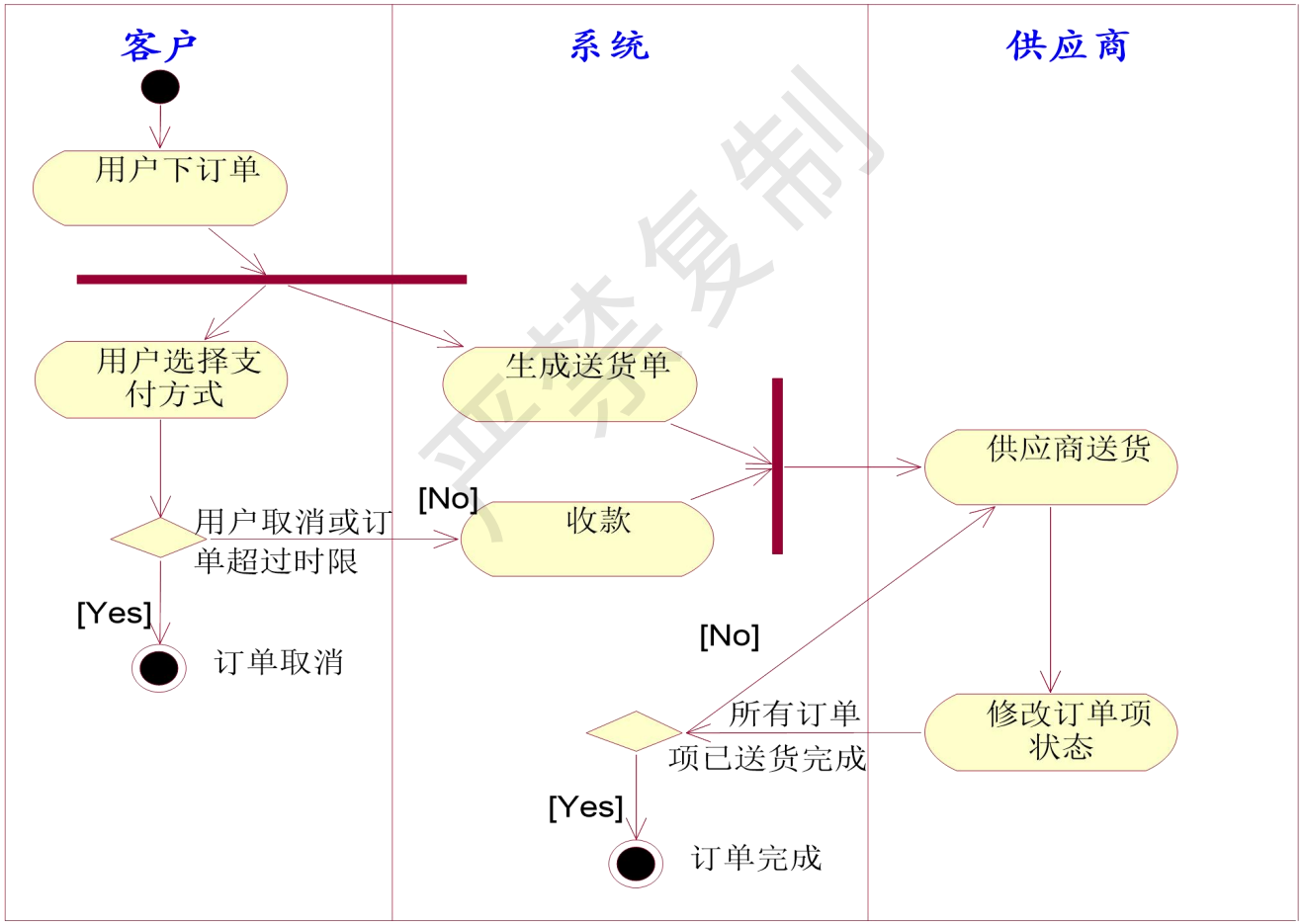
活动图：描述一系列具体动态过程的执行逻辑，展现活动和活动之间转移的控制流，并且它采用一种着重逻辑过程的方式来叙述。相比状态图增加了判定，结合和分叉。



