管理信息系统
Management Information Systems

王谦　博士/副教授
南开大学商学院管理科学与工程系
wangqian70@nankai.edu.cn

Chapter 10

系统设计与开发

（面向对象方法）

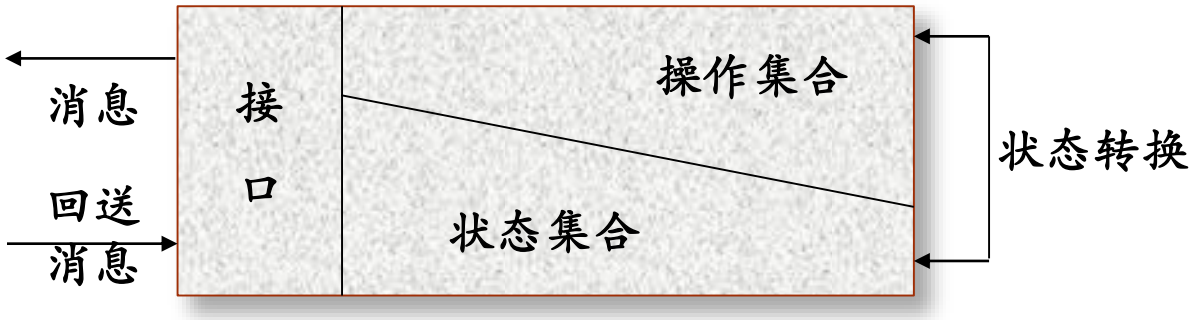
10.1 面向对象系统开发方法论

对象

- 在面向对象系统中，问题对象是基本的运行实体，是由一组数据和施加于这些数据之上的一组操作封闭而成的

对象的特征

- 模块独立性
- 动态连接性
- 易维护性



对象的自动机表示

对象划分的原则

- 寻求大系统中事物的共性，将所有共同的系统成分确定为一个对象
- 系统目的不同，对象划分也就不同

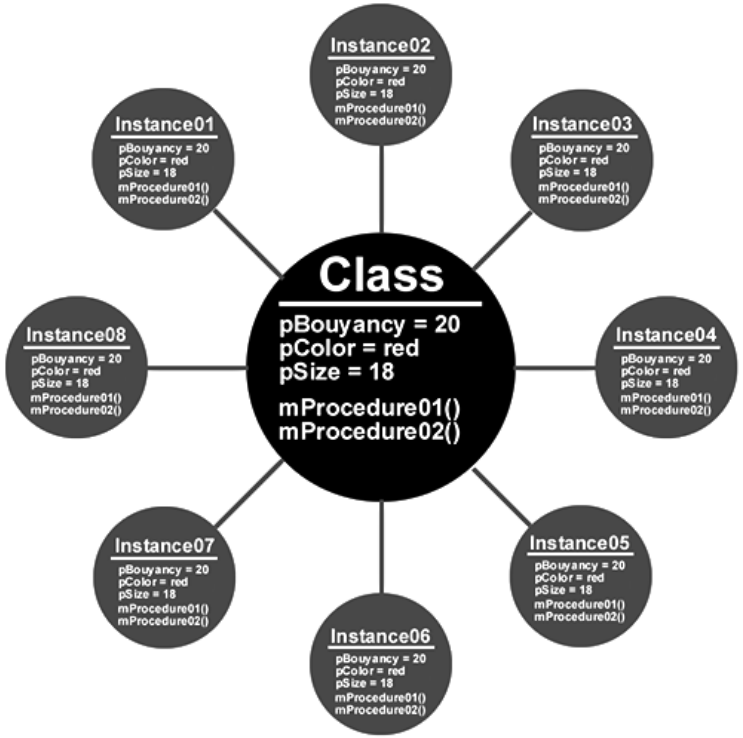
10.1 面向对象系统开发方法论

类

- 现实世界中有许多内部状态和外部行为相似的对象，由这些对象构成的集合就是类

构成类的基本要素

- 标识：类的名称，用于区分其他类
- 继承描述：指子类承袭的父类的名称，以及继承得到的结构与功能
- 数据结构：是对该类数据的组织结构的描述
- 操作：指该类通用功能的具体实现方法
- 接口：指面向其他类的统一的外部通讯协议



