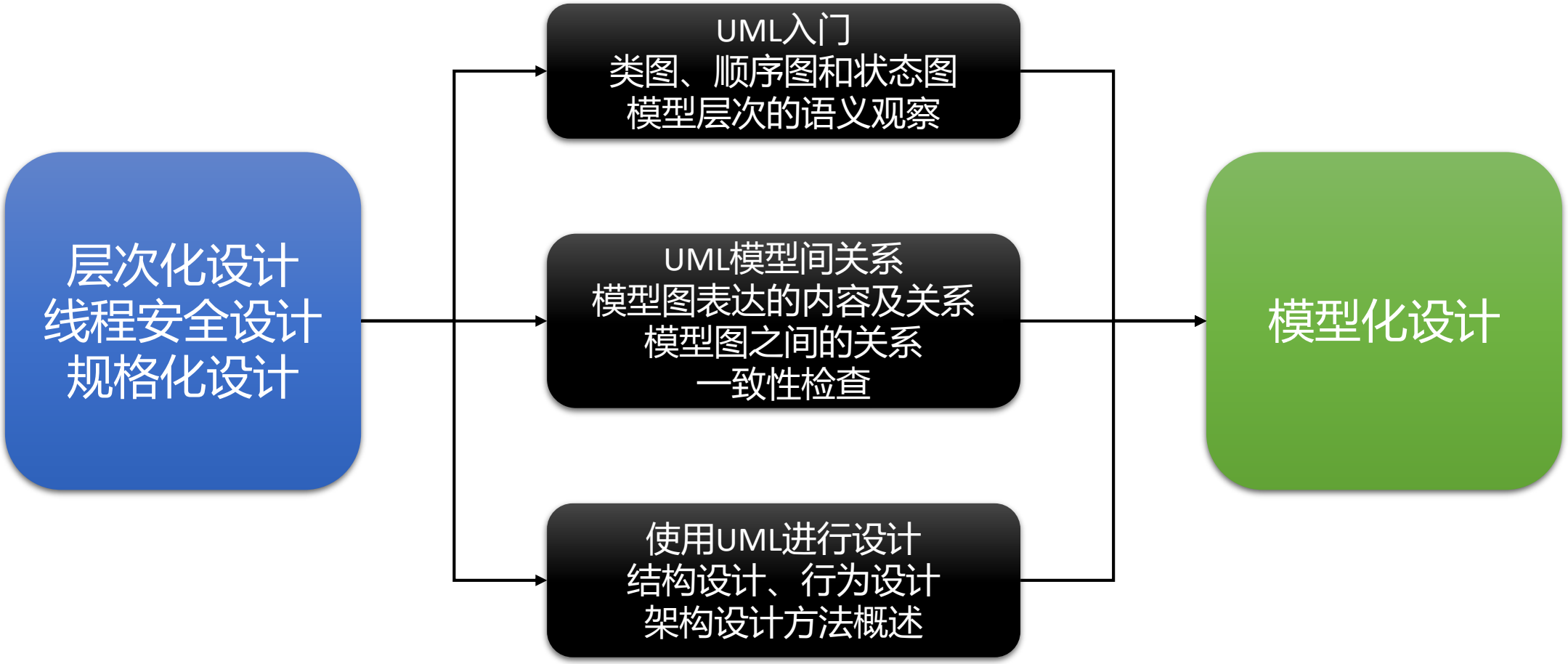
Lec14-综合分析UML模型图

OO2019课程组

北京航空航天大学计算机学院

第四单元内容总览


层次化设计
线程安全设计
规格化设计
UML入门
类图、顺序图和状态图
模型层次的语义观察
UML模型间关系
模型图表达的内容及关系
模型图之间的关系
一致性检查
使用UML进行设计
结构设计、行为设计
架构设计方法概述
模型化设计

提纲

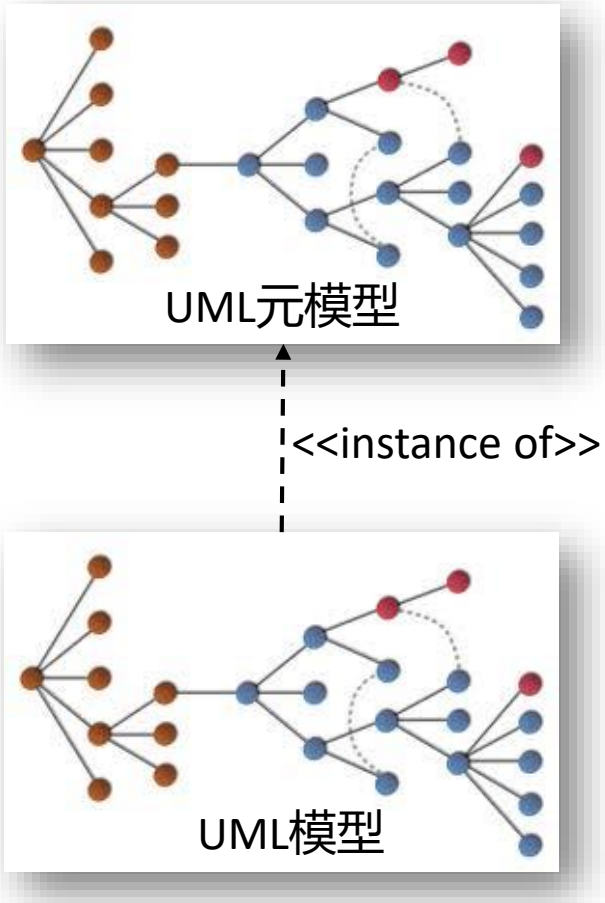
- 什么是模型
- UML类图描述的模型内容
- UML顺序图描述的模型内容
- UML状态图描述的模型内容
- 连接类图内容与顺序图内容
- 连接类图内容与状态图内容
- 连接顺序图内容和状态图内容
- 模型有效性问题
- 有效性检查规则解读
- 作业解析