新的开始，新的起点，让我们一起为梦想而努力。

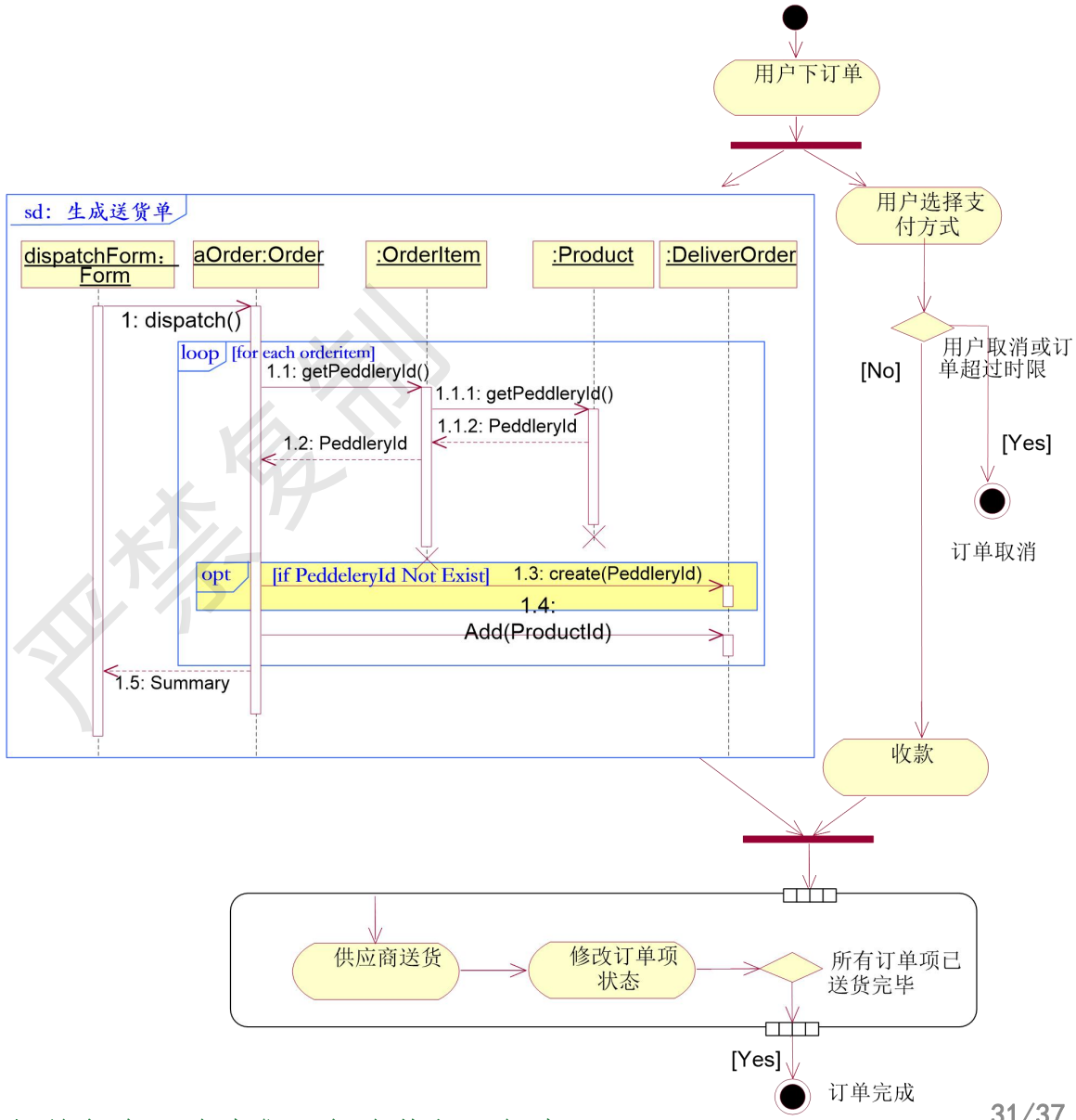
带泳道的活动图

相比活动图更能够说明完成活动的对象



交互图-交互概览图

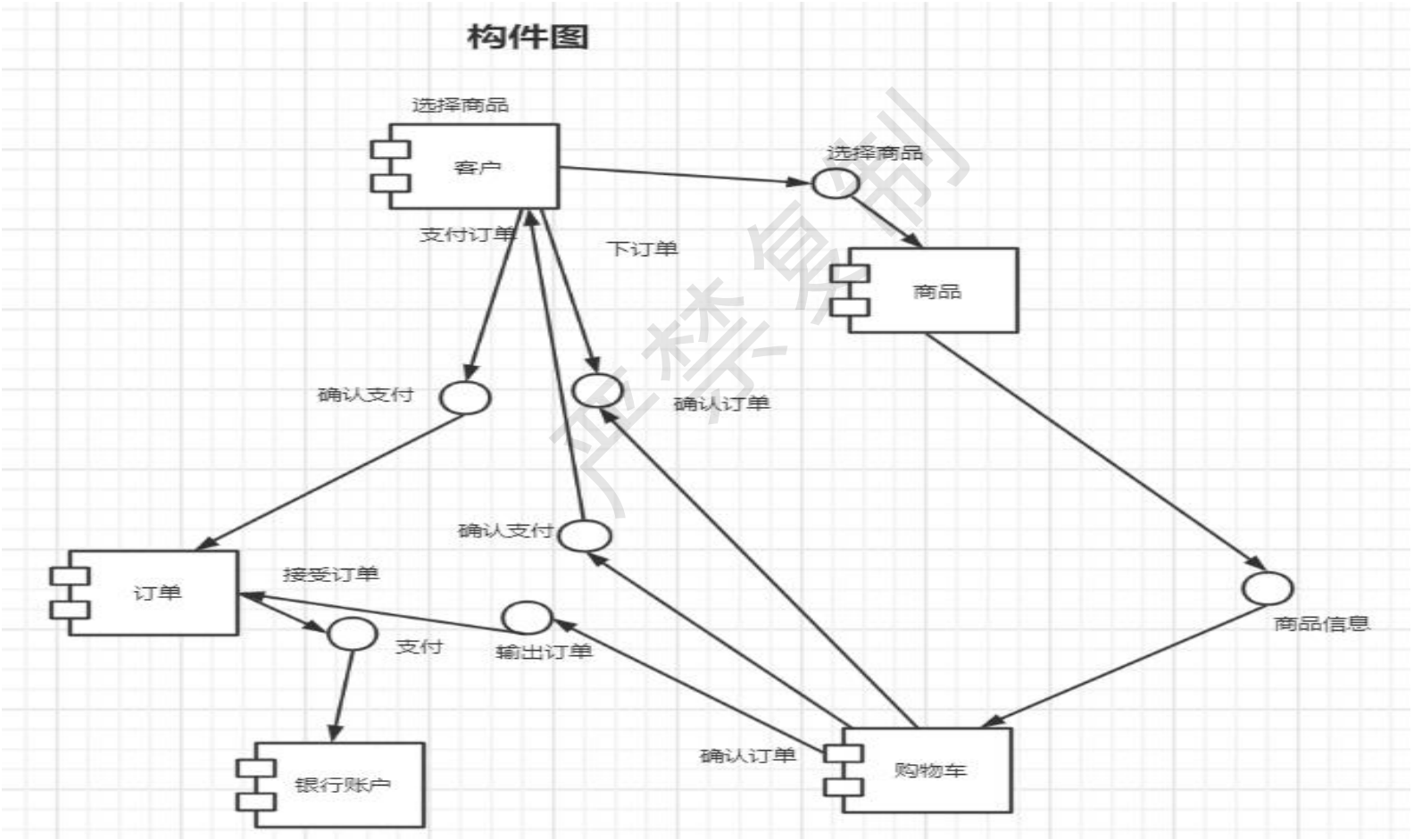
交互概览图是活动图的一种形式，其中节点表示交互图(顺序图，状态图)，交互概览图的大多数符号与活动图相同。