10.1 面向对象系统开发方法论

消息

- 消息是实现对象与对象间相互合作的通信载体,是连接对象的纽带
- 从实现的角度看，消息就是请求对象执行某个处理或提供某些信息的要求，既可以是数据流，又可以是控制流

消息传递

- 当一个消息发送给某个对象时，包含要求接收对象去执行某些活动的信息，接收到消息的对象经过解释予以响应，对象间的这种相互合作需要一个机制协助进行，这样的机制称为“消息传递”

消息传递模型

10.1 面向对象系统开发方法论

继承

- 继承关系经常也被称为“is-a”关系，用来表示应用领域中的抽象和结构
- 有继承关系的类之间应具有如下特性：
  - 共享性
  - 差异性
  - 层次性
- 继承最重要的优点在于支持重用
- 是指一个类（即称子类）因承袭而具有另一个类（或称父类）的能力和特征的机制或关系
- 是一种联结类的层次模型，允许并鼓励类的重用

10.1 面向对象系统开发方法论

面向对象

- “面向对象”是一种认识客观世界的认知方法学。从人们思维模型和认识事物的角度，面向对象很自然地与客观世界的固有特征相对应
- “面向对象”亦是一种解决问题的思维方法。这种方法描述的现实世界模型贴切合理，符合人们认识世界的思维方法

10.1 面向对象系统开发方法论

面向对象的含义

- 世界由许许多多的不同对象构成，每一个对象都有自己的运动规律和内部状态，不同对象之间的相互作用和通讯构成了完整的客观世界
- 从组织结构模型化客观世界，将对象作为需求分析和系统设计的核心和主体，把整个问题域抽象成为相互通讯的一组对象集合，并引用科学方法论中的分类思想，将相似或相近的一组对象聚合成类，采用各种手段将相似的类组织起来，实现问题空间到解空间的映射

数据抽象

数据抽象类型

\+ 继承机制

面向对象

10.1 面向对象系统开发方法论

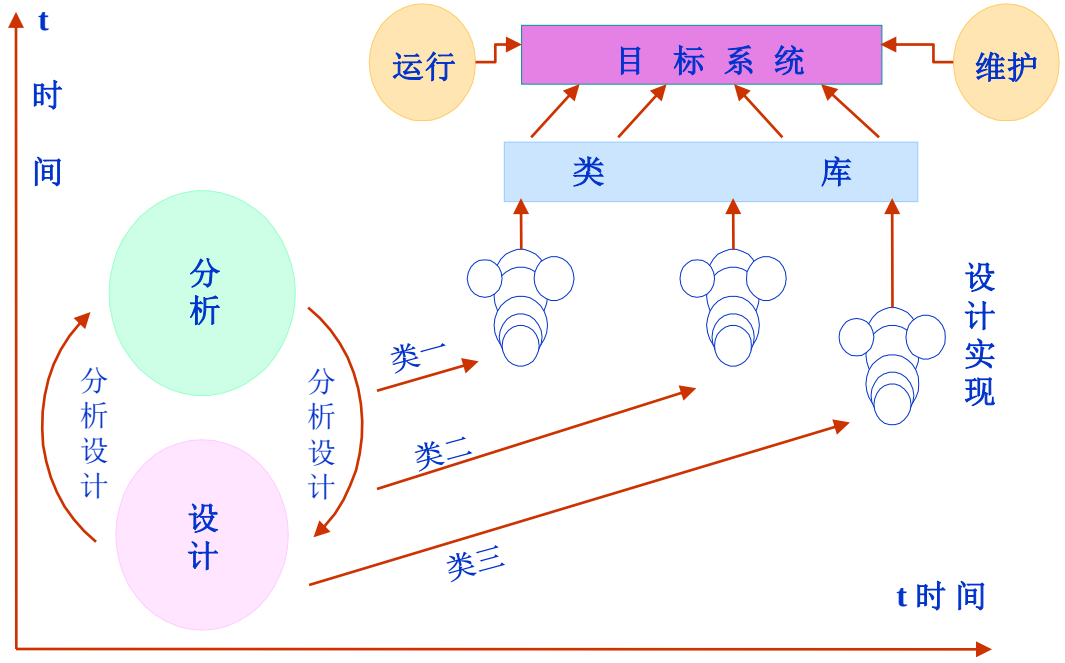
面向对象系统开发

- 将面向对象技术用于系统开发的全过程，包括分析、设计、编程、测试及集成等
- 从面向对象观点出发，以应用领域的问题对象为着眼点，用直观的方式描述客观世界的内部结构，将现实世界的空间模型平滑而自然地过渡到面向对象的系统模型，使系统开发过程与人们认识客观世界的过程保持最大限度的一致
- 整个过程可从宏观和微观两个层面去理解

