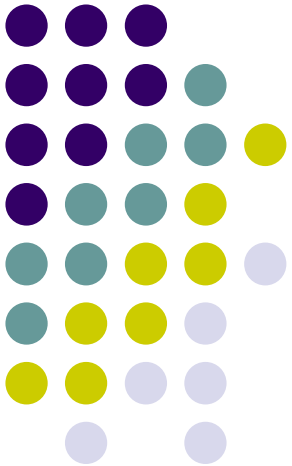
软件系统分析与设计

System Analysis & Design

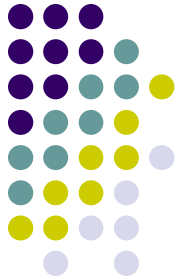
M210007B



目录

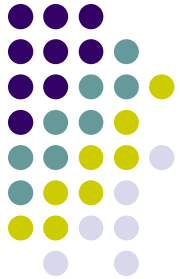
- 用例图定义
- 参与者
- 用例
- 用例图的应用

用例图



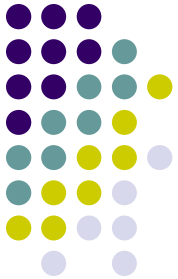
- 用例模型描述的是系统外部的参与者所理解的系统功能。
- 用例模型用于需求分析阶段，它的建立是系统开发者和最终用户反复讨论的结果，也是开发者和用户对需求规格定义达成的共识。

用例图



- 用例模型
  - 描述了待开发系统的功能需求
  - 将系统看作黑盒，从外部参与者的角度来理解系统
  - 驱动了需求分析之后各阶段的开发工作，用例不仅在开发过程中保证了系统所有功能的实现，还被用于验证和检测所开发的系统是否满足系统需求，从而影响到开发工作的各个阶段和UML的各个模型。

