



UML用例建模

张天

软件工程组

ztluck@nju.edu.cn

2021年秋季



三大建模活动流



- 主流的生命周期模型基本都包含需求、分析和设计三个主要的建模过程
 - 需求描述：捕获用户需求，以文档为主
 - 需求分析：建立业务和计算机专家都理解的需求规格说明，以模型为主
 - 软件设计：面向计算机实现，建立业务、架构、数据库等的模型



需求描述



- 在需求描述，所开发的需求模型使用参与者和用例描述系统的功能性需求
 - 参与者：Actor（执行者）
 - 用例：Use Case（用况）
 - 注意，以上术语翻译方式很多，以英文为准
- 每个用例要开发一个叙述性描述
 - 这个才是最重要的，UML指南中称为事件流描述，常用五种方式：非形式化的结构化文字、形式化的结构化文字（带前后置条件）、状态图、活动图、伪代码



分析建模



- 在分析模型阶段，要开发系统的静态和动态模型
 - 静态模型定义问题域类之间的结构关系（类图）
 - 动态模型用于实现需求模型中的用例，以显示每个用例中参与的对象以及对象间是如何交互的（通信图、顺序图、状态图等）



设计建模



- 在设计建模中，一个重要的任务是设计系统的软件体系结构
 - 分析模型被映射到一个运行环境（如具体的框架体系或中间件平台等）
 - 其实，设计也可以分为概要和详细两个层次，但由于界限不是很明确，所以区分他们并不是很重要



需求和分析模型的差别



- 需求模型和分析模型
 - 三个建模阶段都会分别产生具体的模型：需求模型、分析模型和设计模型
 - 其中，需求模型和分析模型的差异性比较微妙（也可以说比较模糊）



用例图 (Use Case Diagram)



用例图描述系统外部的执行者与系统的用例之间的某种联系。

- **用例**：是指对系统提供的功能（或称系统的用途）的一种描述；
- **执行者**：是那些可能使用这些用例的人或外部系统；
- **联系**：用例和执行者之间的联系描述了“谁使用哪个用例”。



用例图 (Use Case Diagram)



- 用例图着重于从系统外部执行者的角度来描述系统需要提供哪些功能，并且指明了这些功能的执行者是谁；
- 用例图在UML方法中占有十分重要的地位，人们甚至称UML是一种用例图驱动的开发方法。



用例 (Use Case)



从本质上将，一个用例是用户与计算机之间为达到某个目的的一次典型交互作用：

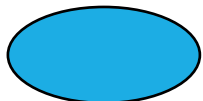
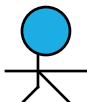


- 用例描述了用户提出的一些可见的需求；
- 用例可大可小；
- 用例对应一个具体的用户目标



用例图的符号



用例图中的图符：

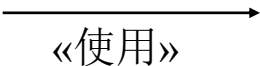
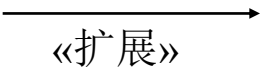


-  用例
-  执行者
-  系统：用于界定系统功能范围，描述该系统功能的用例都置于其中，而描述外部实体的执行者都置于其外。
-  关联：连接执行者和用例，表示执行者所代表的系统外部实体与该用例所描述的系统需求有关。



