COMP130015.02

软件工程

8. 软件需求

复旦大学计算机科学技术学院

沈立炜

shenliwei@fudan.edu.cn

课堂讨论：什么是需求

对于一个大学选课系统开发项目而言，以下哪些陈述属于项目需求分析时通常应当考虑的（多选）？

1）系统的服务器端必须运行在Linux操作系统上

2）系统应当遵循MVC模式

3）系统在选课高峰期能保证页面响应时间小于5秒

4）系统应当使用多服务器部署和负载均衡技术

5）每位教师可以讲授多门课程，每门课程可以由多位教师共同讲授

6）当选课请求超出课程最大人数限制时，系统根据选课请求提交时间从前到后进行满足

7）系统应当提供课程选课学生列表的Excel下载功能并按学号进行排序

8）打印选课学生列表时使用冒泡排序对学生进行排序

课堂讨论：什么是需求

对于一个大学选课系统开发项目而言，以下哪些陈述属于项目需求分析时通常应当考虑的（多选）？

1）系统的服务器端必须运行在Linux操作系统上

2）系统应当遵循MVC模式

3）系统在选课高峰期能保证页面响应时间小于5秒

4）系统应当使用多服务器部署和负载均衡技术

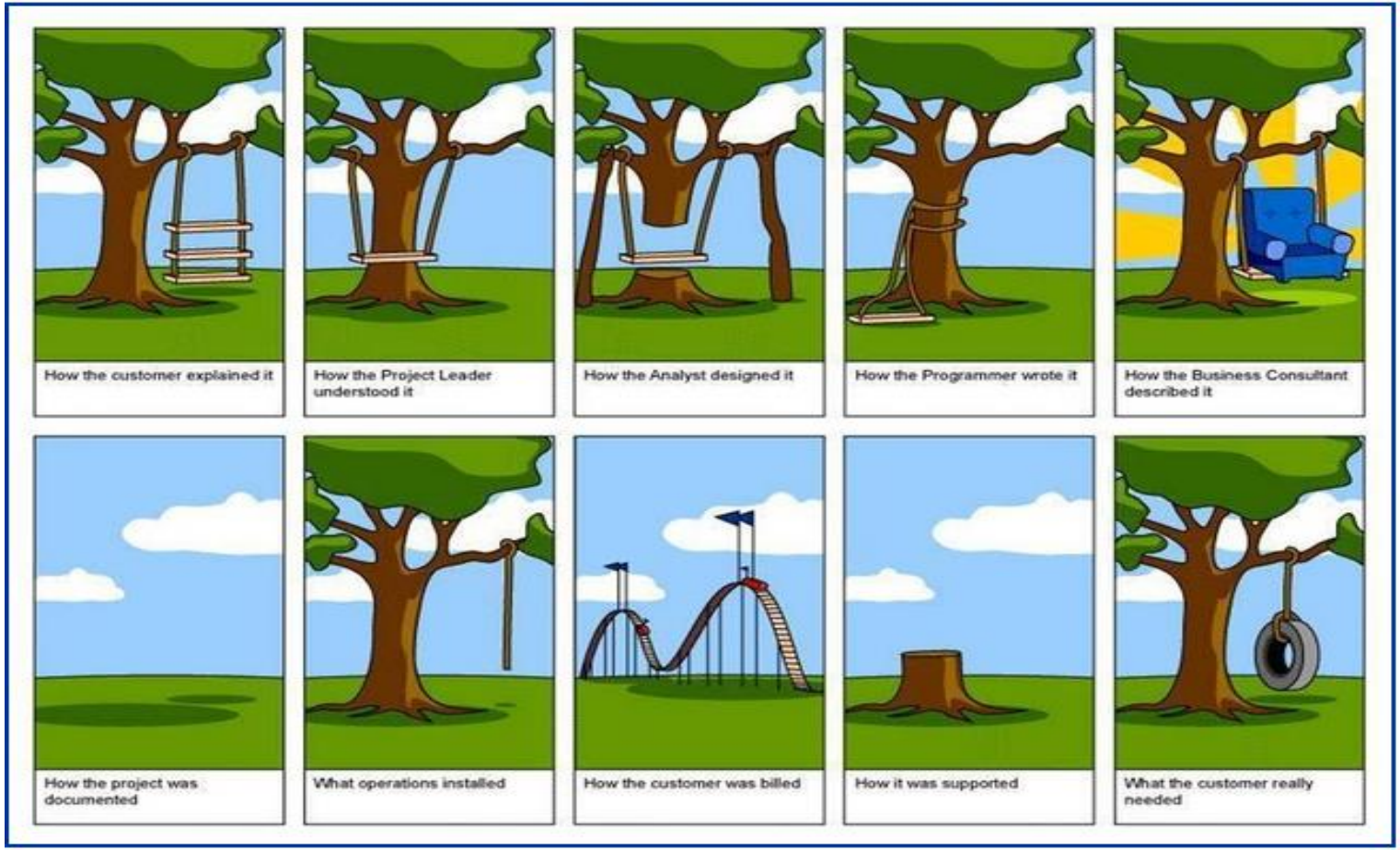
5）每位教师可以讲授多门课程，每门课程可以由多位教师共同讲授

6）当选课请求超出课程最大人数限制时，系统根据选课请求提交时间从前到后进行满足

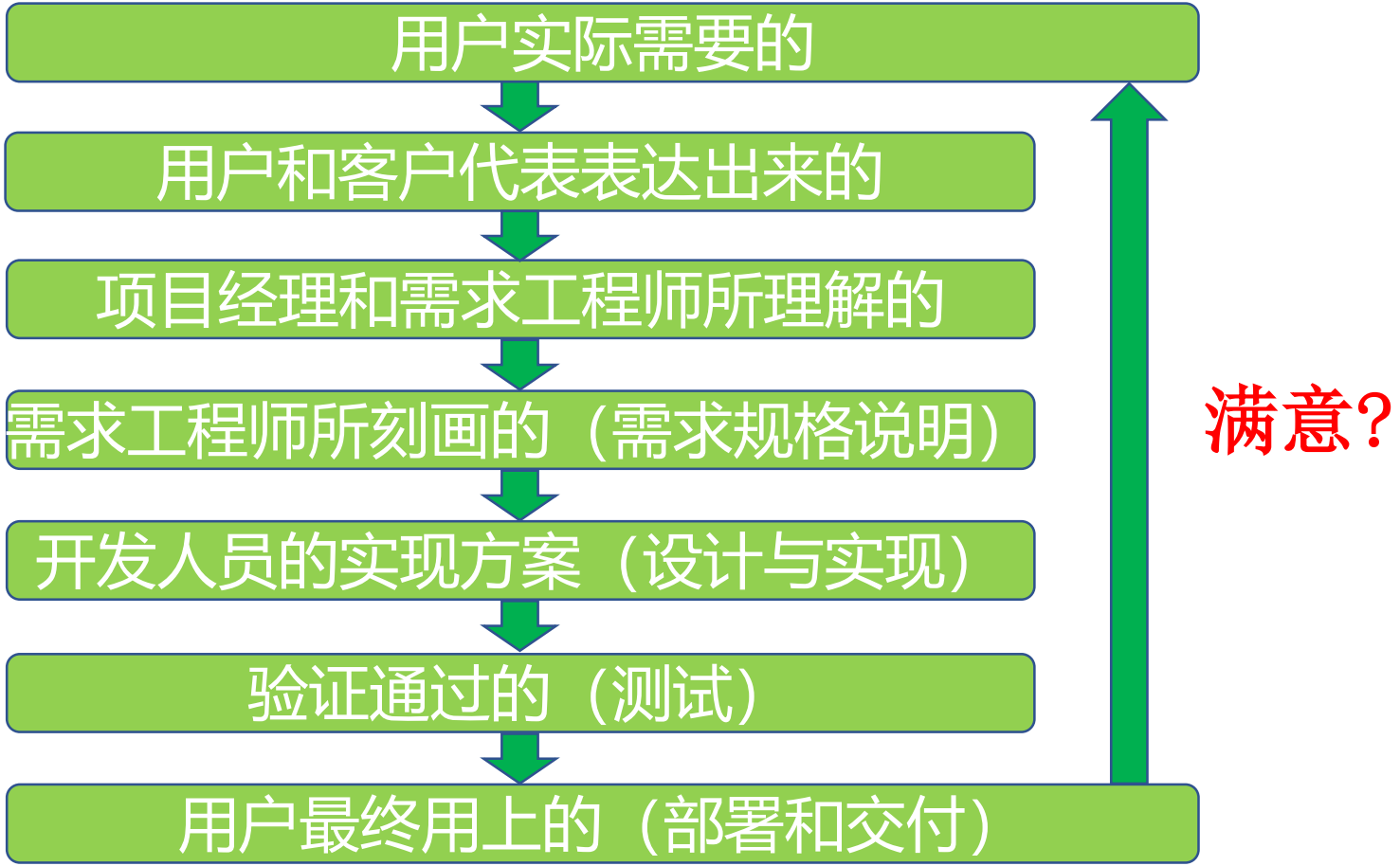
7）系统应当提供课程选课学生列表的Excel下载功能并按学号进行排序

8）打印选课学生列表时使用冒泡排序对学生进行排序

一幅关于需求的漫画


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

需求的传导过程


用户实际需要的
用户和客户代表表达出来的
项目经理和需求工程师所理解的
需求工程师所刻画的（需求规格说明）
开发人员的实现方案（设计与实现）
验证通过的（测试）
用户最终用上的（部署和交付）
满意？