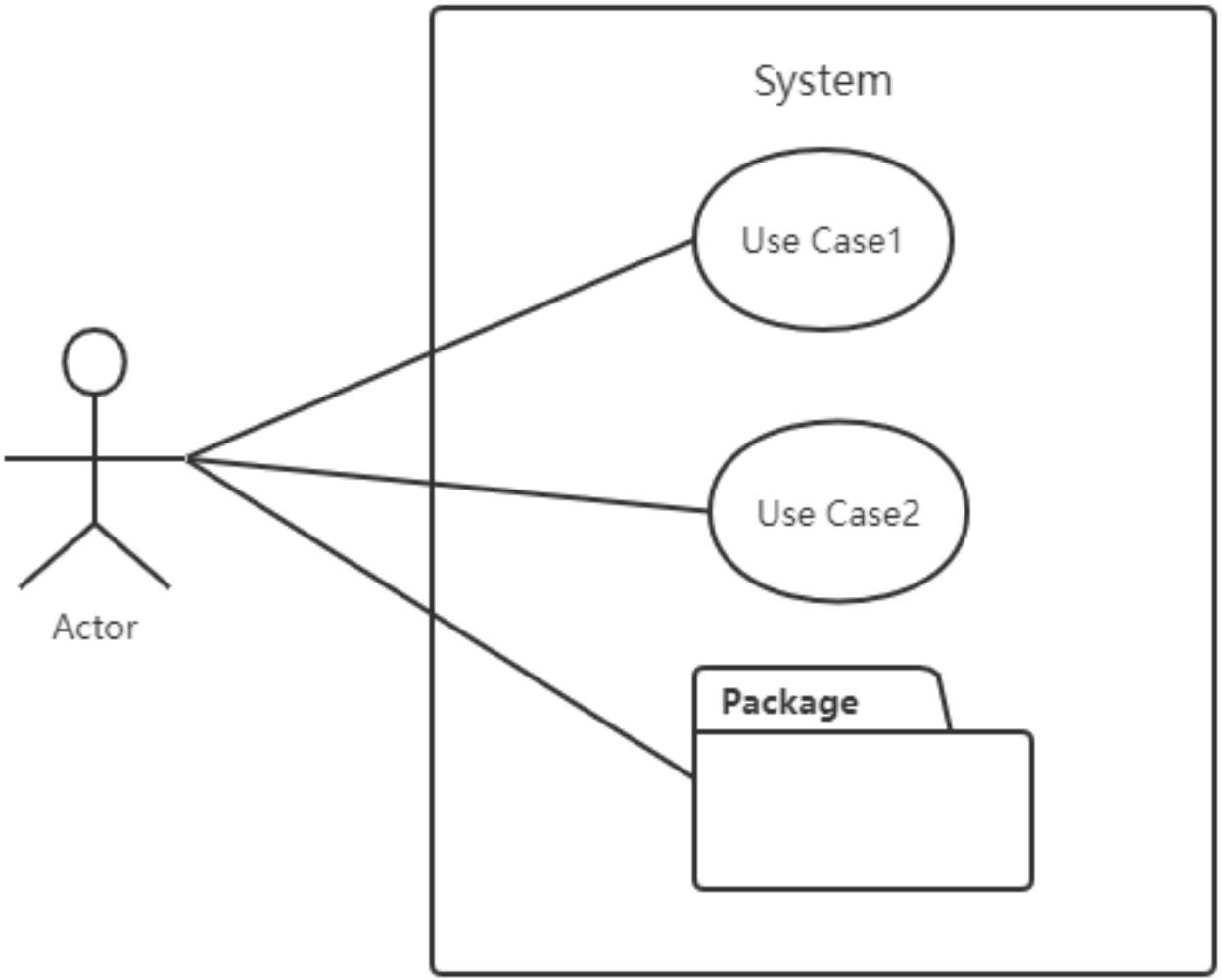
用例图



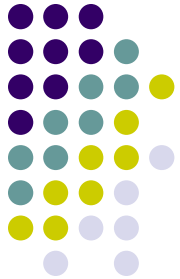
- 用例图的3种建模元素
  - 用例(Use Case)
  - 参与者(Actor)
  - 依赖关系、类属关系和关联关系。

  用例图描述了用例、参与者以及它们之间的关系。

用例图



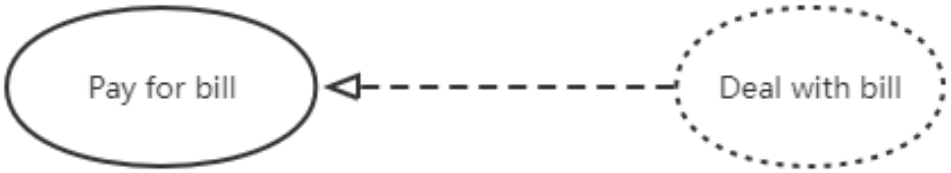
用例图



- 参与者和用例之间存在的关联关系通常被称为通信关联，因为它代表着参与者和用例之间的通信。
- 这个关联可以是双向导航（从参与者到用例，并从用例到参与者），也可以是单向导航（从参与者到用例，或从用例到参与者）。
- 导航的方向表明了是参与者发起了和用例的通信，还是用例发起了和参与者的通信。

用例图

- 在UML中用来实现用例的元素是协作（Collaboration），协作是实现用例行为的类和其他元素的总称。
- 如图所示，可以用协作“Deal with bill”（处理账单）来实现用例“Pay for bill”（付账单）。通常，每个给定的用例都会由一个相应的协作来实现，所以大多数情况下不必显式地为这种关系建模。



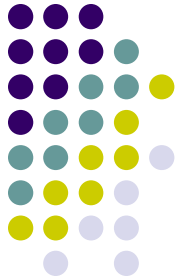
参与者

- 参与者（Actor）代表了与系统接口的事物或人，它是具有某一种特定功能的角色。因此，参与者是虚拟的概念，它可以是人，也可以是外部系统或设备。同一个人可能对应着多个参与者，因为一个人可能扮演了多个角色。参与者不是系统的一部分，它们处于系统的外部。