什么是模型

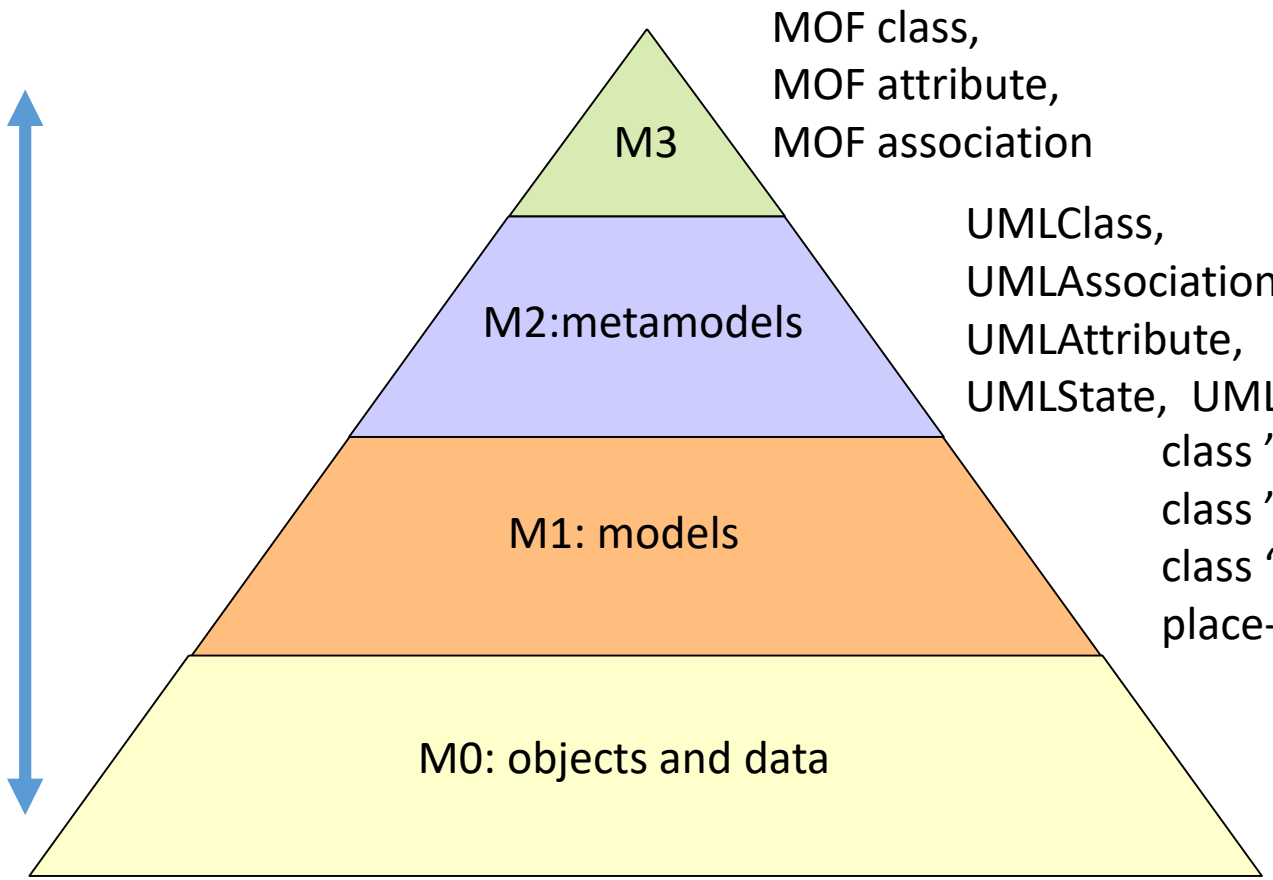
- 模型：所描述内容及其关系
- UML模型：各种<UML***>类型的对象及其关系
- UML建模工具提供可视化图(diagram)，让开发者以‘画图’的方式来构建类型为<UML***>的各种对象，工具后台自动维护和管理这些对象之间的关系
- UML模型中的对象关系
  - 上下层关系：*member*，parent
  - 全局性的引用关系：type, source, target, …

什么是模型

- 模型必须使用一套统一的数据结构来加以表示
  - 所有的类型其实都是事先定义好的：UML****
- 模型就是一个把若干对象连接起来的图(graph)
  - 可视化层的节点对象
  - 可视化层的连接边对象
  - 都是一种UML***类型
  - 这些对象之间存在member-parent或者ref关系
- 为了定义{UML***}，UML语言还定义了诸多中间结构，把这些类型元素连接起来，形成一个更高层次的图(graph)：元模型(meta-model)



什么是模型



MOF class,
MOF attribute,
MOF association

UMLClass,
UMLAssociation
UMLAttribute,
UMLState, UMLTransition, …

class ”Customer”,
class ”Company”,
class “Order”, “OrderItem”,
place-order activity

customer ‘老王’ places an order of computer ‘Lenovo T450s’ from company ‘JD.com’

什么是模型-Java程序类比

- Level 0模型
  - 程序运行起来创建的诸多对象及其连接所形成的graph
- Level 1模型
  - 代码中定义和使用的诸多类、接口、参数、继承、关联及其相互关系
- Level2模型
  - Java语言内置的那些类型定义机制：关键词和句型
- Level 3模型?

类图模型的理解

- 继承关系与接口实现关系的区别
  - 继承在语义上意味着获得父类所**拥有**的内容
  - 接口实现在语义上意味着实现了接口所定义的操作
- UML在类(UMLClass)和接口(UMLInterface)之间有复杂的全组合继承关系
  - 类继承类，类继承接口
  - 接口继承接口，接口继承类
- Java则进行了限制
  - 类继承类，接口继承接口，互相不可交叉