关于软件需求的几个现实

- 软件需求的重要性经常被低估，而需求规约经常包含严重的缺陷
  - ✓ 遗漏、模糊、不完整、自相矛盾的需求
- 源代码中50%的失效都可以追溯到需求中存在的缺陷
- 需求缺陷发现得越晚，所需要的移除成本越高：对于需求质量的早期关注

需求问题案例：伦敦救护车调度系统

- 伦敦救护车服务计算机辅助调度系统
  - ✓ 对救护车通迅设备用户接口考虑不当，结果造成
    - 通过这些设备提供给救护人员的信息不正确或不完整
    - 救护人员感到难以操作这些设备
  - ✓ 未考虑救护人员跟随一辆与系统所调度的救护车不同的另一辆救护车去处理呼救请求的情况
  - ✓ 系统完全没有考虑无线电通信盲点的问题
- 导致一系列问题，系统运行陷入混乱
  - ✓ 处于未就绪状态的救护车被调度到呼救地点
  - ✓ 超出所需数量的救护车被调度到同一个呼叫现场
  - ✓ 导致：现场救护延迟，并直接危及人的生命

需求（Requirements）

- 需求的重要性
  - ✓ 反映客户、用户以及其他相关方的要求、诉求和愿望，同时也是检验最终交付的软件系统是否符合要求的重要依据
  - ✓ 开发团队中的架构师、程序员等开发人员理解开发要求、考虑技术方案以及做出技术决策的重要参考
- 如果需求自身存在缺陷，那么以此为基础开发的产品很可能会导致客户和用户不满意或难以在市场上被接受
- 国际标准ISO/IEC/IEEE 24765:2017的定义
  - ✓ 表达一种需要以及与之相关的约束和条件
  - ✓ 一个系统、系统组件、产品或服务为了满足某种约定、标准、规格说明或其他正式的强制性文件而必须满足的条件或拥有的能力

需求的来源

- 用户：使用系统的人或组织
- 客户：组织系统开发并之支付费用的人或组织
- 系统集成商：负责将目标软件系统与其他应用软件及软硬件和网络基础设施集成在一起的人或组织
- 软件工程师：参与软件设计、开发和测试等任务的工程师
- 销售人员：负责软件产品市场销售的人员
- 公众与监管机构：公众及代表公众利益的监管机构

需求来自于与待开发系统相关的多个方面

我们一般称其为涉众（stakeholder）

不同涉众的需求

- 不同涉众关心的问题不同，相应的需求也不同
  - ✓ 用户主要关心系统是否能满足自己的使用需求
  - ✓ 客户关心系统的整体满意度以及成本控制
  - ✓ 系统集成商关心是否容易和其他软硬件以及基础设施集成
  - ✓ 软件工程师关心软件系统需求的清晰性和实现难度
  - ✓ 销售人员关心软件系统是否能适应不同客户的需要
  - ✓ 公众与监管机构关心系统是否符合公众利益以及监管规定
- 应当尽可能全面地识别相关的涉众并充分考虑他们的期望和诉求
- 同时，不同涉众的要求可能存在不一致，从而导致潜在的需求冲突，因此需要发现潜在的需求冲突或不一致，并通过协商解决

需求的类型

功能性需求是指系统应向用户(人或其他系统)提供的功能

质量需求定义了整个系统或系统组件、服务或功能的质量特性，其中有些需求针对系统整体，有些则针对特定的服务、功能或组件

约束是一种限制了系统开发方式的组织或技术要求，包括文化约束(来源于系统用户的文化背景)、法律约束(来源于法律和标准)、组织约束、物理约束、项目约束等，此外按照作用对象还可以分为产品约束和过程约束

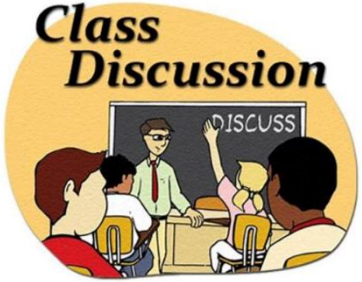
需求类型	子类型	需求示例
功能性需求	系统应提供的服务	能够按照关键字检索图书
	系统针对特定输入的响应	对于格式不正确的身份证号进行提示并请用户重新输入
	系统在特定情形下的行为	用户如果5分钟内没有操作，那么主界面自动进入锁定状态
	系统不应做什么	不允许尝试密码输入三次以上
质量需求	性能（performance）	联机刷卡应当在5秒内返回结果
	可靠性（reliability）	系统的整体可靠性要达到99.99%以上
	安全性（security）	系统应确保手机支付充值账户和密码不会被泄漏和盗用
	易用性（usability）	用户根据提示学会手机支付充值的时间不超过10分钟
约束	产品约束	软件系统要在已有的几台服务器上运行并使用Linux系统
	过程约束	软件系统应当在5个月内交付并严格遵循给定的过程规范

课堂讨论：房屋信息系统的需求

课堂讨论：这些需求中哪些是功能性需求、哪些是系统质量，哪些是约束?