新的开始，新的起点，让我们一起为梦想而努力。

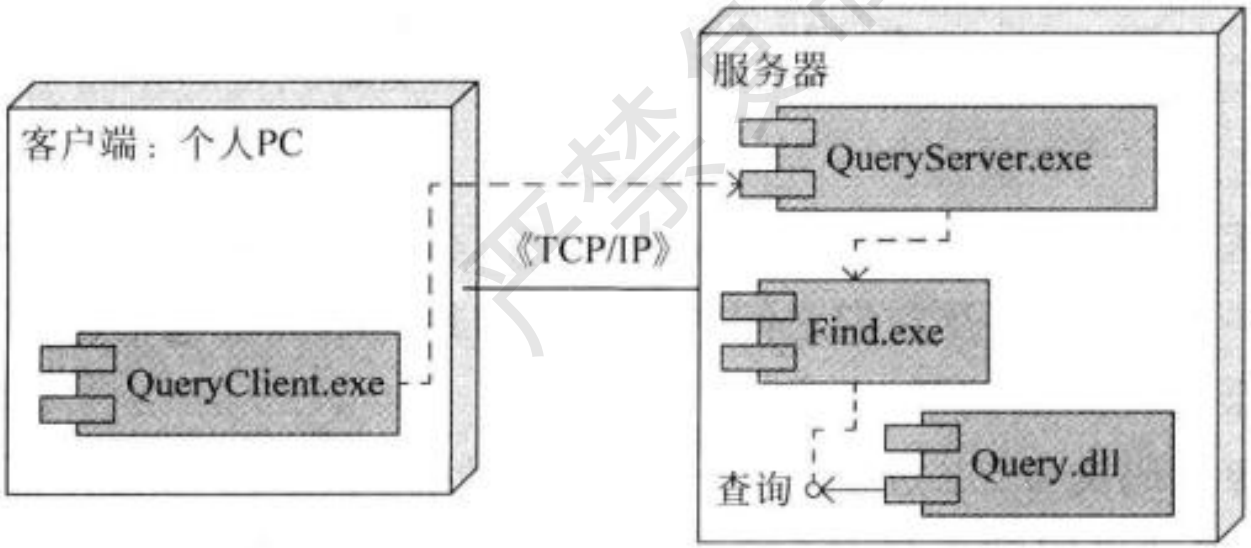
构件图

各构件之间相互依赖关系的图，如客户下单构件图：

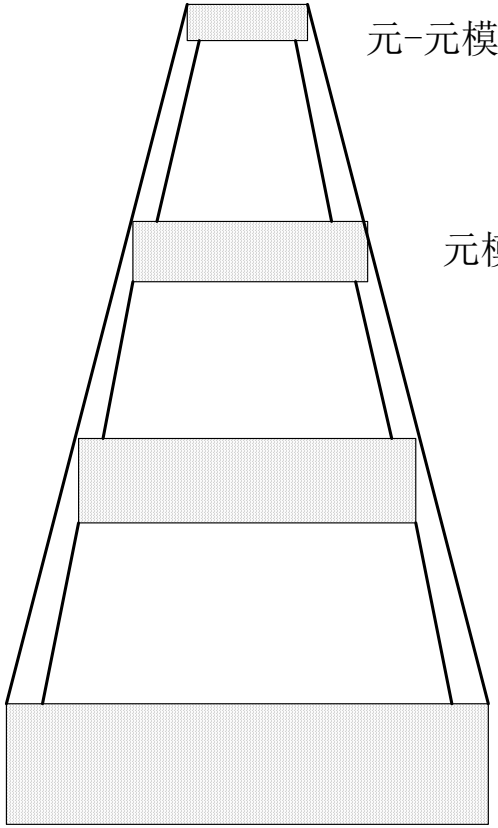


部署图/实施图

描述处理器、硬件设备和软件构件在运行时的架构，它显示系统硬件的物理拓扑结构及在此结构上执行的软件



使用UML建模--UML元模型体系结构



元-元模型

定义描述元模型的规格说明语言，比元模型具有更高的抽象，如：元类，元属性，元操作。

元模型