10.1 面向对象系统开发方法论

面向对象系统开发方法宏观层面

- 从宏观层面看，面向对象的系统开发方法包括分析、设计、实现以及运行与维护等四大阶段，遵循反复累增的生命周期
- 这种反复累增的生命周期与传统的不同，既非严格的自顶向下，也非严格地自底向上
- 反复是指分析、设计与实现各阶段不是顺序完成的，而是经过多次迭代完成，每一次迭代都要以前次迭代结果为基础，进行相应的分析、设计与实现
- 累增是指在每一次迭代过程中，分析、设计与实现都会产生新的成果，系统功能结构逐步得到改进，最后达到用户要求



10.1 面向对象系统开发方法论

面向对象系统开发方法微观层面

- 每一宏观阶段中几乎都会涉及标识对象，确定对象的属性，定义对象的服务，以及确定对象间关系等处理步骤
- 分析阶段，通过详细调查问题领域，列举问题领域主要实体对象，初步梳理出这些实体对象的属性、行为及彼此间的关系
- 设计阶段,对分析阶段所得到的问题领域实体对象/类进行求精，进一步发掘新的问题领域对象，同时要面向目标系统的实现标示出控制对象/类及接口对象/类，在这一过程中，还要定义对象/类之间的关系；
- 实现阶段，一方面构成较高级抽象,例如，图像、按钮、对话框、控制框等低级类可构成一个较高级窗口类，另一方面,在现有类中发现共性，抽象出更高级的类

10.1 面向对象系统开发方法论

面向对象系统开发方法的内容与过程

- 面向对象的系统分析(OOA)，旨在了解问题域内该问题所涉及的对象和对象间的关系，建立问题模型
- 面向对象的系统设计(OOD)，它的任务是调整、完善和充实由OOA建立的模型
- 面向对象的系统实现(OOP)，它的任务是用面向对象的语言实现OOD提出的模型
- 运行及维护

OO方法与传统的生命周期法相似，但各阶段所解决的问题和采用的描述方法却有极大的区别

10.2 面向对象的系统分析-OOA

Object-Oriented Analysis（OOA）概念

- 面向对象的分析方法是利用面向对象的信息建模概念，如实体、关系、属性等，同时运用封装、继承、多态等机制来构造模拟现实系统的方法
- 传统的结构化设计方法的基本点是面向过程，系统被分解成若干个过程。而面向对象的方法是采用构造模型的观点，在系统的开发过程中，各个步骤的共同的目标是建造一个问题域的模型。在面向对象的设计中，初始元素是对象，然后将具有共同特征的对象归纳成类，组织类之间的等级关系，构造类库。在应用时，在类库中选择相应的类

OOA与结构化分析有较大的区别。OOA所强调的是在系统调查资料的基础上，针对OO方法所需要的素材进行的归类分析和整理，而不是对管理业务现状和方法的分析

10.2 面向对象的系统设计-OOA

OOA特征

- OOA在定义属性的同时，要识别实例连接。实例连接是一个实例与另一个实例的映射关系
- OOA在定义服务的同时要识别消息连接。当一个对象需要向另一对象发送消息时，它们之间就存在消息连接

The purpose of any analysis activity in the software life-cycle is to create a model of the system's functional requirements that is independent of implementation constraints.