参与者

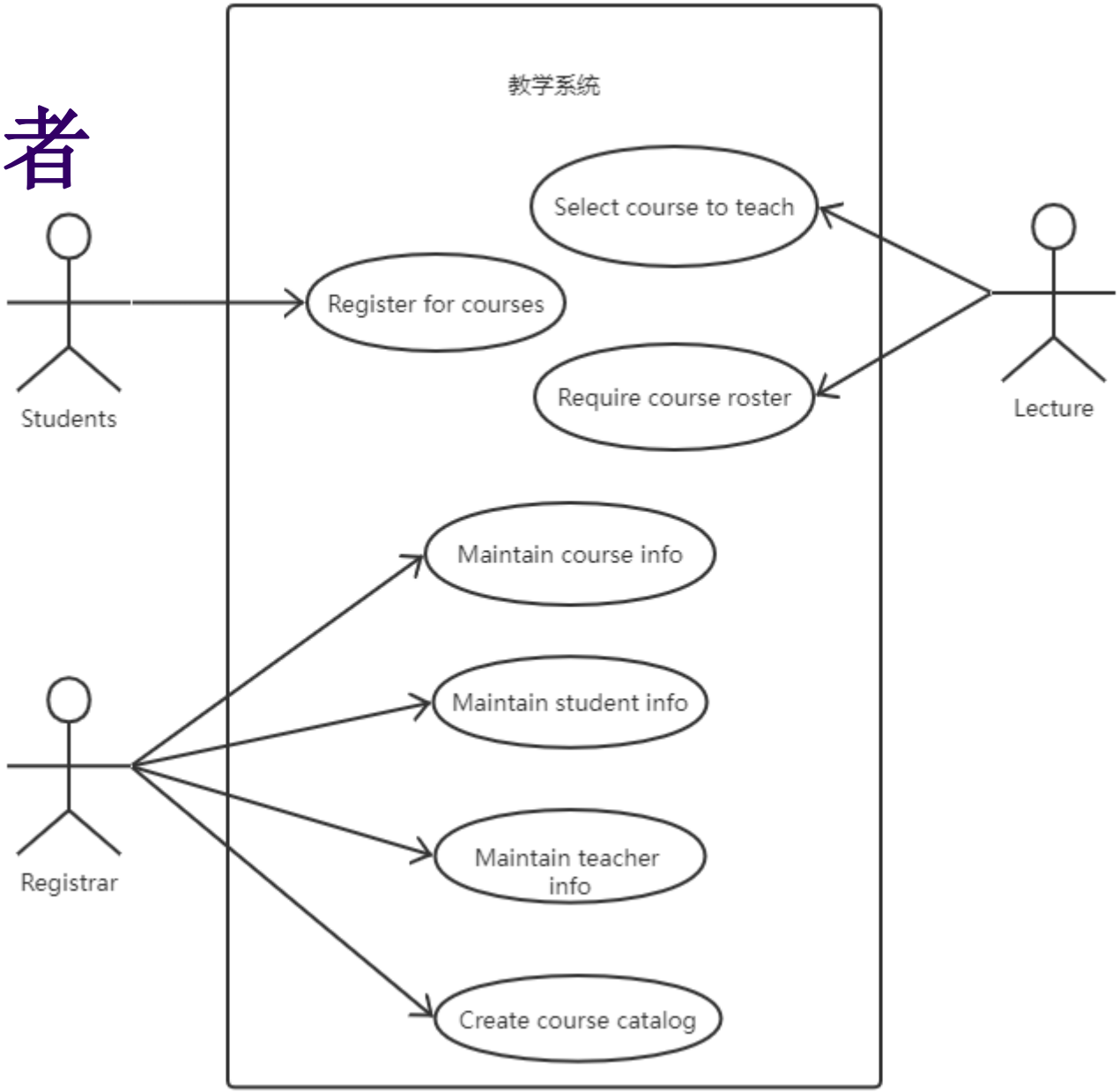
- 如何识别参与者？
  - 可以通过回答一系列问题
    - 谁是系统的主要用户？
    - 谁向系统提供信息？
    - 谁支持、维护系统？
    - 谁从系统获得信息？
    - 谁从系统删除信息？
    - 谁管理系统？
    - 系统需要与其他哪些系统交互（包含其他计算机系统和其他应用程序）？
    - 系统需要操纵哪些硬件？
    - 在预设的时间内，有事情自动发生吗？
    - 系统从哪里获得信息？
    - 谁对系统的特定需求感兴趣？
    - 几个人在扮演同样的角色吗？
    - 一个人扮演几个不同的角色吗？
    - 系统使用外部资源吗？
    - 系统要用在什么地方？