为给定的建模语言定义规格说明，描述模型的语言，如对象(类)，关联，消息

模型

定义特定软件系统的模型，对现实世界的抽象，描述一个领域，如：储户，账户都是元模型中对象的实例

用户对象

构建给定模型的特定实例，特定的信息领域对象，如张三是储户对象中的一个实例

直接使用UML建模 – 语义约束

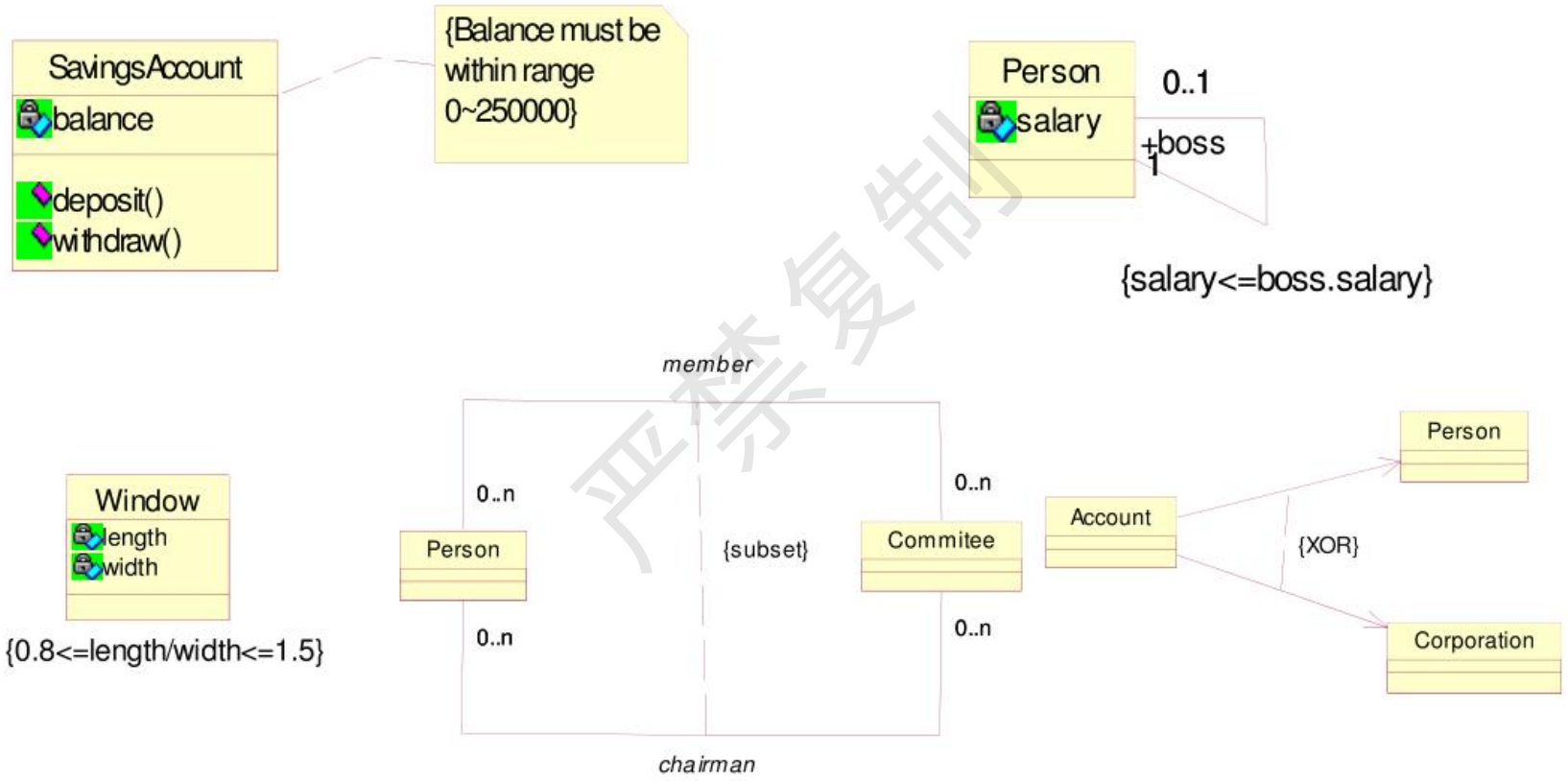
UML中的语义约束由对象约束语言OCL表示，OCL是形式化的无二义的语言。每一个OCL表达式都处于一些UML模型元素的背景下，OCL定义了高级数据类型群、集合、袋和序列。群是抽象数据类型。集合，袋，序列是群的子集。