10.2 面向对象的系统设计-OOA

OOA基本任务

- 找出并规定一组根据系统的各项要求而行动和相互作用的问题领域的对象，依据这些对象及其关系建立问题域模型

10.2 面向对象的系统分析-OOA

OOA模型层次

① 主题层

② 对象类层

③ 结构层

④ 属性层

⑤ 服务层

OOA活动构成

① 认定对象

② 认定结构

③ 认定主题

④ 定义属性和实例关联

⑤ 定义服务和消息关联

传统的分析产生一组过程性的文档，其着眼点是将系统看作一组功能。而面向对象的分析文档将问题看作一组相互作用的实体，并确定实体之间的关系

10.2 面向对象的系统分析-OOA

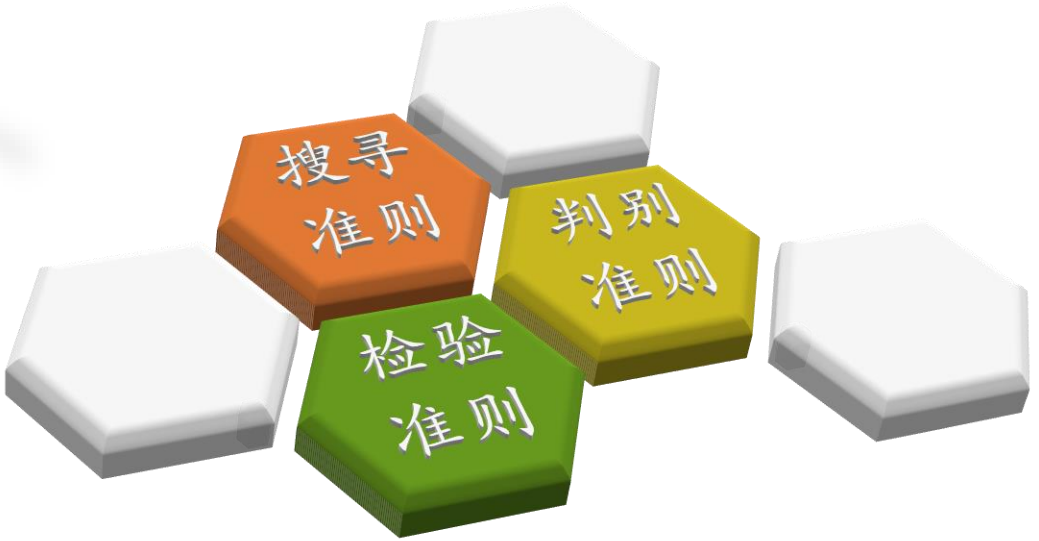
OOA应用的原则

1. 构造和分解相结合。构造是指由基本对象组装成复杂对象的过程。分解是对大粒度对象进行细化，从而完成系统模型细化的过程
2. 抽象与具体相结合。抽象是指强调事务的本质属性而忽略非本质细节。具体则是对必要的细节加以刻画的过程
3. 封装的原则。封装是指对象的各种独立外部特性与内部实现相分离，从而减少了程序间的相互依赖，有助于提高程序的可重用性
4. 继承的原则。继承是指直接获取父类已有的性质和特性，而不必重复定义。
5. 构造问题空间。构造问题空间的常用方法如下：
   - (1) 区分对象及其属性。如区分一棵树和树的大小和位置。
   - (2) 区分整体对象及其组成部分。如区分一棵树和树枝，这一构造过程称为构造分类结构。
   - (3)不同对象类的形成与区分。如所有树的类和所有石头的类的形成与区分。此过程称为组装结构

10.2 面向对象的系统设计-OOA

OOA基本步骤

- 问题域陈述
- **识别对象/类**
- 确定对象的属性
- 确定对象的服务
- 确定对象/类之间的关系



识别对象的准则

10.3 面向对象的系统设计-OOD

- Object-Oriented Design（OOD）
  - OOD阶段要解决的问题是如何把分析阶段确立出来的对象和类配置起来，实现系统功能，建立系统体系结构
  - 针对OOA给出的问题域模型，用面向对象方法设计出软件基础架构（概要设计）和完整的类结构（详细设计），以实现业务功能
  - 生成对象类的动、静态模型

10.3 面向对象的系统设计-OOD

OOD具体任务

- 对实体对象进行增、并、改，并识别接口对象和控制对象
- 确定实体对象、接口对象和控制对象之间的各种关系
- 完善对象类结构图，组织系统的体系结构

10.3 面向对象的系统设计-OOD

Object-Oriented Design（OOD）概念

- OOD是一种解决软件问题的设计范式（paradigm），一种抽象的范式
- 抽象可以分成很多层次，从非常概括的到非常特殊的都有，而对象可能处于任何一个抽象层次上；另外，彼此不同但又互有关联的对象可以共同构成抽象，即只要这些对象之间有相似性，就可以把它们当成同一类的对象来处理
- 使用OOD这种设计范式，我们可以用对象（object）来表现问题领域（problem domain）的实体，每个对象都有相应的状态和行为