参与者



- 识别参与者需要注意：
  - 参与者代表角色。
    - 当建立用例模型时，参与者是用来模拟角色的，而不是用来模拟物理的、现实世界的人、组织或系统本身。
  - 角色不是对职位进行建模。

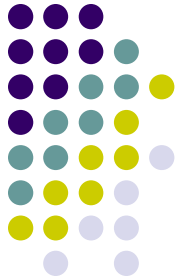
参与者


教学系统
Select course to teach
Register for courses
Require course roster
Students
Lecture
Maintain course info
Maintain student info
Maintain teacher info
Registrar
Create course catalog

用例

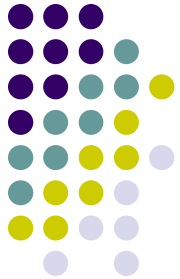
- 用例（Use Case）是对系统行为的动态描述
  - 可以增进系统设计人员、开发人员与用户的沟通，正确地理解系统需求；
  - 还可以划分系统与外部实体的界限。
- 用例是系统设计的起点，是类、对象、操作的来源，可以通过逻辑视图的设计，获得软件的静态结构。

用例



- 如何识别用例？
  - 可以通过以下问题帮助识别：
  - 每个参与者的任务是什么？
  - 有参与者要创建、存储、改变、删除或读取系统中的信息吗？
  - 什么用例会创建、存储、改变、删除或读取这个信息？
  - 参与者需要通知系统外部的突然变化吗？
  - 需要通知参与者系统中正在发生的事情吗？
  - 什么用例将支持和维护系统？
  - 所有的功能需求都能被用例实现吗？

用例



- 在描述用例事件流时，每个软件项目都应使用一个标准模板。下面给出一个目前应用最广泛的模板。

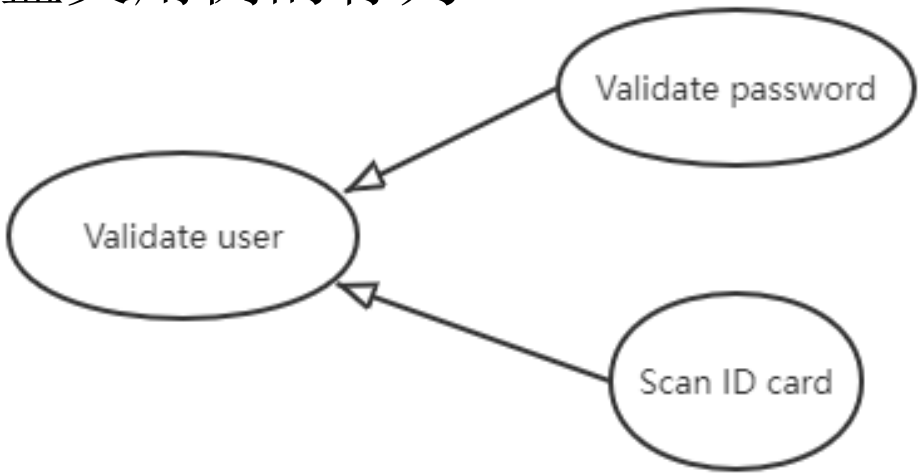
  X. 用例XX（用例名）的事件流
  X.1 前置条件（Pre-Conditions）
  X.2 后置条件（Post-Conditions）
  X.3 扩充点（Extension Points）
  X.4 事件流
  X.4.1 基流（Basic Flow）
  X.4.2 分支流（Subflows）（可选）
  X.4.3 替代流（Alternative Flows）