集合(set):不包含重复元素

袋(bag):允许包含重复元素

序列(sequence)：元素具有有序性的袋

直接使用UML建模 – 语义约束

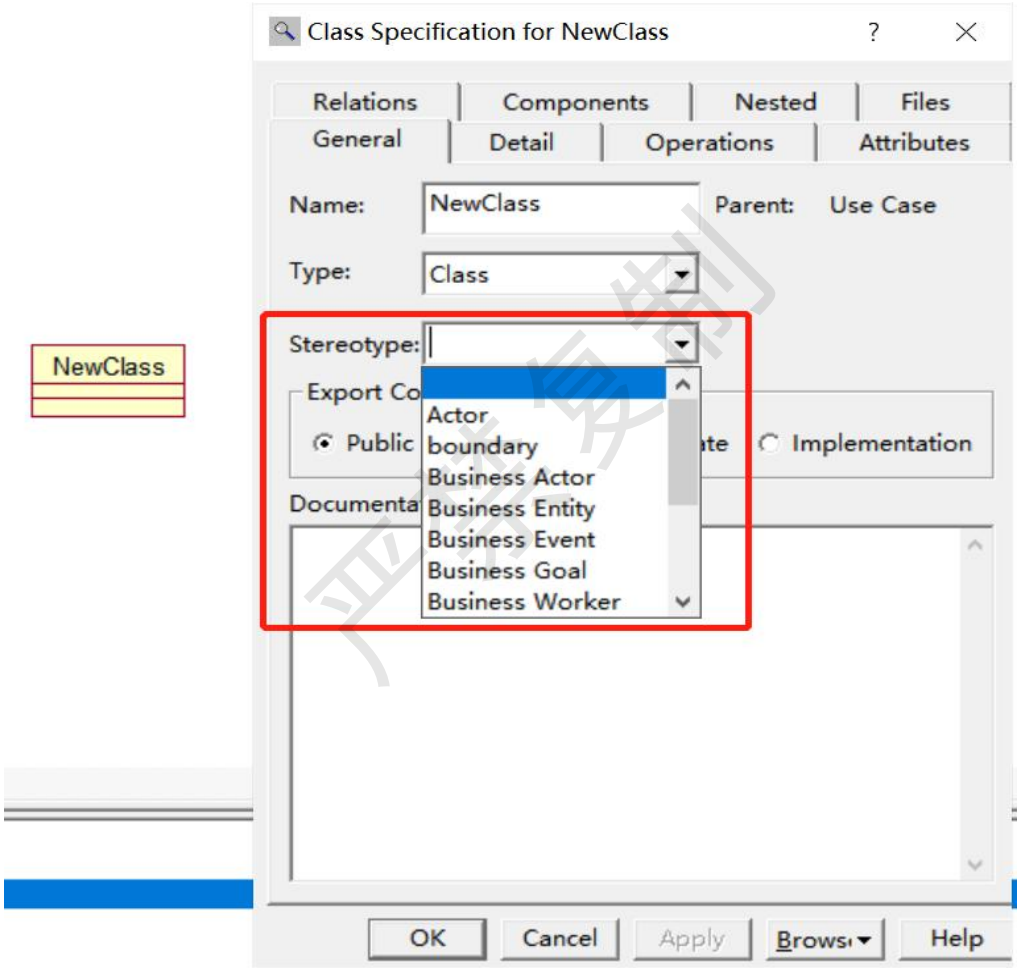


UML中约束的表达形式

直接使用UML建模 – UML中的通用表示

- 字符串：表示有关模型的信息；
- 名字：表示模型元素；
- 标号：不同于编程语言中的标号，是用于表示或说明图形符号的字符串；
- 特殊字符串：表示某一模型元素的特性；
- 类型表达式：声明属性、变量及参数，含义同编程语言中的类型表达式；
- 实体类型(stereotype)：它是UML的扩充机制，运用实体类型可定义新类型的模型元素。