R1：系统应该提供一个用户友好的界面
R2：系统能够生成包含所有被允许进入及被拒绝进入房屋的所有记录的月报表
R3：如果用户输入的PIN码（用户识别码）是正确的，系统应开门并记录此次来访信息，包括日期和时间、拥有PIN码的用户姓名等
R4：系统应在2006年5月1日上市
R5：应在PIN码正确输入后0.8秒内开门

课堂讨论：房屋信息系统的需求



课堂讨论：这些需求中哪些是功能性需求、哪些是系统质量，哪些是约束?

R1：系统应该提供一个用户友好的界面

R2：系统能够生成包含所有被允许进入及被拒绝进入房屋的所有记录的月报表

R3：如果用户输入的PIN码（用户识别码）是正确的，系统应开门并记录此次来访信息，包括日期和时间、拥有PIN码的用户姓名等

R4：系统应在2006年5月1日上市

R5：应在PIN码正确输入后0.8秒内开门

功能性需求　　质量需求　　开发过程约束

需求质量要求

- 完整、无遗漏：所描述的需求完整、无遗漏地反映了相关涉众的要求和期望
- 明确、可测试：所描述的需求含义明确，可通过测试进行验证
- 严格、无歧义：所描述的需求表述严格，不同的人阅读之后不会产生不同的理解
- 一致、无矛盾：不同的需求描述之间保持一致，相互之间不存在矛盾

需求工程过程

- 需求获取：识别相关涉众及文档资料、相关系统等其他需求来源，通过多种手段从这些来源那里获得各种需求信息并建立对于待开发系统的初步理解
- 需求分析：分析所收集的各种需求信息，抽取有用的需求陈述并不断进行细化，同时发现需求陈述中冲突或不一致的地方并对需求进行优先级排序
- 需求描述：通过标准的图形化或文本的方式对所获取和分析的需求陈述进行结构化组织和文档化描述
- 需求确认：通过格式审查和内容评审等方式检查文档化的需求描述是否符合标准格式要求、是否完整、准确地反映了相关涉众的期望和要求

以上这些活动并不是一个一次性的顺序过程
而是一个迭代展开的持续性过程

需求管理

- 需求追踪管理
  - ✓ 前向追踪：需求与其来源（如法规条文、用户或客户的原始陈述）之间的追踪关系
  - ✓ 后向追踪：需求与后续的设计、实现、测试制品之间的追踪关系
- 需求复用管理
  - ✓ 目的是在多个不同的软件开发项目中复用相似或相同的需求及其相关的软件制品
  - ✓ 建立跨项目的需求复用库，让不同项目的开发者都可以很方便地发现共性需求以及相关的软件制品并进行复用
- 需求变更管理
  - ✓ 以一种规范化的方式保障需求变化能够平稳、可靠地实施
  - ✓ 一般包括变更请求评估、变更决策、变更实现、变更验证、变更发布

软件需求、用户需求、系统需求

- 用户需求：陈述用户的期望，即希望系统向各种用户提供什么样的服务以及系统运行应该满足哪些约束
- 系统需求：反映开发方与客户和用户协商后达成的关于系统所需要提供的服务、实现的功能及相关约束的一致意见
- 软件需求：系统需求中关于待开发的软件的功能、质量及约束等方面的描述