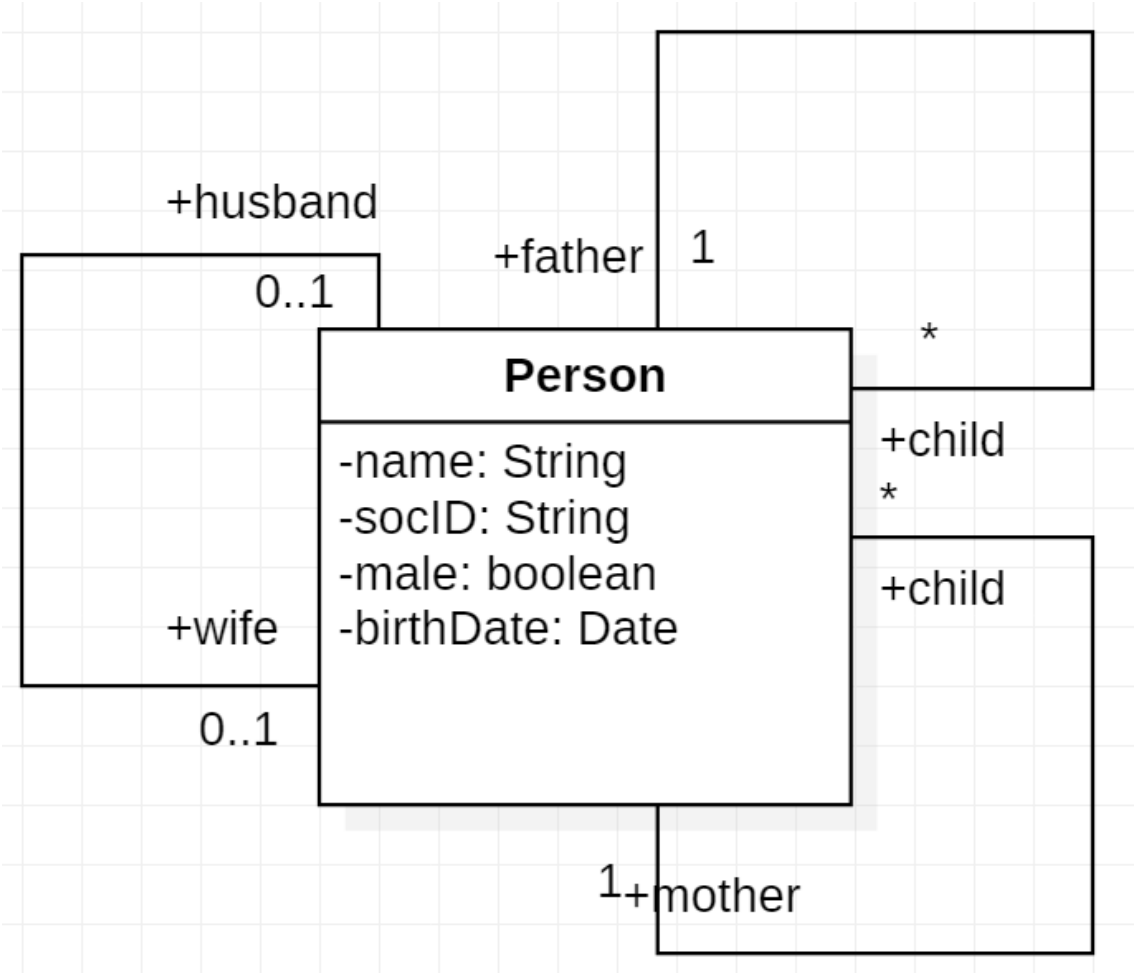
类图模型的理解

- 关联关系
  - 定义**两组对象**之间的关系
  - 双向关系：两个UMLAssociationEnd对象，地位相同
  - 导航(navigable)反映建模者意图，即一个对象是否需要另外一个对象的支持或服务
  - 聚合特性反映end1端连接对象与end2端连接对象之间的关系特性
    - none：两边都不是容器对象
    - shared：构成容器对象与元素对象关系，且共享管理元素对象
    - composite：构成容器对象与元素对象关系，且独享管理元素对象
- 任意两个类之间可以建立多个关联关系，互相独立

类图模型的理解

- 人有父母、子女、配偶
- 中国在相当长一段时间内，一对夫妻只允许生一个孩子
- 中国实施一夫一妻制
- 如何用关联关系把其中的概念和关系表示出来?

如何表达人的兄弟姐妹关系?



建模的核心

- 对诸多细节进行分析，提取共性结果，进行抽象
  - 设出租车的状态有四种：服务状态（响应乘客请求过程中所处的状态），接单状态（被派遣到乘客请求后，去接乘客过程中所处的状态），等待服务状态（无服务无接单的空车运行状态）和停止服务状态（车辆不提供任何服务）。出租者在等待服务状态时可以获得派单，从而变成接单状态，一旦接到乘客，即变为服务状态。出租车只能从等待服务状态变为停止服务状态。

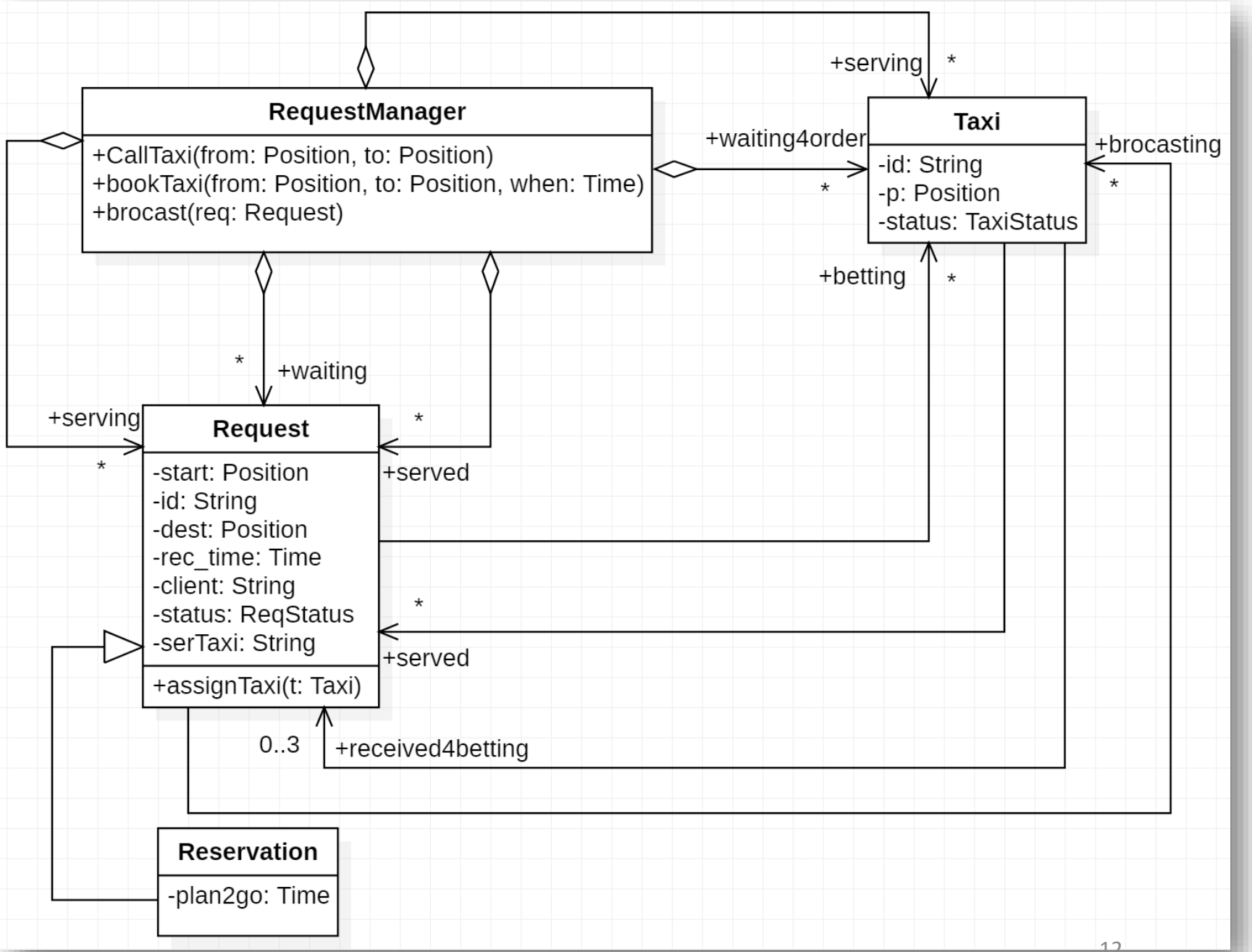
    状态在类图中如何表示？四种状态如何区分？需要在类图中表示状态迁移吗？

  - 对于乘客即时呼叫的叫车请求而言，系统一旦收到乘客的叫车请求，会设置一个抢单时间窗口，在时间窗口内系统向处于等待服务状态的出租车广播发送乘客的叫车请求。系统同时最多向一个出租车发送三个叫车请求。在抢单时间窗口关闭时，系统在抢单的出租车中，随机选择一辆派单。如果没有出租车抢单，则系统提示用户无出租车响应。一辆出租车可以对广播收到的请求进行抢单，但每辆车最多只能被选中并指派一单。