用例

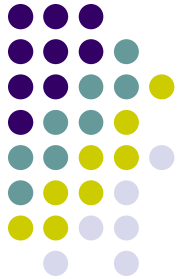
- 用例与脚本
  - 一个用例描述了一个序列集，而序列集中的每一个序列描述了一个流，这个流代表了用例的一个变种，每一个这样的序列就被称为一个脚本或场景（Scenario）。
  - 脚本是系统行为的一个特定动作序列。
  - 脚本与用例的关系就像实例与类的关系，即脚本是用例的一个实例。

用例

- 用例间的关系
  - 类属关系
    - 用例间的类属关系如同类间的类属关系。也就是说，子用例继承父用例的行为和含义，它也可以添加新行为或覆盖父用例的行为。



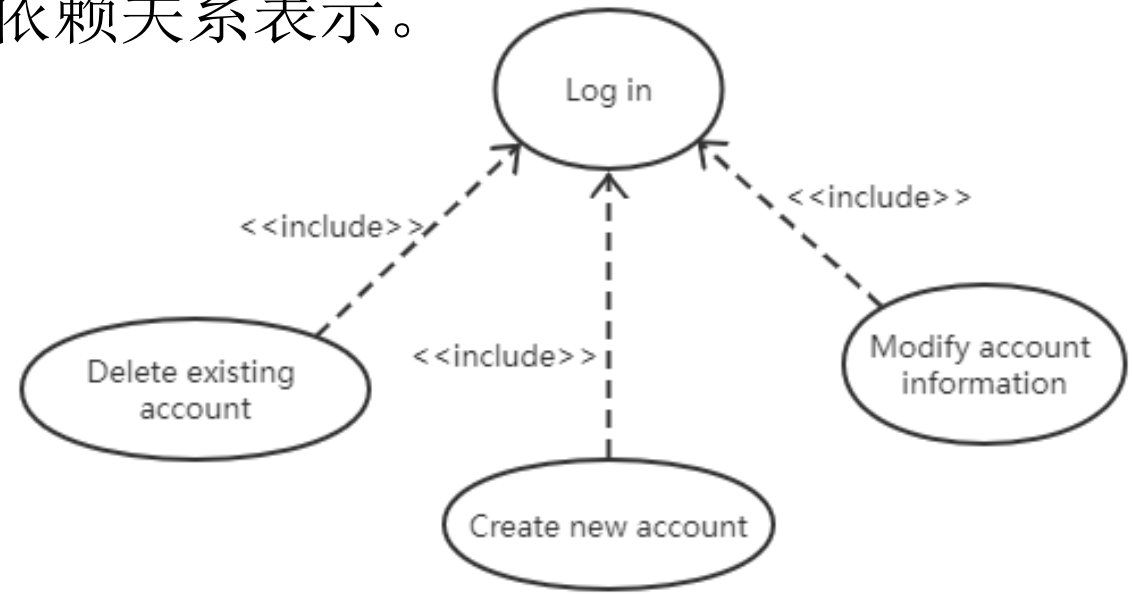
用例



- 用例间的关系
  - 包含关系
    - 多个用例可能具有一些相同的功能，通常将这些共享的功能放在一个单独的用例中，在这个新用例和其他需要使用其功能的用例之间创建包含（Include）关系。
    - 用例间的包含关系表示在基用例的指定位置，基用例显式地包含另一个用例的行为。被包含的用例是不能独立存在的，只是作为包含它的更大用例的一部分。