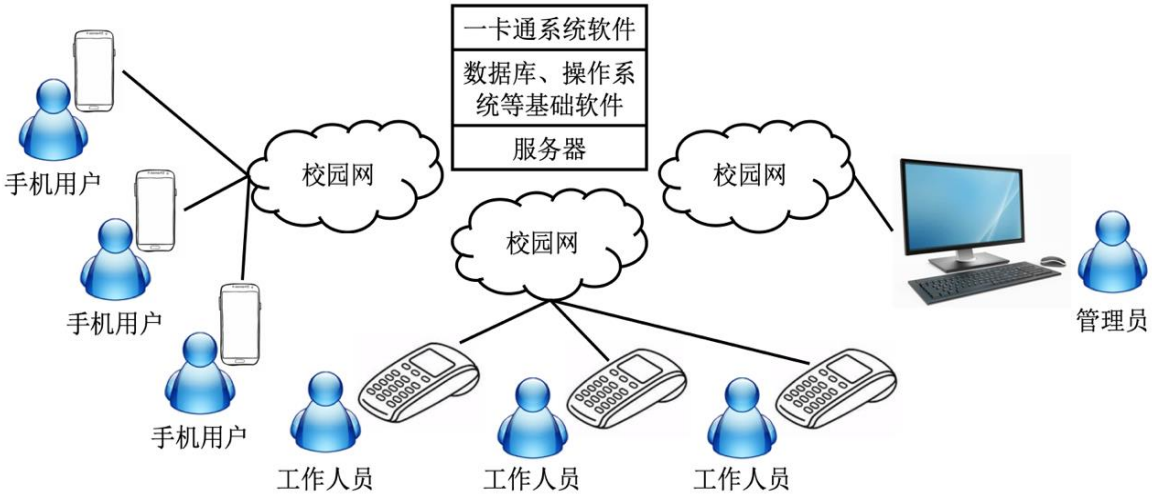
课堂讨论：从用户需求到系统需求

课堂讨论：ATM机软件系统中的用户需求“系统应当提供多种手段防止银行卡被盗刷”可以被细化为哪些系统需求?

单笔取款和全天取款金额有限制
卡长时间无操作吞卡
存取款时进行短信提醒
密码输入错误多次后锁定
读卡器能够识别银行卡真伪
指纹识别用户身份
与监控联网，识别周围是否有人
系统内部网络访问限制IP等

系统需求与软件需求

- 软件并不是存在于真空之中，而是与计算机网络、硬件、设备、人工处理和工作流程等构成一个完整的系统
- 理解软件需求需要建立一种“系统观”：必须在考虑整个系统，特别是软件与其他系统元素的依存关系和职责分配的基础上理解其中软件所需要满足的需求
- 因此，理解软件需求首先要从考虑整个软件密集型系统的需求开始，通过不断分解和精化逐步明确软件与其他系统元素的职责分配，从而使软件需求逐步清晰和明确起来



需求分解与精化

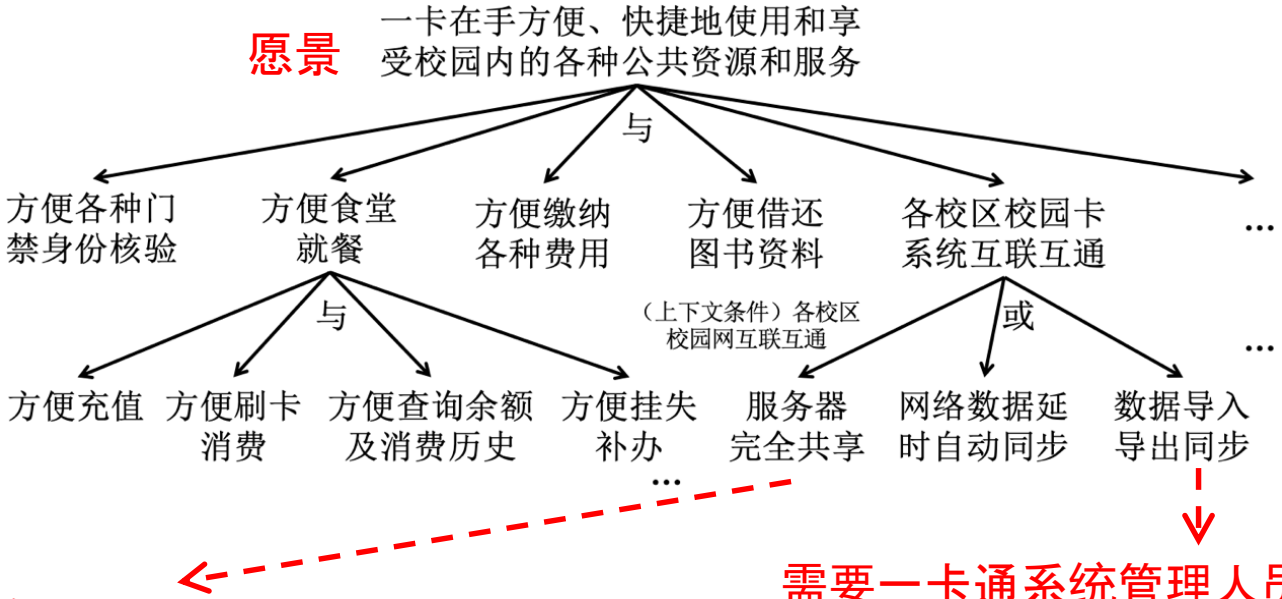
- 项目一般都源自于一个改变当前现状的“愿景”（vision）
- 需求工程的主要目标是在现有的系统上下文环境中建立并实现愿景
  - ✓ 愿景：给出了关于软件系统需要实现的目标的主观愿望
  - ✓ 上下文环境：目标实现可以利用的客观便利条件或必须考虑的客观约束
- 系统上下文环境
  - ✓ 处于待开发的软件系统边界之外且对目标系统的开发及其应用效果有影响的各种因素的综合
  - ✓ 一般包括空间和物理环境、用户群体、文化和社会环境、软硬件和网络基础设施、第三方软件系统与服务等