建模的核心

流程式描述

抢单时间窗口其实是一个控制机制，本质上是要在Request与Taxi之间建立关系。而所谓时间窗口无非是关联关系实例的建立时刻和拆除时刻！



建模的核心

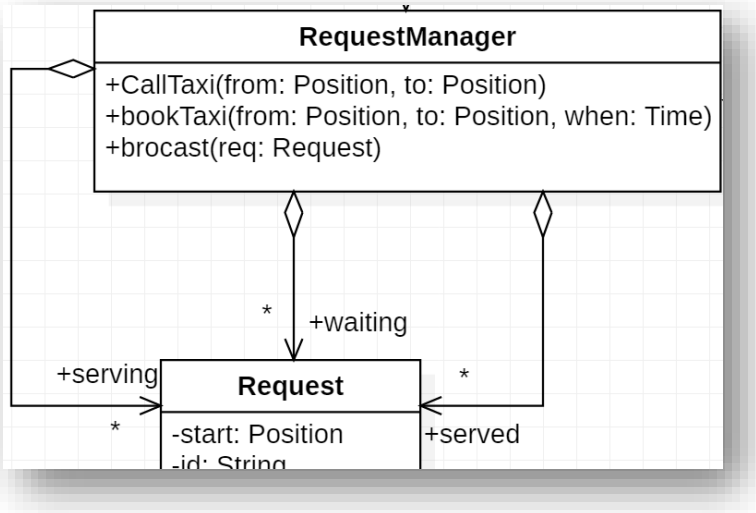
- 类图是UML建模的核心和基础
- 广泛采用的“入门级”方法：名词识别法
  - 具有明确的概念内涵和相应数据内容的名词
- 容易出现混淆的概念
  - 类-名词：这类名词往往蕴含着多维数据，如请求、出租车等
  - 属性-名词：这类名词往往蕴含着单维数据(但可能多例)，如目的地点、出发时刻等
  - 关联角色：这类名词往往蕴含着组合层次和对象实例分类，比如抢单出租车、等待服务出租车
  - 控制策略/机制：这类名词往往是对一种动态控制机制或策略的概括描述，如优先级调度、抢单时间窗口等

建模的核心

- 属性识别
  - 从问题域角度，要完成相应的功能，需要记录和管理的相关数据
  - 出租车需要管理哪些数据?
  - 请求需要管理哪些数据?
- 操作识别
  - 从所识别的数据角度来识别对数据的处理
  - 从系统用户与系统的交互事件角度，分配相应的职责
    - RequestManager类的操作
  - 系统事件处理往往对应着需求描述中的一些策略机制概述

建模的核心

- 关联的识别和处理
  - 从模型角度，只要识别了一个类，意味着就能构造该类的大量实例对象
  - 不同的类往往关心的是这些实例对象的一个子集，因而特别表示出来
  - 通过对象分类/分组，可以有效简化系统设计
  - 关联角色容易被误识别为特殊的类
    - 等待处理的请求 vs 请求
- 关联是为了让一个类使用对方来管理数据或完成自己的行为
  - 关联一般实现为属性数据

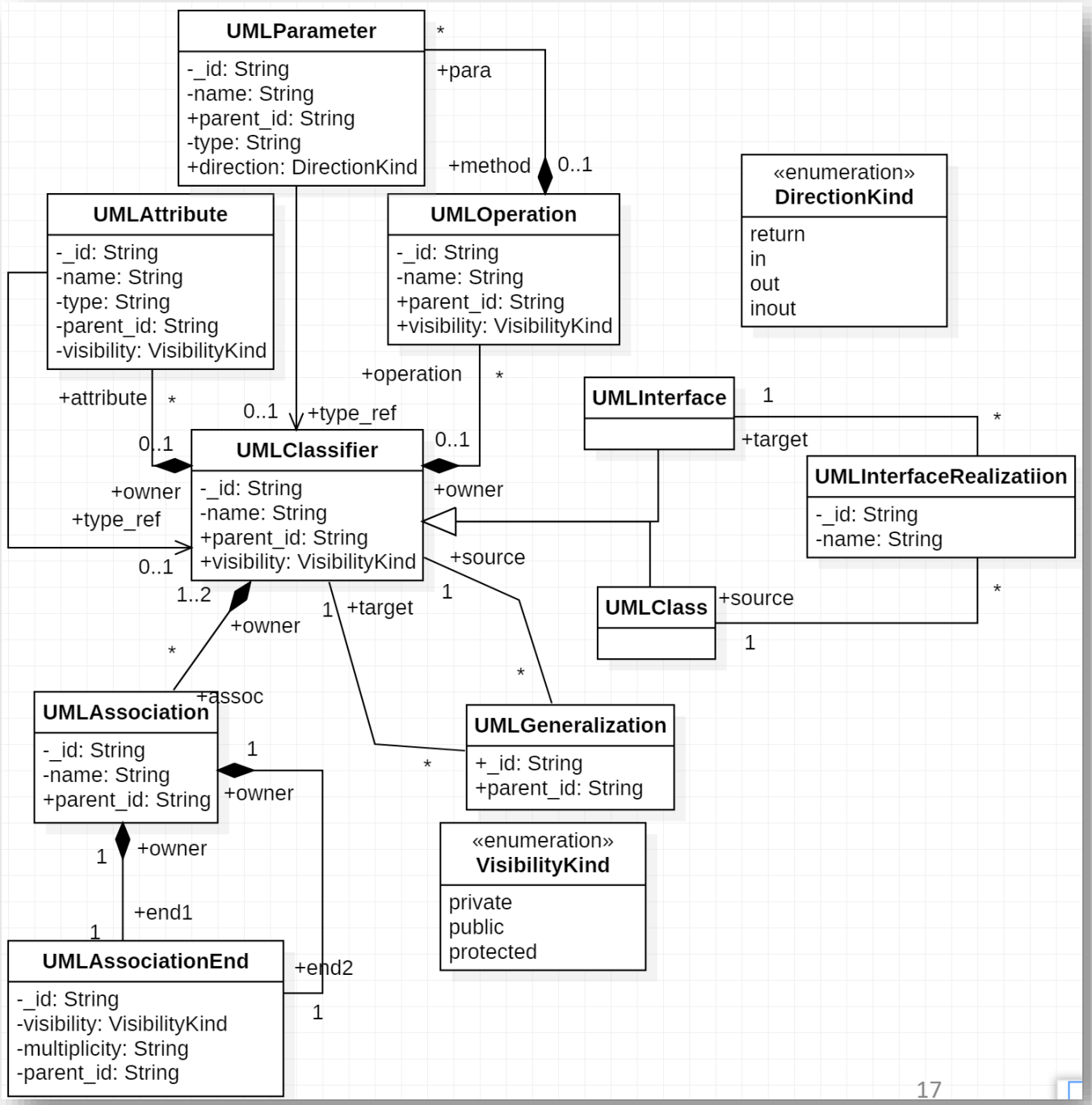


建模的核心

- 继承的识别与处理
  - 问题描述中往往出现多种形态的实体描述：请求、即时请求、预约请求，服务中请求、已服务请求等
  - 其中有些实体描述其实是按照角色的分类描述
  - 从继承角度，核心是分析实体描述是否在数据上有显著不同的内容
    - 请求，即时请求，预约请求
    - 抓住数据抽象这个本质！
- 一个实体需要管理或记录哪些数据根本上来自于系统的领域需求
- 如果一个实体只关心它的行为，不关心数据，说明应该识别为一个接口，比识别为类带来更多灵活性