用例

- 用例间的关系（接上页）
  - 包含关系
    - 在UML中，Include关系可以用衍型为<<include>>的依赖关系表示。

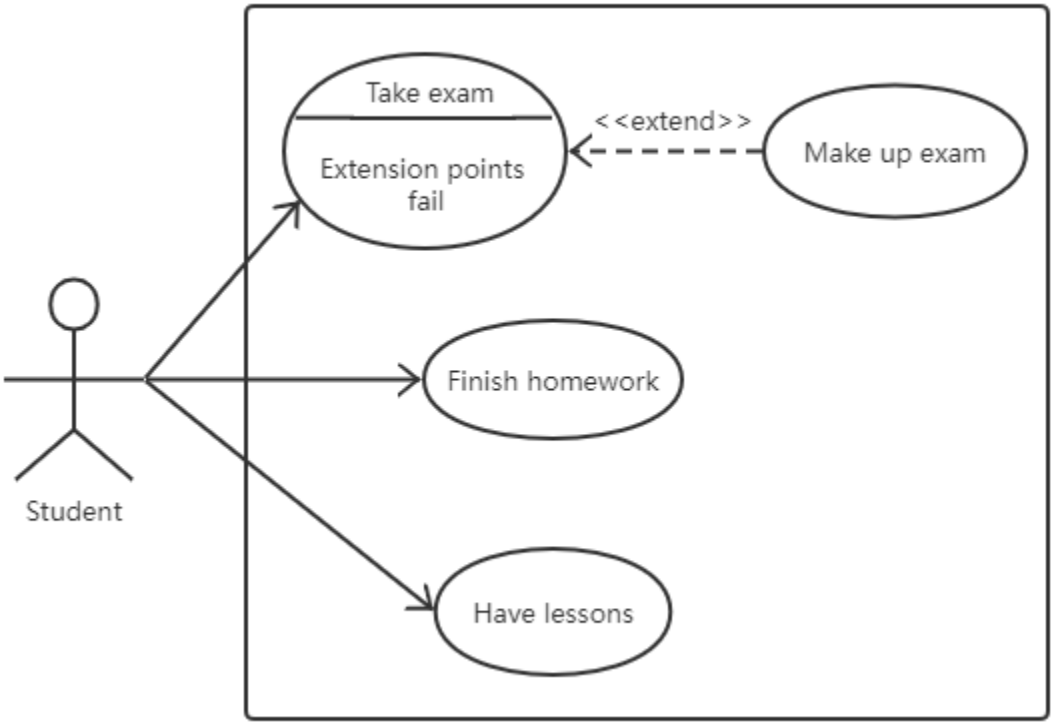


用例

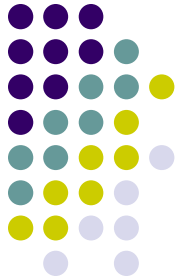
- 用例间的关系
  - 扩充关系
    - 扩充关系用来说明可选的、只在特定条件下运行的行为。根据参与者的选择，具有扩充关系的用例可以运行几个不同的流。
    - 用例间的扩充关系表示基用例在指定的扩充点隐式地包含另一个用例的行为。
    - 扩充关系被用来描述特定的用例部分，该用例部分被用户视为可选的系统行为，这样就将可选行为与义务行为区分开来。