10.3 面向对象的系统设计-OOD

OOD主要内容

- 识别接口对象和控制对象
  - 确认接口对象/类的准则
  - 控制对象的识别
  - 三类对象间的关系
- 识别控制对象
- 实体对象、接口对象和控制对象间的联系
- 系统结构设计
  - 问题领域子系统的设计
  - 人机交互子系统的设计
  - 外部接口子系统的设计
  - 数据管理子系统的设计
  - 任务管理子系统的设计
  - 基础对象子系统的设计

设计阶段的主要工作集中在交互图的开发上，但其最终结果都体现在设计类图中。因此说，设计类图是面向对象设计的核心，是呈现设计结果的重要模型

10.4 面向对象的系统实施

- 这一阶段主要是将OOD中得到的模型利用程序设计实现
- 主要工作内容
  - 编码语言的选择
  - 面向对象应用程序框架的构建
  - 面向对象应用程序编写
  - 构建应用软件平台
  - 调试
  - 试运行

10.4 面向对象的系统实施

主要工作

- OOA和OOD两阶段得到的对象及其关系最终都必须由程序语言、数据库等技术实现，但由于在设计阶段对此有所侧重考虑，故系统实现不会受具体语言的制约，因而本阶段占整个开发周期的比重较小
- 宜采用面向对象程序设计语言，一方面由于面向对象技术日趋成熟，支持这种技术的语言已成为程序设计语言的主流；另一方面，选用面向对象语言能够更容易、安全和有效地利用面向对象机制，更好地实现OOD阶段所选的模型

10.4 面向对象的系统实施

UML与系统实施

- 通常，UML建模软件（如Rational Rose）会提供由设计模型生成代码的功能，其中包括：
  - 前向工程。可从模型生成程序源代码和关系数据库中的表
  - 逆向工程。用反向生成器功能实现将程序源代码转换为UML模型的图
- 这就为反复修改、采用迭代式系统开发过程和实现业务流程优化创造了条件，从而提高系统的适应性和可维护性

10.5 统一建模语言UML

- Unified Modeling Language (UML)
  - 又称统一建模语言或标准建模语言，是始于1997年一个OMG标准，它是一个支持模型化和软件系统开发的图形化语言，为软件开发的所有阶段提供模型化和可视化支持，包括由需求分析到规格，到构造和配置
  - UML为开发人员提供了标准的、易于理解的表达方式用于构建系统蓝图，便于不同的开发人员共享和交流工作结果

10.5 统一建模语言UML

- Unified Modeling Language (UML)
  - UML提供一套相互组合的图表元素，支持以图形方式对系统需求、功能、结构等内容进行建模，描述系统组成结构、功能结构及实现细节，为开发者或开发工具使用这些图形符号和文本语法为系统建模提供了标准
  - UML从考虑系统的不同角度出发，定义了用例图、类图、对象图、状态图、活动图、序列图、协作图、构件图、部署图等9种图。这些图从不同的侧面对系统进行描述。系统模型将这些不同的侧面综合成一致的整体，便于系统的分析和构造
  - 尽管UML和其它开发工具还会设计出许多派生的视图，但上述这些图和其它辅助性的文档是软件开发人员所见的最基本的构造

10.5 统一建模语言UML

- UML规范
  - UML规范用来描述建模的概念有：类（对象的）、对象、关联、职责、行为、接口、用例、包、顺序、协作，以及状态

10.5 统一建模语言UML

类图

- 类图描述类和类之间的静态关系，如关联、聚类、组成和继承等关系
- 类图不仅显示了信息的结构，还描述了系统的行为
- 类图是定义其它图的基础
- 矩形方框代表类的图标，分三个区域
  - 最上面的区域标识类名
  - 中间区域是类的属性
  - 最下面区域里列的是类的操作