UML类图及其描述内容

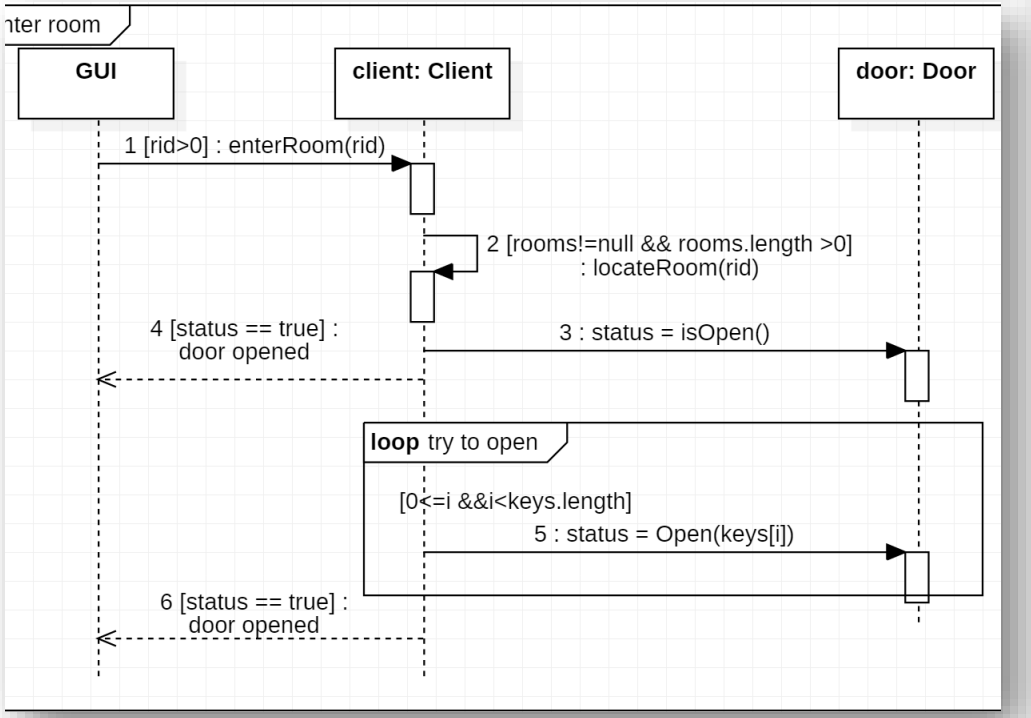
- UML类图提供了一个描述类及其关系的视图
  - 顶层：UMLClass，UMLInterface，UMLAssociation，UMLGeneralization，UMLInterfaceRealization
  - 下一层：UMLAttribute，UMLOperation，UMLParameter，UMLAssociationEnd
  - 再下一层：{<property, value>}
- 可以从输入的{<property, value>}来构造相应的graph
- 在graph上可以进行查询和推理

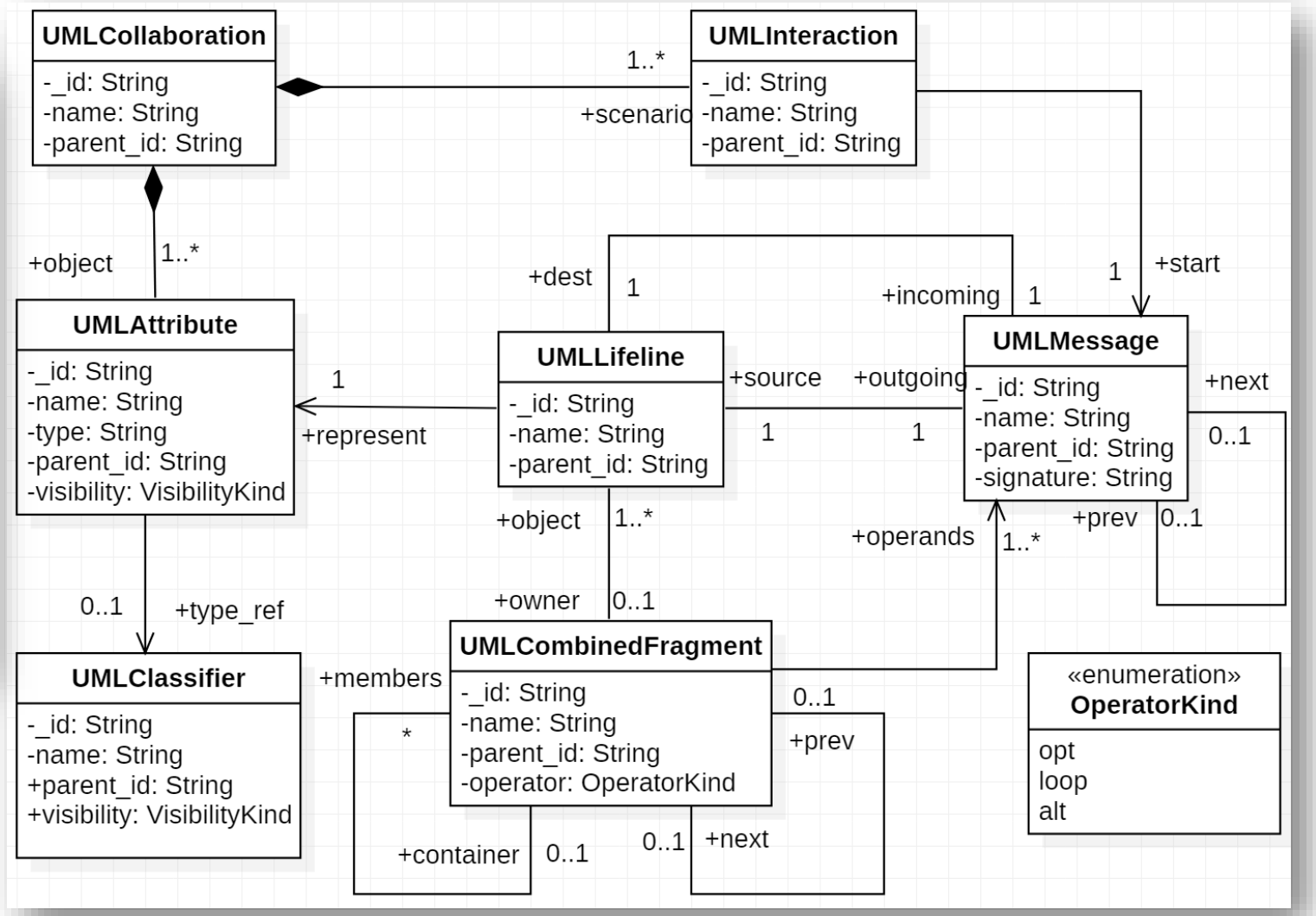


UML顺序图描述的内容

- 顺序图描述了基于消息机制的对象协作关系，应具有明确的协作主题
  - 顶层：协作对象(UMLAttribute)和交互模型(UMLInteraction)
  - 下一层：UMLLifeline，UMLMessage，UMLCombinedFragment
  - 最下层：{<property, value>}
- 可以从输入的{<property, value>}来构造相应的graph
- 可以在graph上进行对象和消息的相关推理分析