用例

- 用例间的关系（接上页）
  - 扩充关系
    - 扩充关系用衍型为<<extend>>的依赖关系表示。

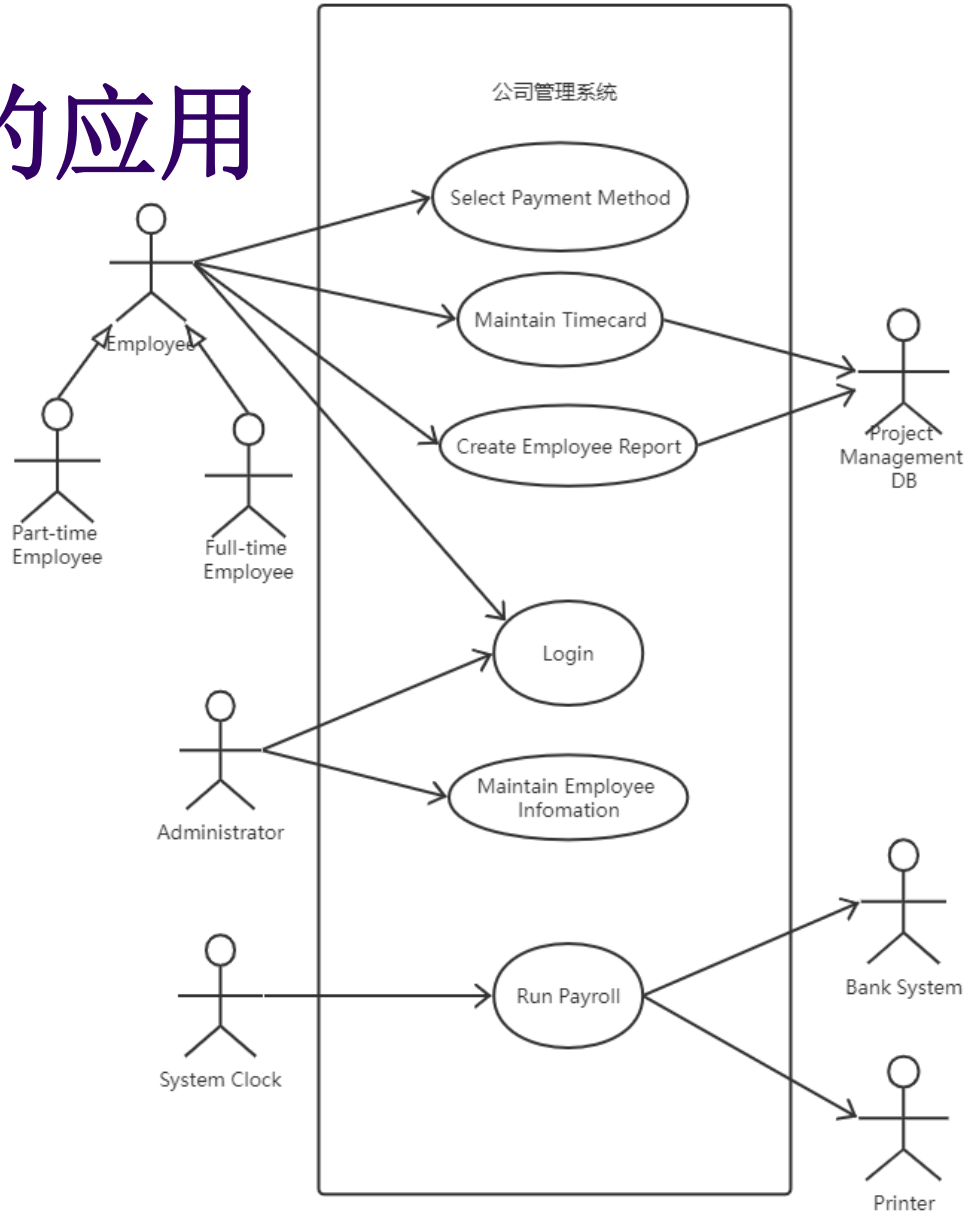


用例图的应用



- 用例图可以用来为系统的静态用例视建模。静态用例视体现系统的行为，即系统提供的外部可见的服务。
- 用例图可以被用来完成以下功能：
  - 为系统的上下文建模。
  - 为系统的需求建模。

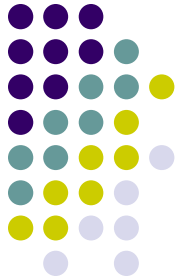
用例图的应用


公司管理系统
Select Payment Method
Maintain Timecard
Create Employee Report
Login
Maintain Employee Infomation
Run Payroll
Employee
Part-time Employee
Full-time Employee
Administrator
System Clock
Project Management DB
Bank System
Printer

用例图的应用

- 为系统的上下文建模。
  - 如上页图所示，用例图描述了一个公司管理系统的上下文，这个图强调了系统周围的参与者。
- 为系统的需求建模。
  - 如上页图所示，用例图可视化地描述了公司管理系统的功能需求，为最终用户、领域专家和开发人员之间的交流提供了途径。

小结



- 用例模型用于需求分析阶段，它描述了待开发系统的功能需求，并驱动了需求分析之后各阶段的开发工作。
- 用例图（Use Case Diagram）是UML中用来对系统的动态方面进行建模的7种图之一。用例图描述了用例、参与者以及它们之间的关系。

小结

- 本章介绍了用例图的语义和功能，描述了如何识别参与者、用例，如何使用事件流描述用例；还介绍了用例和脚本的关系，举例说明了用例间的类属关系、包含关系和扩充关系的语义、功能和应用；最后举例说明了如何使用用例图为系统的上下文以及系统的需求建模。