类图示例

10.5 统一建模语言UML

对象图

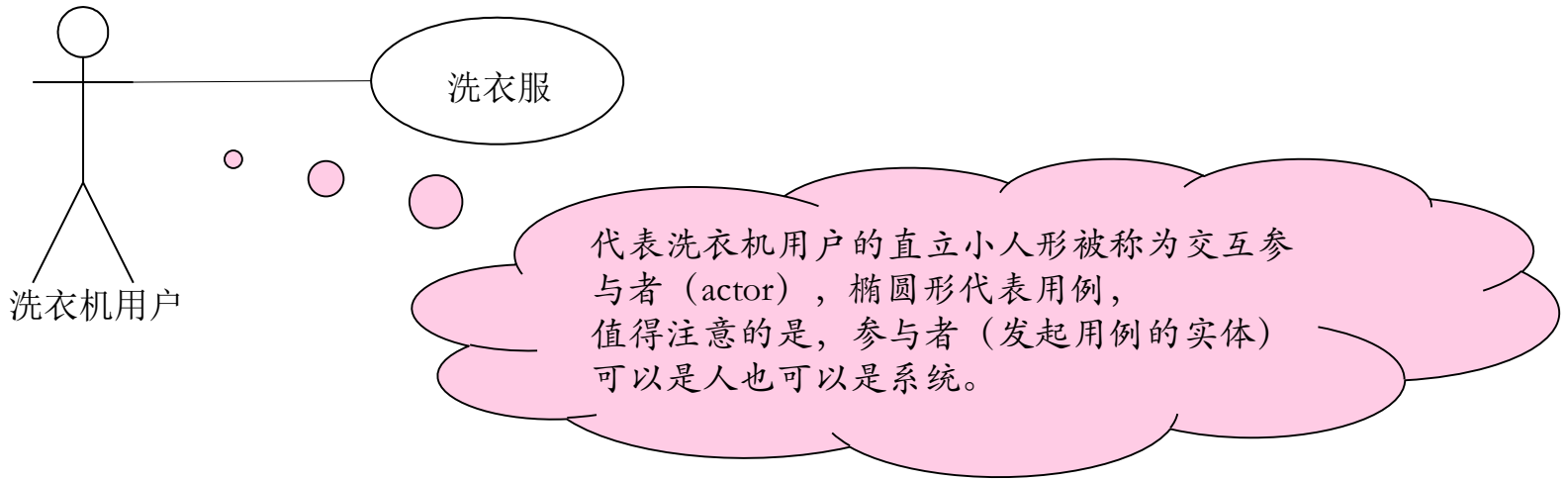
- 对象(object)是类的实例，具有具体属性值和行为
- 对象图常用于表示复杂类图的一个实例，对象之间的链(Link)是类之间的关联的实例
- 与类的图形表示相似，对象的图标也是个矩形，只是对象名下面要带下划线。具体实例的名字位于冒号的左边而该实例所属的类名位于冒号的右边

10.5 统一建模语言UML

- 用例图（use case diagram）
  - 用例是从用户的观点对系统行为或系统使用场景的描述。可以帮助系统开发人员从用户的观察角度收集可靠的系统需求
  - 一个用例是用户与计算机之间的一次典型交互作用。这对于建立人机交互的信息系统（而非计算机设备使用的）尤为重要

10.5 统一建模语言UML

- 用例图（use case diagram）
  - 下图说明了如何通过用例图来描述使用一台洗衣机洗衣服



10.5 统一建模语言UML

用例图

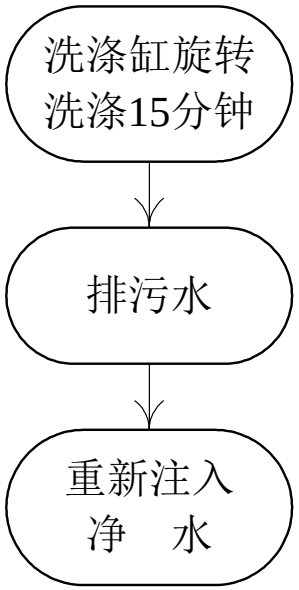
- 需求分析阶段的用例模型是系统开发者和用户反复讨论的结果，应能够充分表达开发者和用户共同认可的需求内容。其特点包括：
  - 首先，用例模型可以描述待开发系统的功能需求
  - 其次，用例模型将系统看作黑盒，从外部执行者的角度来看待系统
  - 第三，用例模型是需求分析之后各阶段开发工作的主要驱动因素，是验证和检测目标系统的依据

10.5 统一建模语言UML

- 活动图
  - 用途
    - 既可用来描述操作（类的方法）的行为
    - 也可以描述用例和对象内部的工作过程
  - 依据对象状态的变化来捕获动作（将要执行的工作或活动）与动作的结果