UML顺序图描述的内容

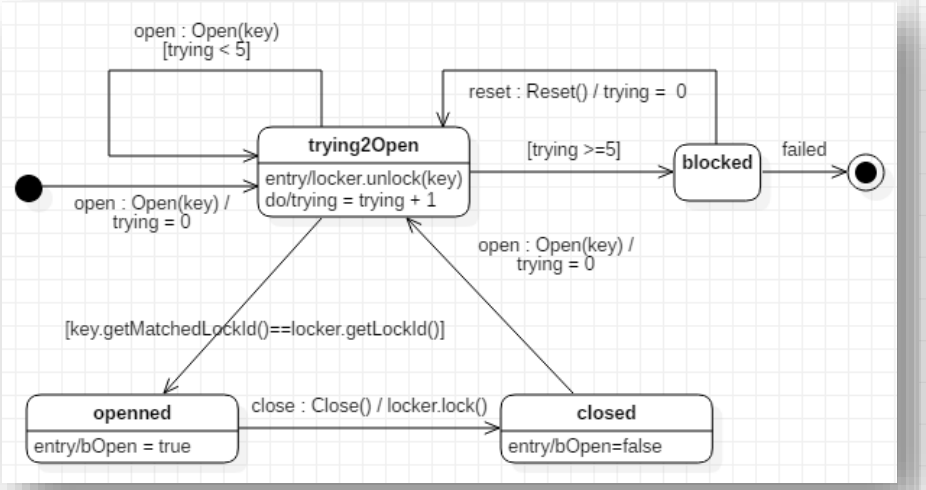


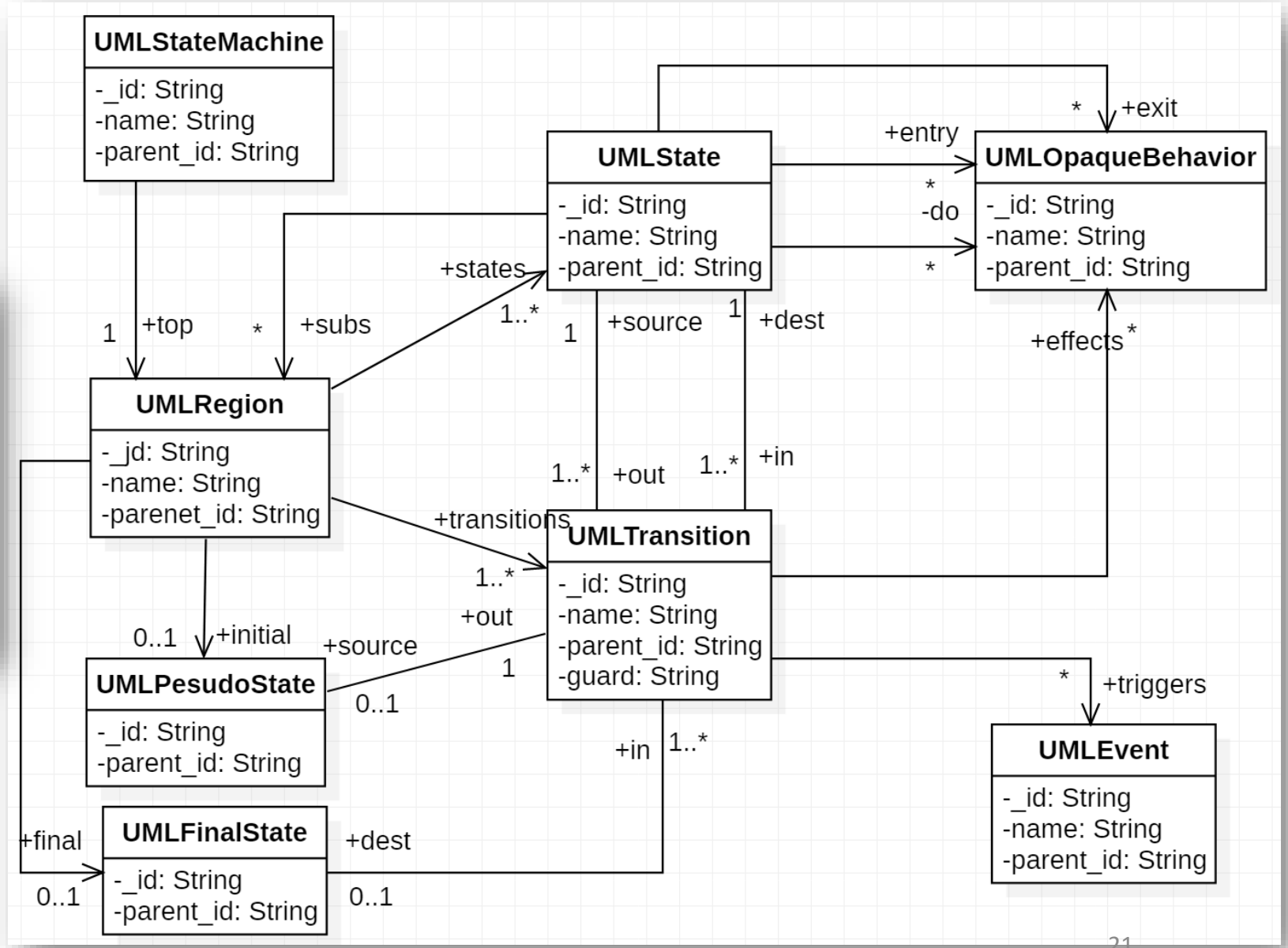


UML状态图描述的内容

- 状态图描述了状态及其关系，一般用来描述一个特定类/组件的行为
  - 顶层：UMLStateMachine， UMLRegion
  - 下一层：UMLState, UMLTransition, UMLEvent, UMLOpaqueBehavior
  - 最下层：{<property, value>}
- 依据{<property, value>}可以构造出相应的graph
- 在graph可以进行状态和迁移行为的推理分析