直接使用UML建模 – UML中的通用表示

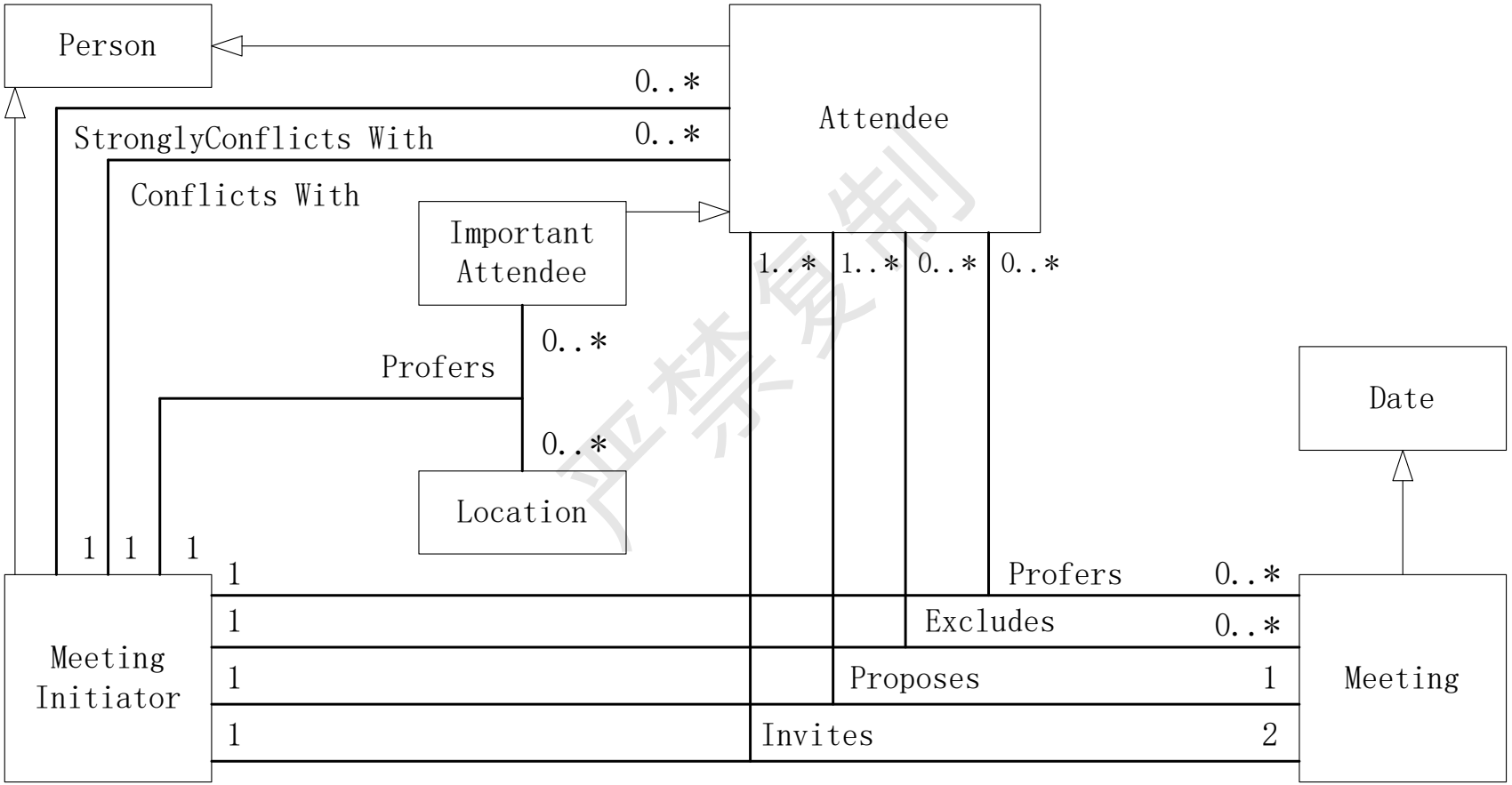


新的开始，新的起点，让我们一起为梦想而努力。

直接使用UML建模 – UML语义部分

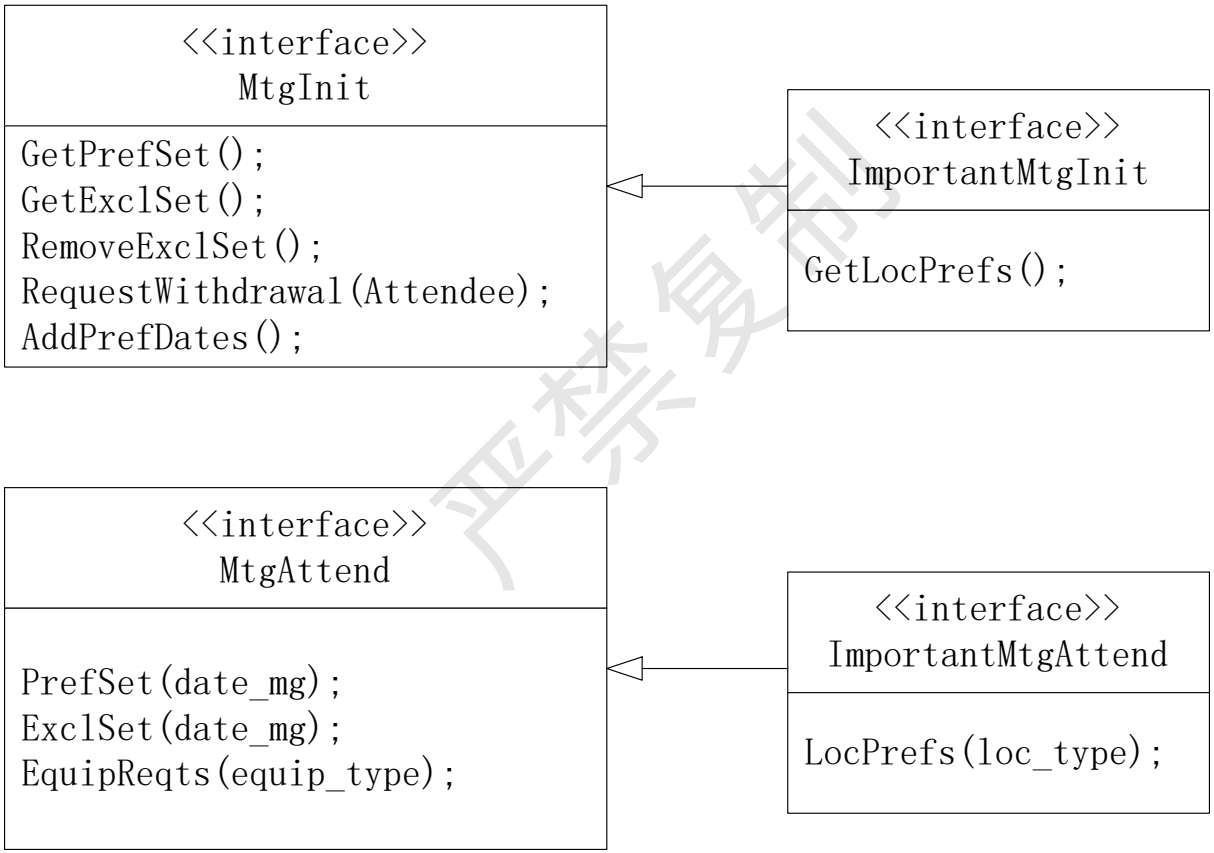
- 通用元素：主要描述UML中各元素的语义。通用元素是UML中的基本构造单位，包括模型元素和视图元素，模型元素用来构造系统，视图元素用来构成系统的表示成分；
- 通用机制：主要描述使UML保持简单和概念上一致的机制的语义。包括定制、标记值、注记、约束、依赖关系、类型-实例、类型-类的对应关系等机制；
- 通用类型：主要描述UML中各种类型的语义。这些类型包括布尔类型、表达式类型、列表类型、多重性类型、名字类型、坐标类型、字符串类型、时间类型、用户自定义类型等。