哪些属于项目内部需要规划并实现的解决方案、哪些属于项目外部的客观上下文环境取决于项目边界的划分

愿景与目标分解和精化

- 愿景需要分解和精化才能用于指导软件开发
  - ✓ 愿景分解：针对愿景提问“如何才能实现”，识别一组更加具体的目标，这些目标的达成可以确保愿景的实现
  - ✓ 目标分解：得到的目标又可以进一步被分解成下一级子目标，方式同样是针对目标提问“如何才能实现”
- 目标分解和精化的两种方式
  - ✓ “与”分解（AND）：所有的子目标之间是“与”的关系，必须全部满足才能保证父目标满足
  - ✓ “或”分解（OR）：所有的子目标之间是“或”的关系，其中任何一个子目标满足就能保证父目标满足
- 目标分解和精化过程不断迭代进行，直至得到可明确指导设计和实现的详细需求，其中伴随着问题和解决方案的不断迭代转换

校园一卡通系统愿景与目标分解



分解和精化

所依赖的上下文环境假设：
各校区校园网互联互通

需要一卡通系统管理人员每隔一段时间进行数据收集和手工导入导出且存在同步延迟，优势是实现较为简单

对于可选(OR)的解决方案，需要从客观(依赖条件是否具备)和主观(权衡利弊)两个方面进行权衡决策

需求问题案例：电水壶控制系统

需求：当水壶中的水温低于100摄氏度时，加热装置应当一直处于工作状态

隐含的（想当然）假设：水的沸点永远是100摄氏度

问题：当假设不满足时（如高原上）水可能被烧干

错误的需求导致错误的产品！

课堂讨论：作为一个面向市场广泛销售的产品，这个需求是否有问题?

几点启示

用户及客户直接表述的需求并不一定代表其背后的深层次意图！

需求表述：水温低于100摄氏度时一直加热

真实意图：将水烧开的同时避免烧干

将用户及客户意图转化为具体可实现的系统需求需要考虑系统上下文！

系统上下文：大气压及其与水的沸点的关系

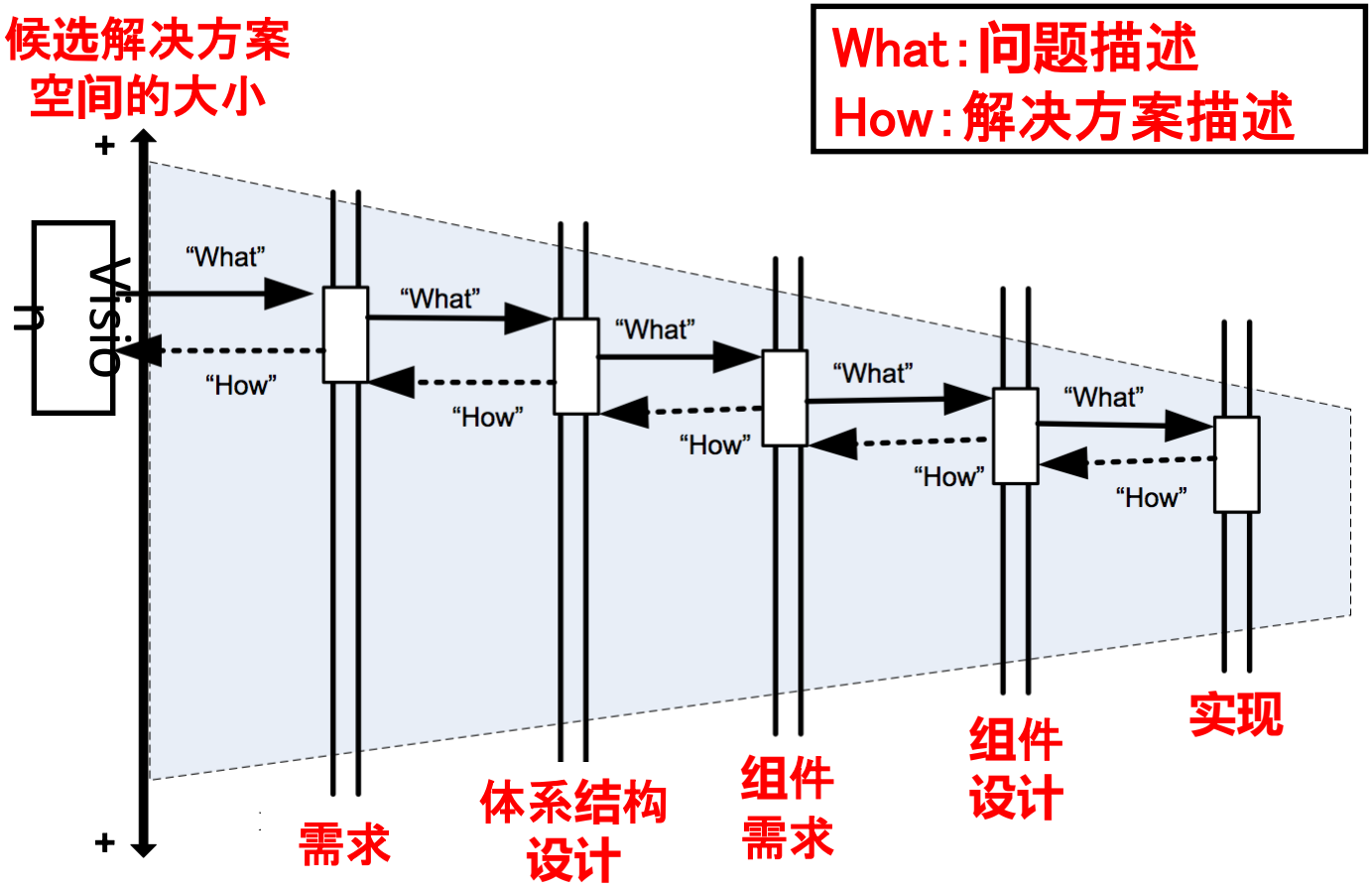
系统需求：假设电水壶在标准大气压下使用的前提下（需在使用说明中体现），要求在水温达到100摄氏度之前一直加热