UML状态图描述的内容



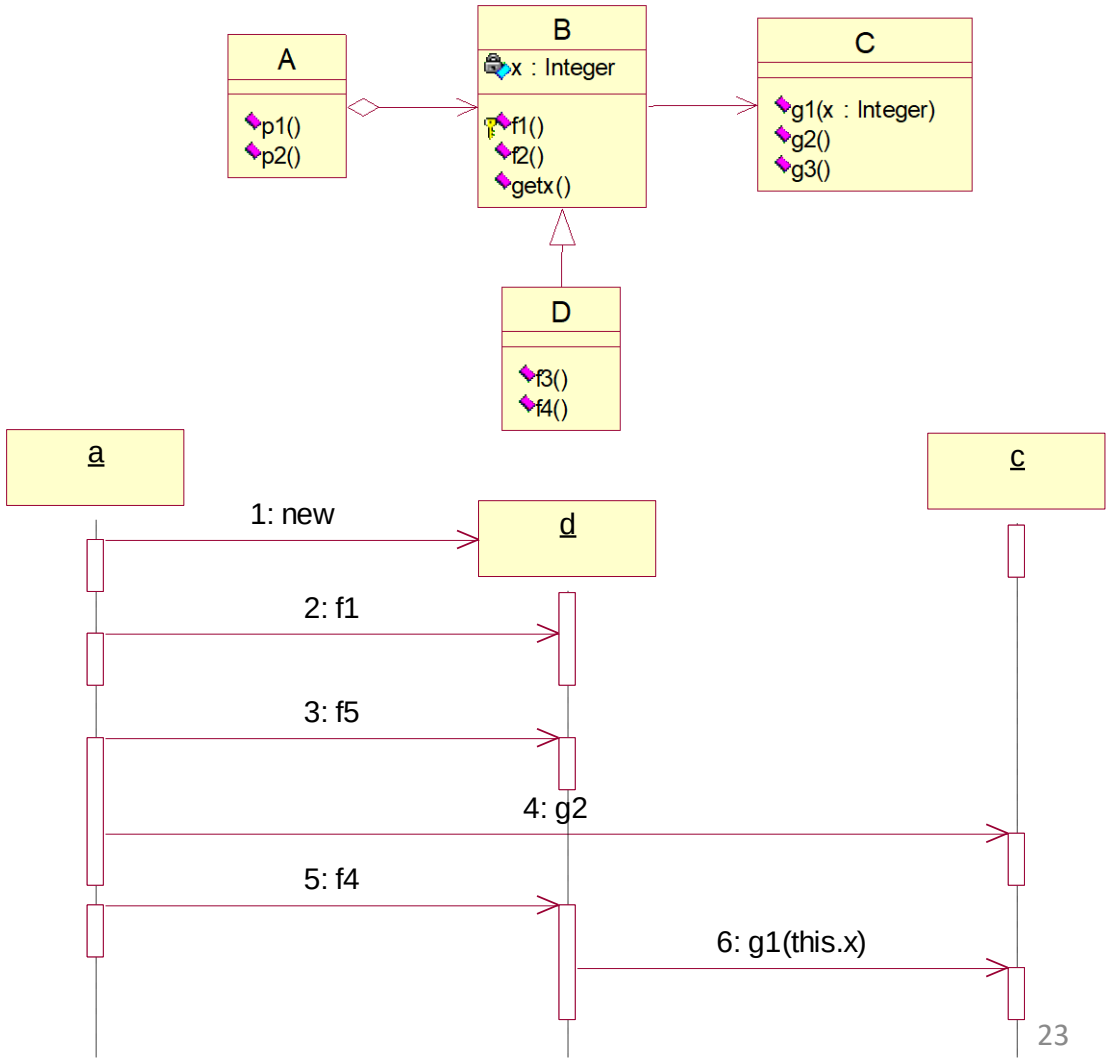


类图内容与顺序图内容的关系

- 基础：顺序图中的UMLAttribute引用到类图中定义的类
- 推导：每个发送给UMLLifeline的UMLMessage都带来一个问题
  - 该UMLLifeline关联的UMLAttribute是否能够处理?
- 从OO角度来看
  - 消息是一种交互机制，映射到消息receiver的operation
  - 同步operation→messageSort == synchCall
  - 异步operation→messageSort == asynchCall
- starUML提供了一个signature属性，用来建立这种连接关系

Example: 指出不一致

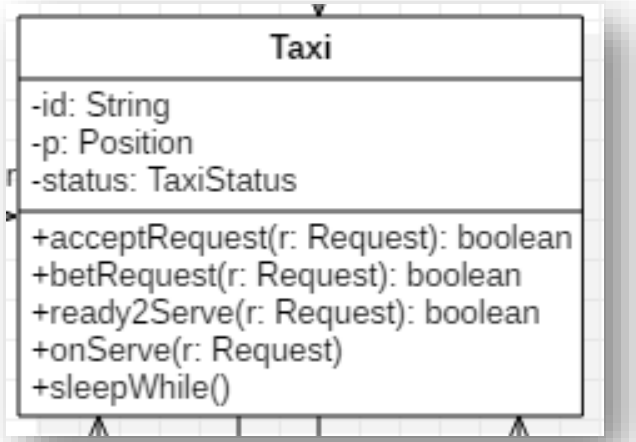
- 检查顺序图中的消息与类图中的相关内容的一致性
- 检查规则
  - Sender对象与receiver对象之间是否有关联?
  - 消息的signature与receiver提供的operation是否匹配?
  - receiver对象的相应operation能否被外部访问?

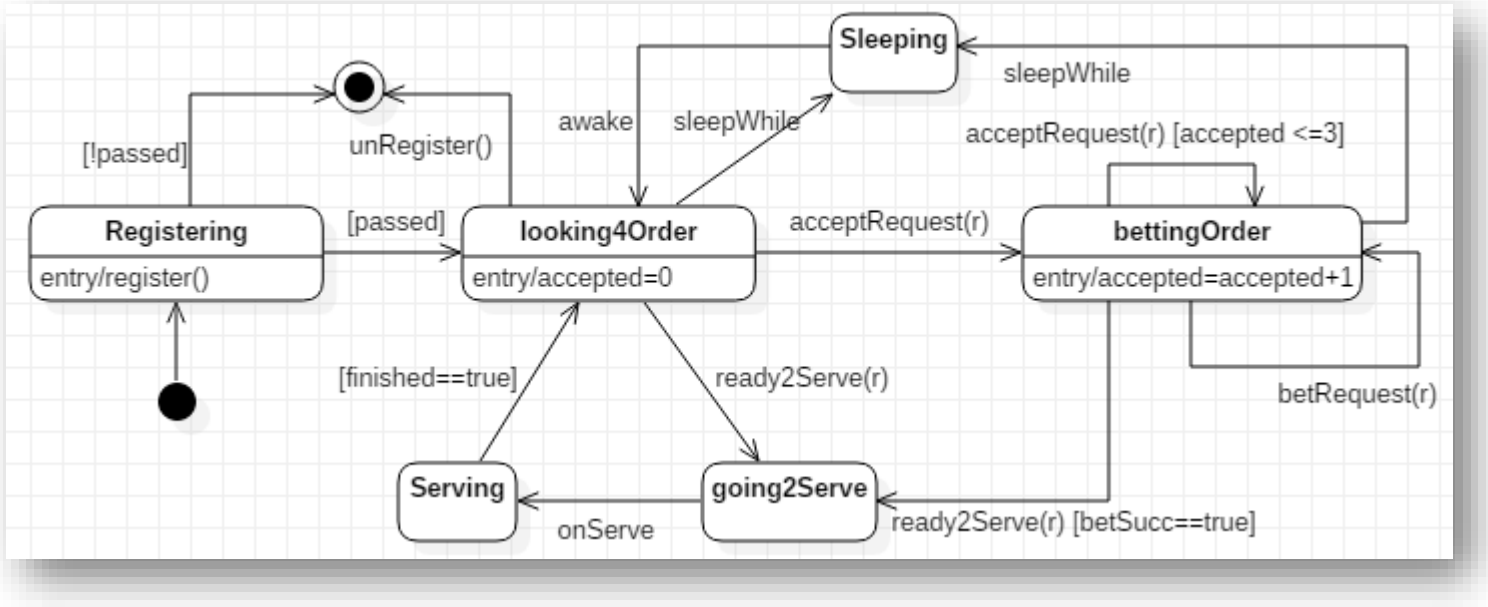


类图内容与状态图内容的关系

- 基础：状态图表示一个类的行为
  - UMLStateMachine的parent必须引用到某个UMLClass
- 推导：状态图中的内容必然都和相应类中的内容对应起来
  - 状态行为：所在类的行为
  - 状态迁移触发：所在类的行为
  - 状态迁移守护：对所在类属性数据取值的检查
  - 状态迁移效果行为：对其他类行为的触发