直接使用UML建模 – 会议安排系统的类图

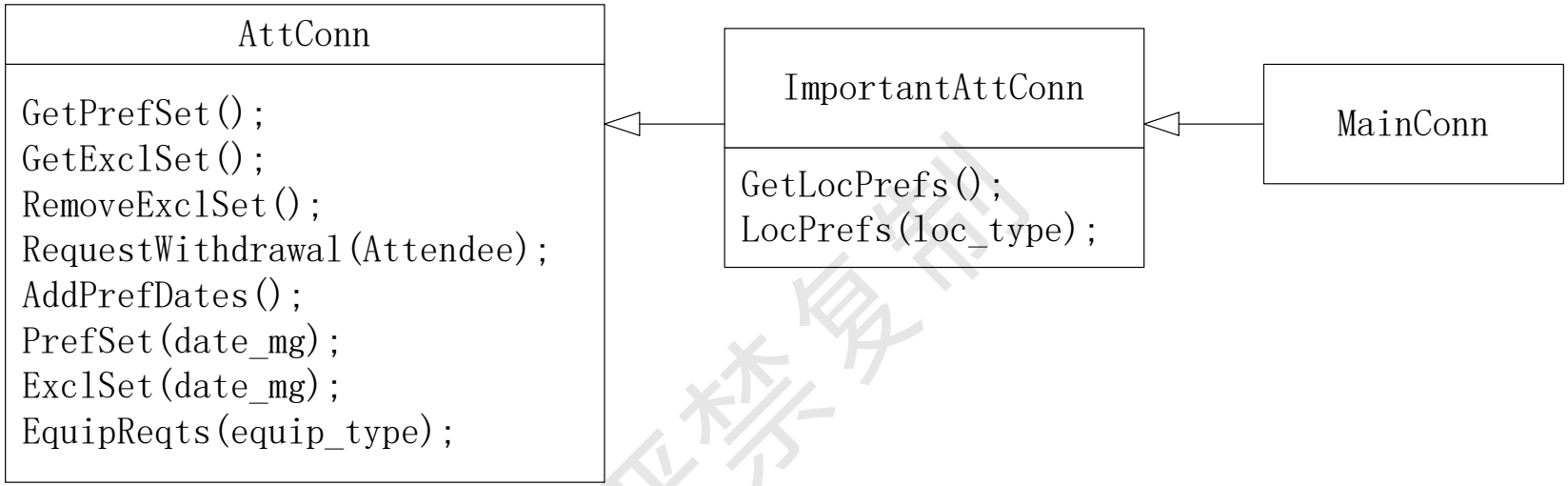


新的开始，新的起点，让我们一起为梦想而努力。

直接使用UML建模 – 会议安排系统的类接口

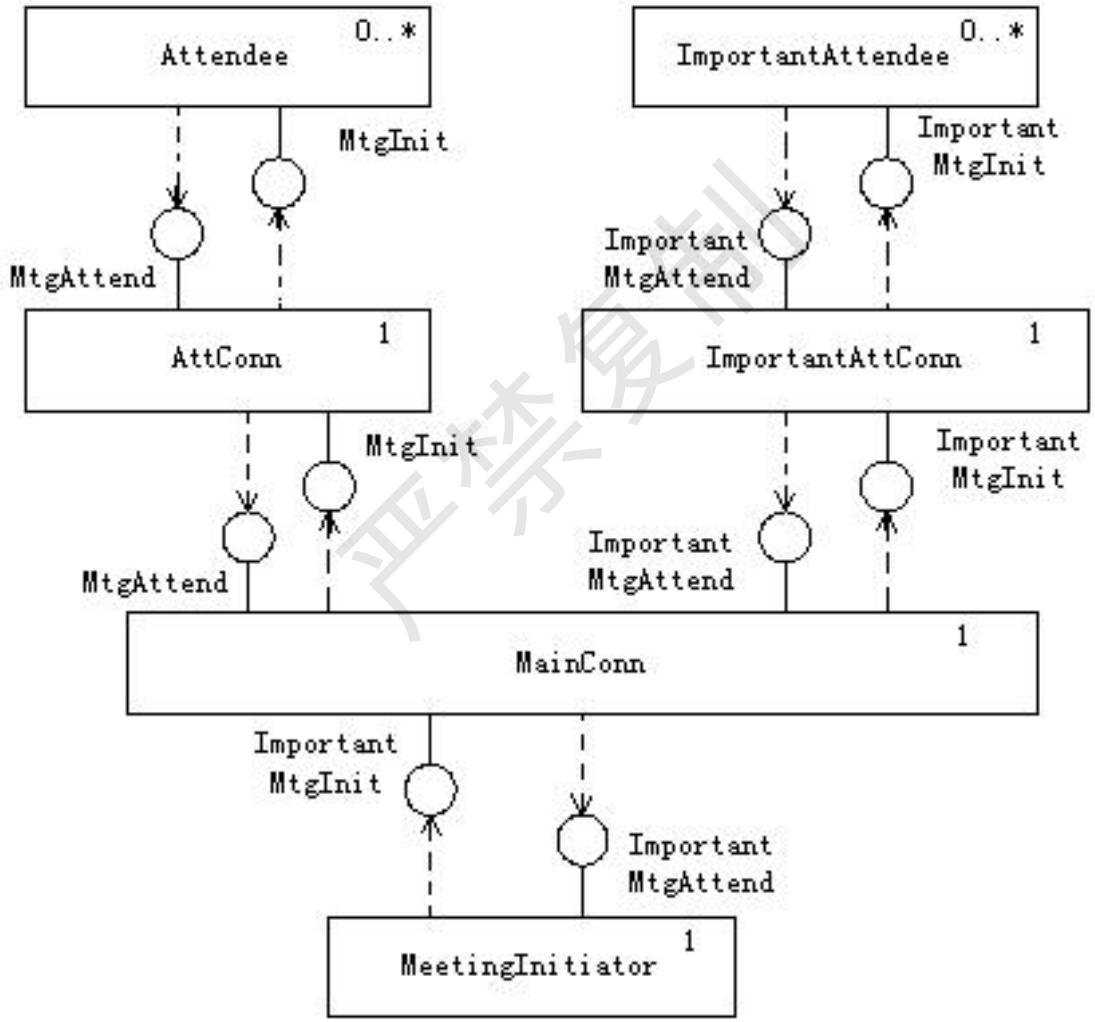


直接使用UML建模 – C2连接件模型



- 为了在UML中为体系结构建模，必须定义连接件，在UML中指定的连接件必须是与应用特定的，必须有固定的接口。
- 为体现C2连接件特性，会议安排系统的连接件类实现了与其关联的构件的同样的接口
- 每个连接件被当做一个简单的类，这个类能够把自己接收到的消息传递给另外一个构件

直接使用UML建模 – 细化的类图



直接使用UML建模 – 会议安排系统的通信图/协作图