用例图的符号

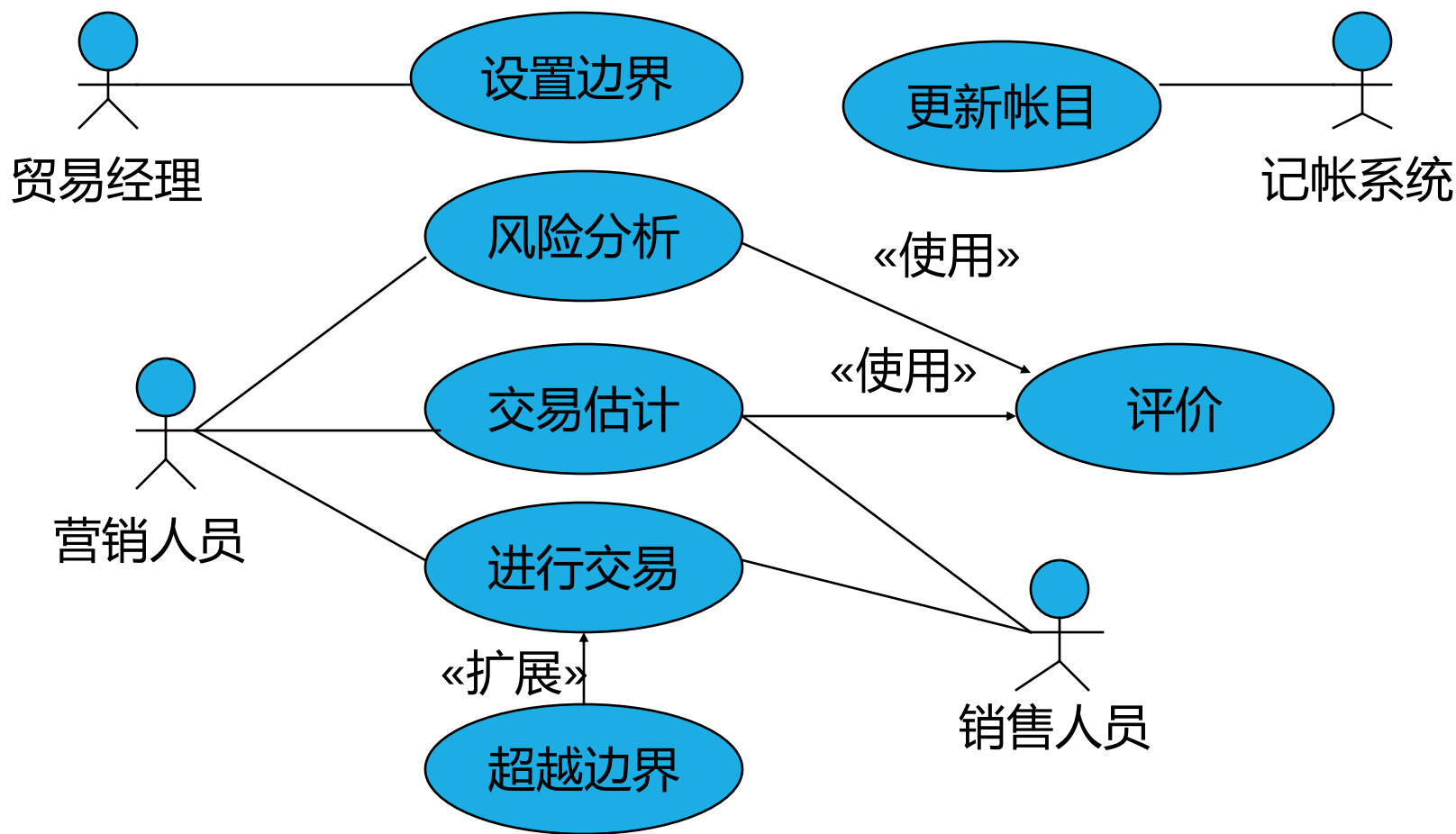


用例图中的图符：

-  使用：由用例A连向用例B，表示用例A中使用了用例B中的行为或功能。
-  扩展：由用例A连向用例B，表示用例B描述了一项基本需求，而用例A则描述了该基本需求的特殊情况。
-  注释体：对UML实体进行文字描述
-  注释连接：将注释体与要描述的实体连接，说明该注释体是针对该实体所进行的描述。



用例图 (Use Case Diagram)





文档化用例



- 用例模型中的每个用例都采用叙述性描述给出功能需求的具体内容
 - 常用五种描述方式：非形式化的结构化文字、形式化的结构化文字（带前后置条件）、状态图、活动图、伪代码
- 在最初的需求获取阶段，通常使用非形式化的结构化文字进行描述



用例描述



- 用例名称：用于标示用例的唯一名称
- 概述：对用例的简短描述，一般一两句话
- 依赖：可选，是否依赖其他用例（包含或扩展）
- 参与者：总有一个主要参与者来启动用例。可以有次要参与者。
- 前置条件：可选，该用例开始时必须满足的条件。
- 主序列描述：参与者和系统之间最经常的交互序列，具体形式是参与者的输入和系统响应。
- 可替换序列描述：主序列的可替换分支，可以有 multiple 条。
- 非功能需求：性能、安全等
- 后置条件：到达用例终点处所满足的条件。
- 未解决问题：相当于“备注”部分表明和该用例相关的问题，用于和用户进一步讨论。