10.5 统一建模语言UML

- 活动图
  - 活动图中，一个活动结束后将立即进入下一个活动
  - 用例和对象行为的各个活动之间通常也具有时间顺序

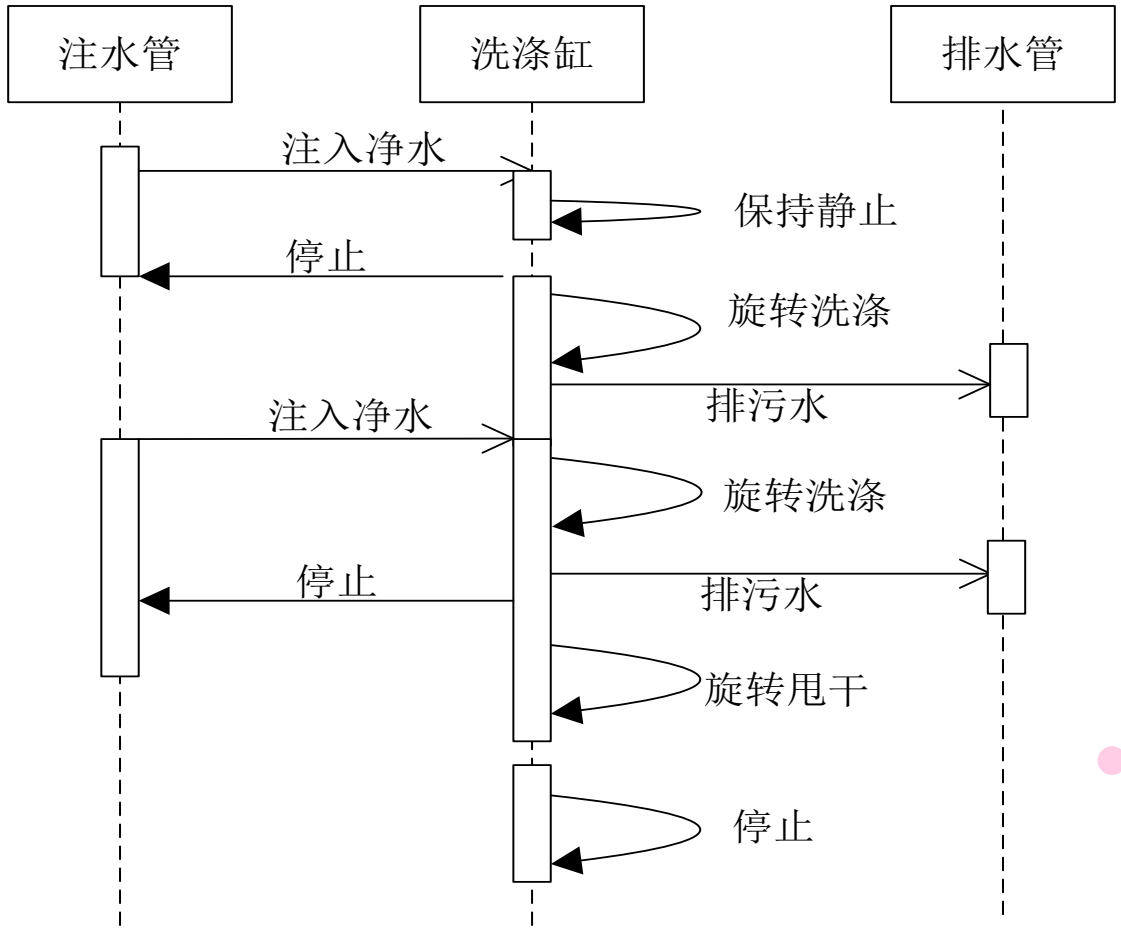


活动图图示

10.5 统一建模语言UML

- 顺序图
  - 表达对象之间的基于时间的动态交互关系，着重体现对象间消息传递的时间顺序
  - 顺序图存在两个轴：
    - 水平轴表示不同的对象
    - 垂直轴表示时间