挖掘用户及客户的深层次意图

明确相关的上下文假设

确定具体可实现的系统需求

从问题到解决方案：What与How的转换



What与How的转换示例

- what：多校区之间实现互连互通（一卡通系统）
  - ✓ How（Solution 1）:各校区之间校园网不连通，其他校区一卡通办公室工作人员每天下班后导出数据通过邮件发送到主校区进行导入同步
  - ✓ How（Solution 2）：各校区之间使用专线直连，使用统一的服务器
- what：登录时验证用户身份（网上银行系统）
  - ✓ How（Solution 1）：用户输入用户名、密码登陆
  - ✓ How（Solution 2）：用户插入随身携带的USB Key，然后输入软口令登录

通过How到What的反向转换识别客户意图

笔试考试计算机辅助阅卷系统

- 客户表述的需求：通过手写识别技术自动识别并记录考生在答卷上的方格内所书写的考号
- 分析客户的目的（what）：准确、自动地识别每一份答卷的考生身份
- 建议的需求方案（How）
  - ✓ 方案1：通过铅笔涂黑的方式确定考号
  - ✓ 方案2：为每位考生发放一张与考号对应的条形码并要求其贴在考卷指定位置上

在此基础上分析利弊（成本、准确性等）引导客户做出选择

需求优先级排序

- 需求并非都是同等重要
- 需求优先级排序对于软件开发有着重要的作用
  - ✓ 当项目开发资源或时间有限，难以按期完成所有需求时需要根据优先级对需求进行取舍
  - ✓ 当需求发生冲突难以同时满足时，要按照优先级进行取舍
- 需求优先级的层次
  - ✓ 基本需求：必须实现软件产品才能使用或者进入市场
  - ✓ 期望型需求：用户、客户及相关涉众明确提出的需求，如果实现那么对于涉众的满意度有正面作用
  - ✓ 兴奋型需求：用户、客户及相关涉众没有意识到某个需求或者没有期望该需求在系统中被实现，但如果实现了可以提高涉众的满意度
- 优先级很多时候也反映了需求之间的依赖关系：低优先级需求往往依赖于一些高优先级的需求

需求冲突识别与协商

- 需求冲突源于不同涉众对于待开发的软件系统的不同看法和利益诉求
  - ✓ 数据冲突：数据冲突是由于缺少信息、错误的信息或者对同一问题的不同理解而引起的
  - ✓ 利益冲突：利益冲突是由涉众主观或客观上的不同利益或者目标引起的
  - ✓ 价值冲突：价值冲突是由涉众评价问题时采用的不同准则（如文化差异）引起的
- 协商的结果一般是一方或多方做出一些让步，或者由上级决定做出一个结论
- 需求冲突也可能是产生创新性想法的契机，双赢的方案可能会成为一个软件产品的创新