基于用例的需求精化过程



- 1. 获取原始需求
- 2. 开发一个可以理解的需求
 - 2.1 识别参与者
 - 2.2 识别用例
 - 2.3 构建用例图
- 3 详细、完整地描述需求
 - 进行用例阐述
- 4 重构用例模型
 - 4.1 识别用例间的关系
 - 4.2 对用例进行组织和分包



用例图 (Use Case Diagram)



用例模型的获取：

- 获取执行者
- 获取用例



获取执行者的启发规则



获取执行者：

- 谁使用系统的主要功能（主要使用者）？
- 谁需要系统支持他们的日常工作？
- 谁来维护、管理系统使其能正常工作（辅助使用者）？
- 系统需要控制哪些硬件？
- 系统需要与其他哪些系统交互？
- 对系统产生的结果感兴趣的是哪些人？



获取用例的启发规则



获取用例：

- 执行者要求系统提供哪些功能？
- 执行者需要读、产生、删除、修改或存储系统中的信息有哪些类型？
- 必须提醒执行者的系统事件有哪些？
- 执行者必须提醒系统事件有哪些？怎样把这些事件表示成用例中的功能？



讨论



- 以研究生教务系统为例，使用用例驱动的需求分析方法，进行需求分析，并设计用例图



教务系统需求描述



- 研究生院/教务处建立一个本学期的课程目录
 - 一种课程可能有不同的时间、地点和听课对象的安排
 - 有不同的数据库来管理有关课程，学生和教师的不同信息
- 每个学生可选**4**门必修课和**2**门选修课，以便出现课程报满或取消的情况可以从备选中选择。
- 每门课程不超过**10**个学生。不少于**3**个学生。少于**3**个学生的课程将被取消。
- 一旦学生进行了选课注册，学校计费系统根据学生的注册和奖学金状态向学生发出交款通知。
- 学生可以利用这个系统在注册后的一段时间内修改选课计划，如增加或删除课程。
- 教授必须能够在线访问系统，以了解他教授的课程，他们也必须能够看见登记上课的学生情况。
- 系统的每一个用户通过他自己的口令验证来访问系统



执行者（Actor）的确定



执行者不是系统的一部分——它们代表任何必与系统交互的人或事物。

一个执行者有可能：

- 只向系统输入信息
- 只从系统获得信息
- 即向系统输入信息，又从系统获得信息



执行者（Actor）的确定



通常，执行者需要在问题陈述中挖掘，或者通过与用户或领域专家进行对话来获取。为了获取角色，可以在人，其他的软件，硬件设备，数据存储，或者网络目录中进行找寻。

下面一些问题对获取执行者是很有帮助的。

- 谁使用系统
- 谁安装系统
- 谁启动系统
- 谁维护系统



执行者（Actor）的确定



- 谁关闭系统
- 哪些其他的系统使用此系统
- 谁从此系统获取信息
- 谁为此系统提供信息
- 系统是否用到外部资源吗？
- 是否有人扮演多个角色？
- 是否有多人扮演同一个角色？
- 系统是否同遗留系统有交互？



执行者（Actor）的确定



- 学生要使用本系统注册所选课程
- 教授要使用本系统选择所教授的课程
- **教务员**要使用本系统创建课程目录和课表
- **教务员**要使用本系统维护所有相关课程，教授和学生的信息
- 计费系统能从本系统获取有关选课学生的信息

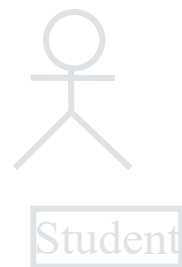


执行者（Actor）的确定



- 执行者是人或其它外部系统他/它将在系统开发和运行过程中和系统进行交互、对话。

- 学生
- 教师
- 帐单系统
- 注册系统





执行者（Actor）描述



每一个执行者都应该有一个简要的描述，用以指出该执行者在与系统交互式时扮演的角色

- **Student**—a person who is registered to take classes at the university
- **Professor**—a person who is certified to teach classes at the university
- **Registrar**—the person who is responsible for the maintenance of the Course Registration System
- **Billing System**—the external system responsible for student billing