10.5 统一建模语言UML



顺序图图示

图中的对象用带有垂直虚线的矩形框表示，标有对象名和类名。

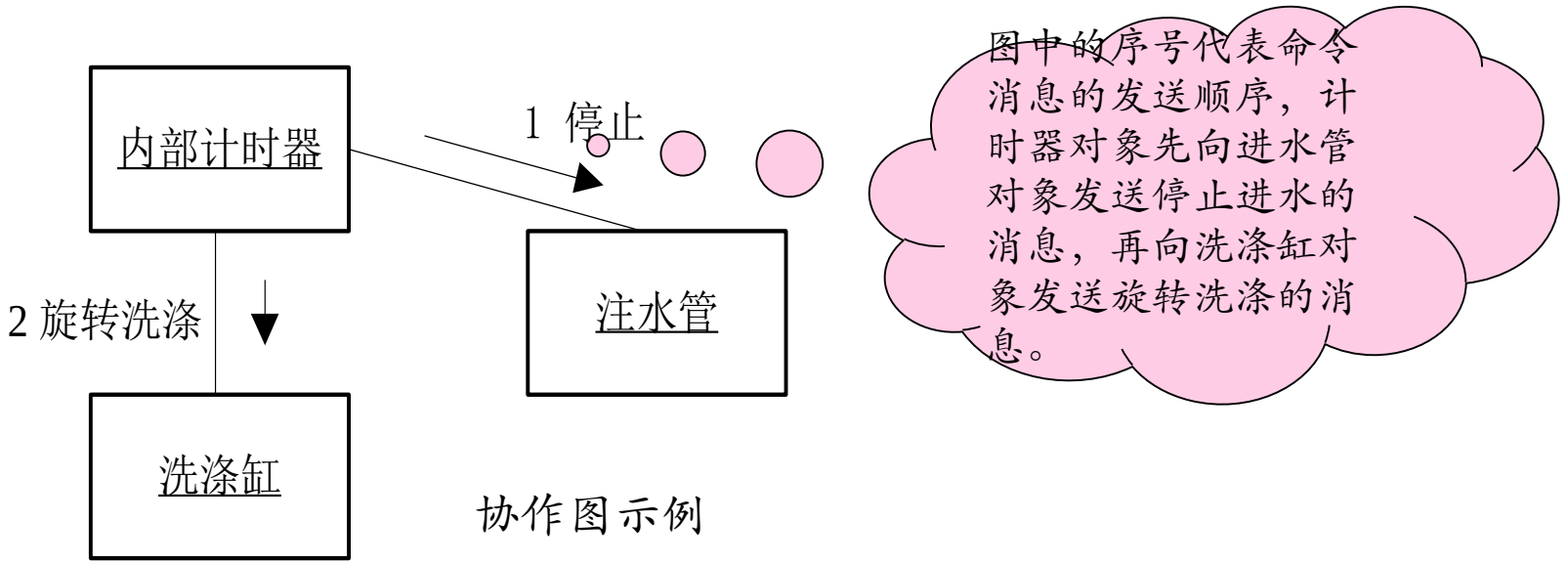
垂直虚线是对象的生命线，表示某段时间内对象处在活动状态。

对象间的通信则通过对象生命线间消息来表示。

10.5 统一建模语言UML

协作图

- 表达系统中相互合作的对象为完成目标之间的交互关系和链接关系
- 与顺序图着重体现交互的时间顺序不同，协作图强调交互对象间的静态链接关系

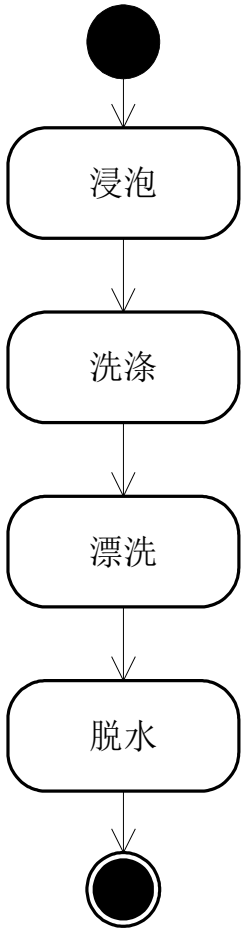


协作图示例

10.5 统一建模语言UML

状态图

- 表述在任何给定的时刻，一个对象所处的某一特定状态
  - 比如，电梯可以处于上升、停止或下降状态。洗衣机可以处于浸泡、洗涤、漂洗、脱水或关机等状态
- 状态图包括一系列的状态以及状态之间的转移



状态图图示

图中最顶端的符号代表起始状态，而底端的符号表示终止状态。

10.5 统一建模语言UML

- 组件图用于明确系统各部分的功能，例如在图书管理系统中可以包括“借／还书处理”、“信息查询”等组件
- 配置图则用于显示信息系统的物理体系结构，可以描述计算机和设备，展示其连接以及驻留在每台机器中的软件

The End