Example：指出不一致

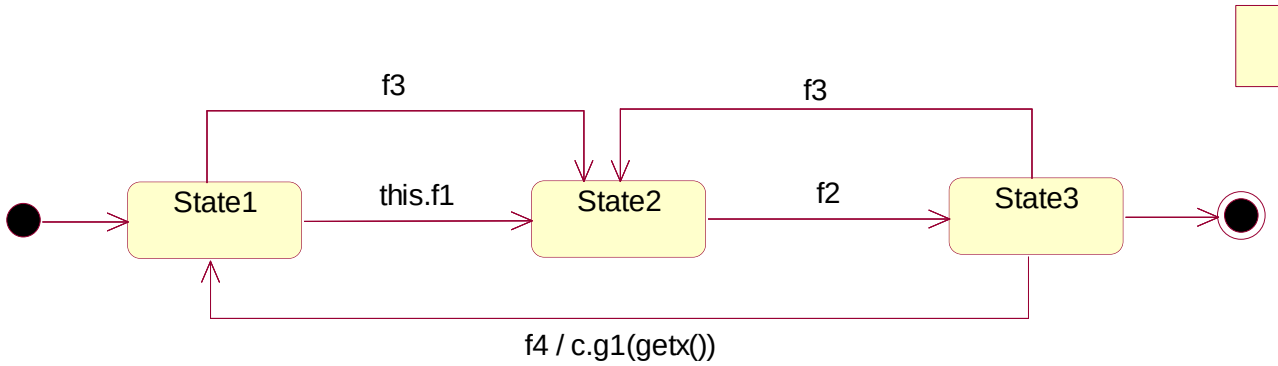


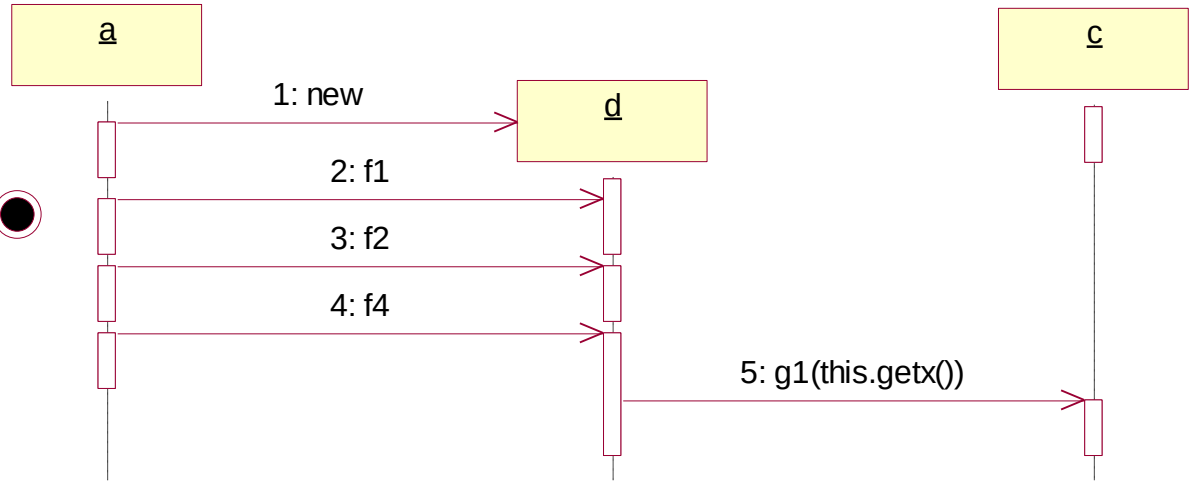


顺序图内容与状态图内容的关系

- 基础：顺序描述多个对象之间的交互行为
  - 消息与对象操作关联起来
- 事实：在消息交互过程中，对象状态可能会发生变化
  - 消息接收时所处状态
  - 消息处理后所处状态
- 推导：在给定状态下是否能够响应的消息？
  - 对照状态图进行检查

Example: 指出不一致





- 检查对象d是否具备处理这些消息的能力
  - 顺着消息连接检查接收消息的对象当前处于什么状态
  - 对照状态图检查在相应状态下是否可以响应发送来的消息

模型有效性问题

- 模型有效性是建模中的一个核心问题
  - 每个图中的元素有效
  - 不同图元素之间如果**关联**，相关属性应该一致
- 不一致的模型会导致最终实现的系统无法集成，或者运行时出现莫名其妙的错误
- 模型有效性和一致性是个复杂问题，同时也是一个学术研究问题
  - 课程目标：理解这个问题，并能就一些简单的规则进行推理分析

模型有效性与一致性问题

- 是判定问题
  - 需要定义清楚判定规则
- 举例：类操作定义与使用的不一致
  - 类A只提供了操作func
  - 类B关联到类A，并在一个顺序图中给A对象发送消息，对应的操作为func1
- 举例：循环继承带来的无效继承范围
  - 类A定义了属性x
  - 类B继承了类A，定义了属性y
  - 类A也继承了类B

关于类的检查规则

- starUML定义的几个规则:
  - (UML002) Name is already defined.
    - If element has a name, then it should be unique in the namespace.
    - Applies to: `UMLModelElement`.
    - Exceptions: `UMLOperation`.
  - (UML003) Conflict with inherited attributes.
    - Applies to: UMLAttribute
  - (UML004) Signature conflict.
    - Same signature is not allowed in a classifier.
    - →不允许在一个classifier中新定义两个signature一致的operation
    - Applies to: UMLOperation

https://docs.staruml.io/user-guide/validation-rules

关于类的检查规则

- (UML007) Duplicated generalizations.
  - Do not make duplicated generalizations from the same element.
  - Applies to: UMLClassifier.
- (UML008) Circular generalizations.
  - Do not generalize from one of the children.
  - Applies to: UMLClassifier.
- (UML009) Duplicated realizations.
  - Applies to: UMLClassifier.
- (UML010) Duplicated role names of associated classifiers.
  - Applies to: UMLClassifier

关于状态图的检查规则

- (UML021) An initial vertex can have at most one outgoing transition.
  - Applies To: `UMLPseudostate (kind = 'initial')`
- (UML022) The outgoing transition from an initial vertex must not have a trigger or guard.
  - Applies To: UMLPseudostate (kind = 'initial')
- (UML033) A final state cannot have any outgoing transitions.
  - Applies To: UMLFinalState
- (UML044) The classifier context of a state machine cannot be an interface.
  - Applies To: UMLStateMachine

关于顺序图的检查规则

- A lifeline must represent an attribute defined in the same collaboration model
  - Applies to: UMLLifeline, UMLAttribute

- A UMLMessage without explicit messageSort property specified, it must be a synchCall message
  - Applies to: UMLMessage

关于顺序图/状态图和类图的检查规则

- 应该为顺序图中的对象属性定义相应的类型，且该类型应该在类图中有定义
  - Applies to: UMLCollaboration, UMLAttribute, UMLClassifier
- 针对所有发送到一个UMLLifeline的UMLMessage (messageSort为synchCall)，则该lifeline所关联的attribute中一定存在一个UMLOperation，使得message.name=operation.name
- 状态图中的每个UMLTransition中的trigger，都应关联到context的某个UMLOperation

作业解析

- 课程的最后一次project
- 增加对顺序图和状态图的解析
- 增加关于顺序图和状态图的查询命令
- 增加模型元素有效性的检查功能
  - 基本规则