用例（Use Case）的确定



- 用例描述了系统对外表现的特征和性能
 - 用例是系统的一个功能模块，是系统执行者和系统之间进行对话的一系列相关活动
- 如何寻找用例
 - 查看用户提交的文档
 - 询问系统的使用者
 - 对每个执行者进行分析，抽象他和系统之间可能的交互方法



用例（Use Case）的确定



一旦获取了角色，就可以对每个角色提出问题以获取用例。以下问题可供参考：

- 执行者要求系统提供哪些功能？
- 执行者需要读、产生、删除、修改或存储的信息有哪些类型？
- 必须提醒执行者的系统事件有哪些？或者执行者必须提醒系统的事件有哪些？怎样把这些事件表示成用例中的功能？
- 为了完整的描述用例，还需要知道执行者的某些典型功能能否被系统自动实现
- 系统需要何种输入输出？输入从何来？输出到何处？
- 当前系统（可能是手工系统而不是计算机系统）的主要问题有哪些？



用例（Use Case）的确定



- 一个好的用例是一个完整的 (complete from beginning to end)、关于系统某一项主要功能的描述
- 设计用例时的注意事项
 - 用例独立于实现
 - 用例是系统的高级视图，数目不易过多
 - 用例的命名使用业务术语，而不是技术术语



用例（Use Case）的确定



■ 讨论：

- 学生选课、把学生加入对应课程的选课名单、通知学生交费，3个还是1个用例？
- 教务员向系统添加、删除、修改课程信息，3个还是1个用例？



用例（Use Case）的确定



- 学生
 - 注册，浏览，增加，删除具体课程
- 教授
 - 索取课程花名册，选择执教的课程
- 教务员
 - 维护课程信息，维护学生信息，维护教授信息，产生课程目录
- 计费系统
 - 接受注册情况、计算注册费用，发出交款通知



用例（Use Case）的确定



- Register for courses
- Select courses to teach
- Request course roster
- Maintain course information
- Maintain professor information
- Maintain student information
- Create course catalog



用例（Use Case）的描述



■ Use Cases被描述在

○ 简短的描述

- Use Case 的高级描述，用以指出它的执行者以及它提供的功能。

○ 事件流程

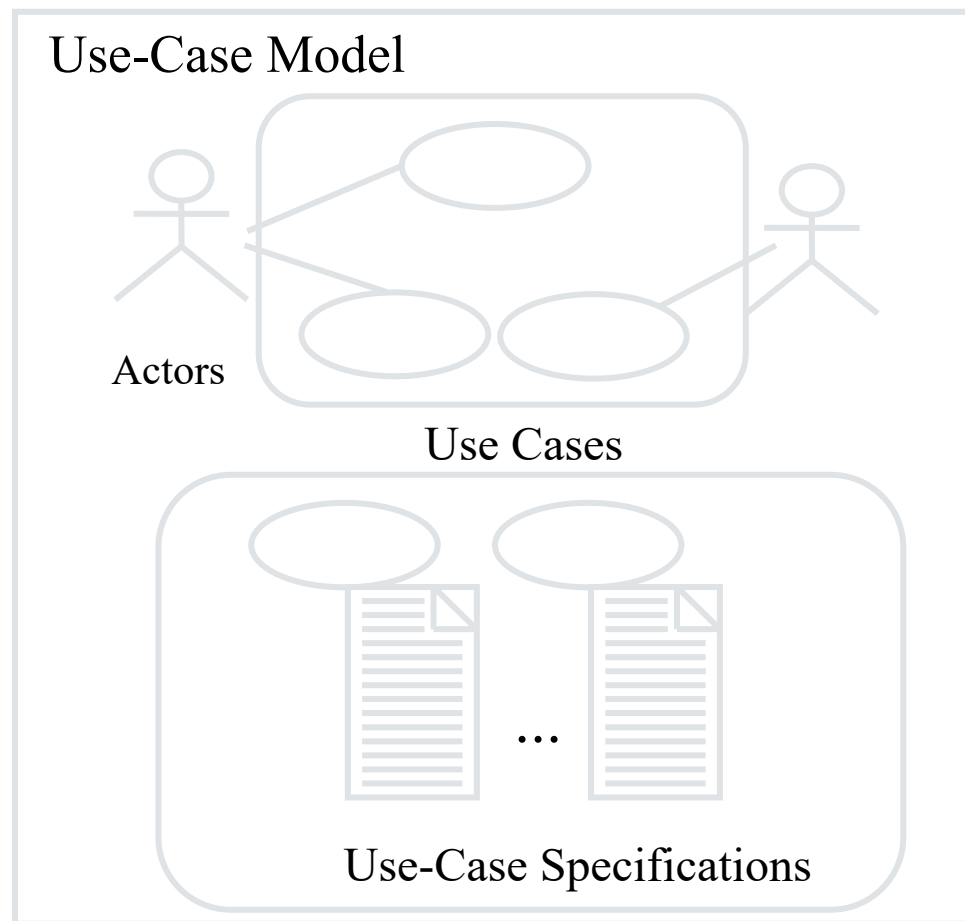
- 运行过程中的执行序列



用例（Use Case）的规约



- 名称
- 简要描述
- 事件流
- 关系
- 活动图
- 用例图
- 特殊要求
- 前提条件
- 后置条件
- 其它要素





用例（Use Case）的事件流



- 具体说明为了执行这样一个使用范例，系统需要经历哪些过程内容
- 应该描述系统做什么(what)，而不是描述怎么做(how)
- 应该用问题域(business domain)的术语描述，而不是用解域(implementation domain)的术语描述



用例（Use Case）的事件流



- 用例什么时候、如何启动/终止
- 用例与执行者有什么交互
- 用例需要什么数据
- 常规事件流
- 可选事件流
- 异常事件流



用例（Use Case）的事件流



- **X Flow of Events for the <name> Use Case**
- **X.1 Brief Description**
- **X.2 Flow of Events**
 - *X.2.1 Basic Flow*
 - *X.2.2 Alternative Flows*
- **X.3 Special Requirements**
- **X.4 Pre-Conditions**
- **X.5 Post-Conditions**
- **X.6 Extension Points**



注册课程的事件流



n 注册课程:

1. 学生输入ID号
2. 系统验证ID号有效, 并且提示学生选择当前的学期或未来的学期。
3. 学生输入学期, 系统提示学生选择活动:
 - q 创建一个课表, 创建的子事件流将被激发
 - q 浏览一个课表, 浏览的子事件流将被激发
 - q 改变一个课表, 改变的子事件流将被激发
 - | 删除一门课程, 删除的子事件流将被激发
 - | 添加一门课程, 添加的子事件流将被激发
4. 学生指示活动结束, 系统将打印学生课表, 通知学生注册结束。
 - o
5. 注册系统送一个帐单给计费系统处理。



注册课程的异常事件流



■ 另一种情况

- 在前页中2，如果是无效ID号输入，系统禁止访问。
- 在前页中3，如果在创建一个课表时，系统中课表已经存在，系统将提示进行其他选择（浏览或改变）。



注册课程的子事件流



■ 创建课表

- 学生输入四个主课号，和两门备选课程号。
学生提交选课请求。系统将：
 - 1: 检查选课条件满足
 - 2: 如果课程仍然没有选满，将学生加入到课程中。
- 另一种情况(异常事件流)
如果主课程不能提供，系统将选择备选课程



注册课程的子事件流



■ 浏览课表

- 学生请求一个给定学期的所有注册课程的信息，系统显示学生注册的所有课程信息，包括课程名，课程号，选课号，一周的天数，时间，地点，学分。



注册课程的子事件流



- 改变课表—删除一个课程
 - 学生指示哪个课程被删除，系统检查没有超过变更的最后期限，系统从课程中删除学生，系统通知学生请求被处理。
- 改变课表—增加一个课程
 - 学生指示哪个课程被增加，系统检查没有超过变更的最后期限，系统将：
 1. 验证增加的课程没有超过最大人数
 2. 检查选课条件满足
 3. 如果课程开放，将学生加入到课程中。

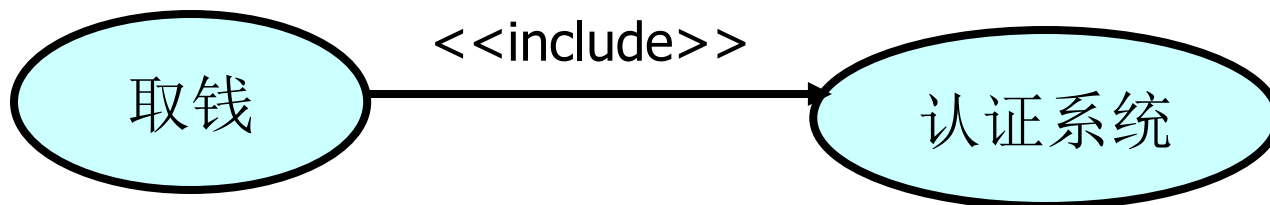


用例（Use Case）间的关系



■ Include

- «include»关系可用来描述某一个特性可能为一个或多个其它使用范例所共有。



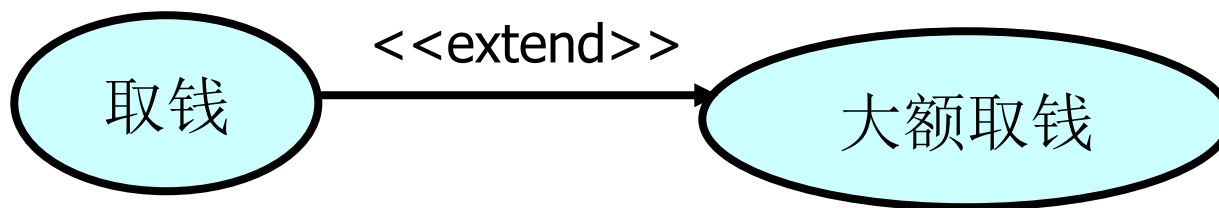


用例（Use Case）间的关系



■ Extend

- «extend»关系描述的是可能扩展（optional）的特性





用例图 (Use Case Diagram)



- 用例图描述各个执行者在各个用例中的参与情况，描述系统为用户所感知的外部视图。
- 用例图的功能：
 - 捕获系统用户需求
 - 描述系统边界
 - 指明系统外部行为
 - 指导系统开发者的功能开发
 - 系统建模的起点，指导所有的类图和交互图的设计
 - 产生测试用例，用户文档
 - 估计项目大小和进度。



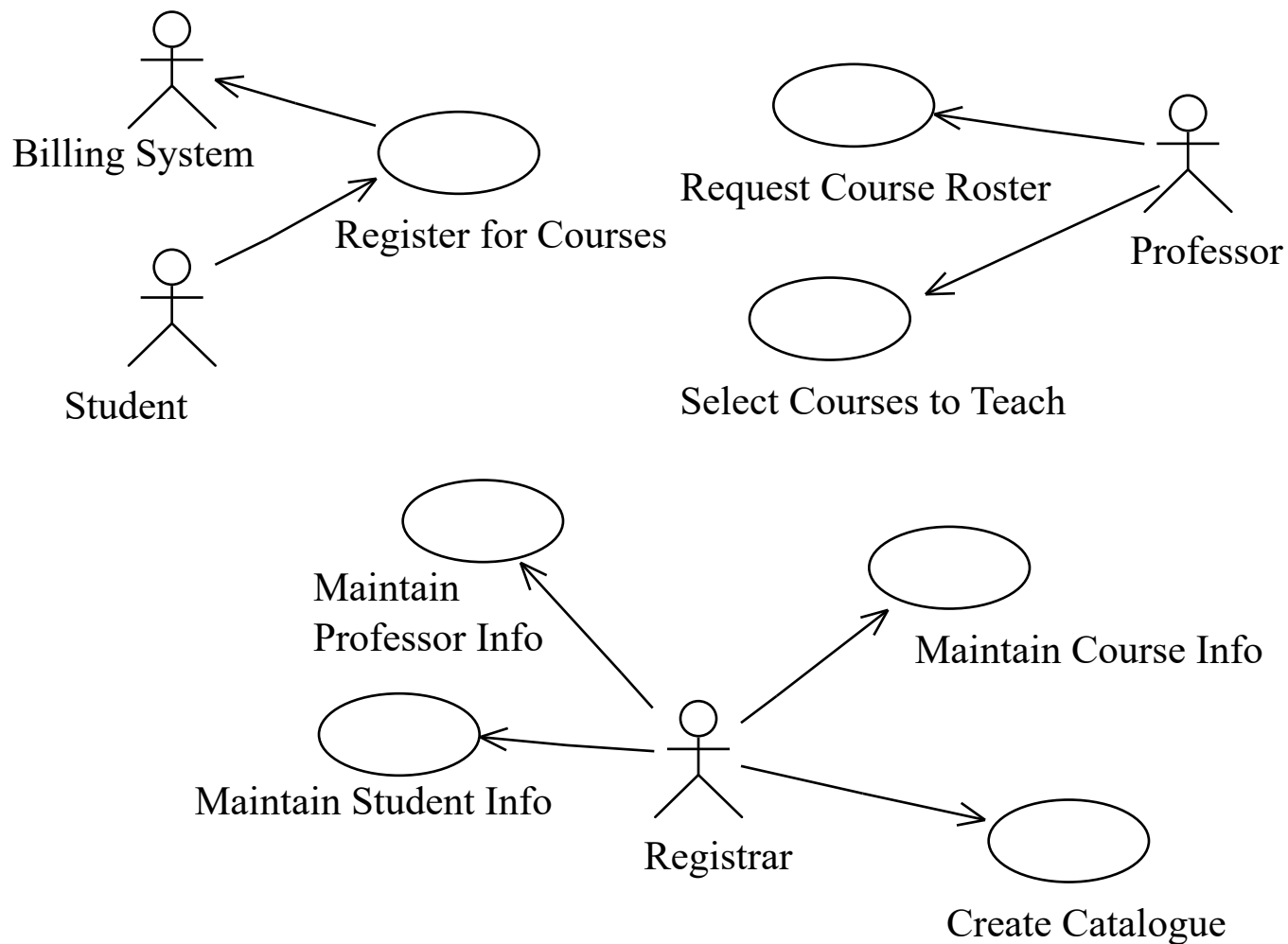
用例图 (Use Case Diagram)



- 主用例图 (Main Use Case diagram) 描述了系统的边界 (actors) 和系统的主要功能 (use cases)。
 -
- 通常每个系统都有一个主用例图。
- 其它用例图则根据需要添加。



用例图 (Use Case Diagram)





Reference



[Courtois85] Courtois, P. June 1985. On Time and Space Decomposition of Complex Structures. *Communications of the ACM* vol. 28(6), p. 596.

[Ibid] Ibid., p. 221.

[Gall86] Gall, J. 1986. Systemantics: How Systems Really Work and How They Fail. Second Edition. Ann Arbor, MI: The General Systemantics Press, p. 65.

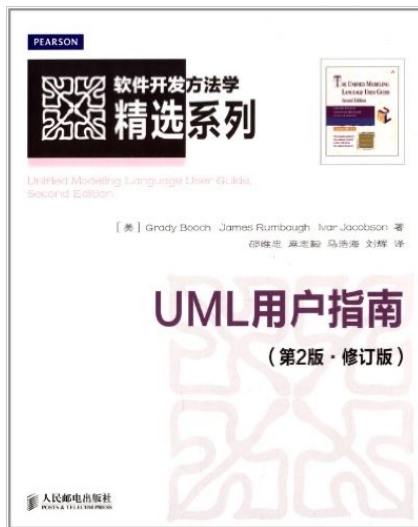
[Schach11] Stephen Schach, Object-Oriented and Classical Software Engineering 8th



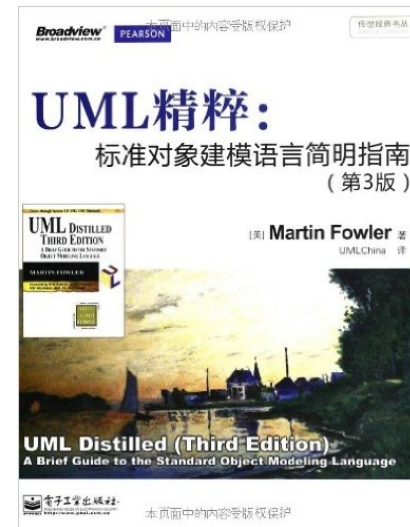
推荐书籍



软件建模与设计:UML、用例、模式和软件体系结构



UML用户指南(第2版)(修订版)



UML精粹:标准对象建模语言简明指南(第3版)