示例：创新性的需求冲突解决方式

师生员工代表希望能在所有一卡通消费点（如学校食堂、超市）设置充值点以及供余额和消费历史查询的触摸屏

校方代表希望能控制成本，希望仅设置少量充值点且由充值点工作人员提供余额和消费历史查询服务

冲突源于人工充值点及触摸屏查询这一传统的服务方式

利用手机小程序支持充值和查询这一新的解决方案
既方便了使用，又不增加人工及触摸屏购置和维护成本
而且还可以成为系统的一个创新性的产品特性

相应的具体软件需求中就会增加使用手机小程序实现在线充值以及余额和消费历史查询

需求分析与描述

- 随着需求分解和精化，抽象的系统愿景逐渐被细化为明确和具体的需求
- 此时，需要进一步针对这些需求进行分析，并将分析结果进行描述和记录
  - ✓ 使用最广泛的是场景分析，这种分析方式站在参与者的视角分析与待开发系统的交互过程
  - ✓ 关注于业务实体和实体之间关系的类分析、关注于系统响应外部事件的方式的行为分析也都有一些应用
- 需求分析结果描述方式
  - ✓ 自然语言描述：具有普适性和灵活性强、容易理解和使用的优势，但其固有的二义性是一个主要问题
  - ✓ 模型化描述：具有规范性和精确性的优势，但在普适性、灵活性和易用性等方面则存在不足
  - ✓ 在软件开发实践中，这两种描述方式经常结合在一起使用

需求分析与描述内容

- 场景分析与描述
- 类分析与描述
- 行为分析与描述
- 需求文档

场景分析与描述

- 以一种直观和形象化方式描述和谈论待开发的软件系统将如何与各类用户打交道
- 场景（scenario）表示一组参与者（actor）与待开发系统之间的一系列交互步骤
  - ✓ 参与者是待开发系统边界之外的人（如图书管理员）、设备（如刷卡机）或其他系统（如在线支付系统）
  - ✓ 交互序列有明确的开始和结束，构成一个相对完整的整体，其目的是满足相关涉众的一个或一组目标
  - ✓ 存在一些失败场景，其中所追求的目标最终并未实现
- 具有相同目标且交互过程相似的一组场景可以被组织为一个用例（use case）
  - ✓ 存在一个在大多数情况下会发生且满足目标的主场景
  - ✓ 还包括与主场景有所差异的可能场景以及异常处理场景

现场图书借阅用例的几个不同场景

类型	场景名	简要解释
主场景	现场刷卡借阅图书成功	图书管理员利用系统刷读者校园卡成功为读者办理现场图书借阅手续
可能场景	现场输入卡号借阅图书成功	图书管理员利用系统输入读者卡号并核实身份后成功为读者办理现场图书借阅手续
	在借图书超限导致图书借阅失败	由于读者在借图书超过指定上限，因此图书管理员无法帮助读者完成本次图书借阅
	存在未缴纳罚款导致图书借阅失败	由于读者存在未缴纳的图书罚款，因此图书管理员无法帮助读者完成本次图书借阅
异常场景	校园卡刷卡异常	读者校园卡刷卡因不明原因失败，提醒读者去一卡通管理中心检查校园卡问题并更换，同时可改用输入卡号的方式进行借阅
	图书扫描异常	图书扫描因不明原因失败，可改用输入图书唯一编码的方式进行借阅处理

用例参与者的类型

- 由人来扮演的角色
  - ✓ 需要与目标系统交互的各类用户，代表一种抽象的角色
  - ✓ 例如食堂消费者、食堂窗口工作人员、图书馆读者等
- 外部设备
  - ✓ 与目标系统交互的各种外部设备，包括传感器和执行器
  - ✓ 外部设备拓展了软件系统感知和操控物理世界的能力
  - ✓ 例如条码扫描枪、温度和湿度传感器、机械臂等
- 外部系统
  - ✓ 与目标系统交互的其他外部软件系统，可以提供所需的服务或信息或者请求目标系统提供所需的服务或信息
  - ✓ 例如提供在线支付服务的第三方在线支付系统
- 定时器
  - ✓ 一种特殊的参与者，会定时触发预定义的处理逻辑

与参与者的交互接口

- 除了定时器外其他参与者都位于目标软件系统边界之外，他们（它们）与目标系统进行交互，但并不属于目标系统的一部分
- 目标系统通过开发各种接口（interface）来实现与这些参与者的交互
  - ✓ 与外部用户角色交互的用户接口（又称用户界面，简称UI）
  - ✓ 与外部设备交互的硬件接口（如访问校园卡读卡器的接口）
  - ✓ 与外部系统交互的软件接口（如第三方在线支付服务接口）
- 目标软件系统中所实现的这些接口属于系统的一部分，而这些接口交互的对象（例如用户、硬件、外部系统）则属于系统边界之外

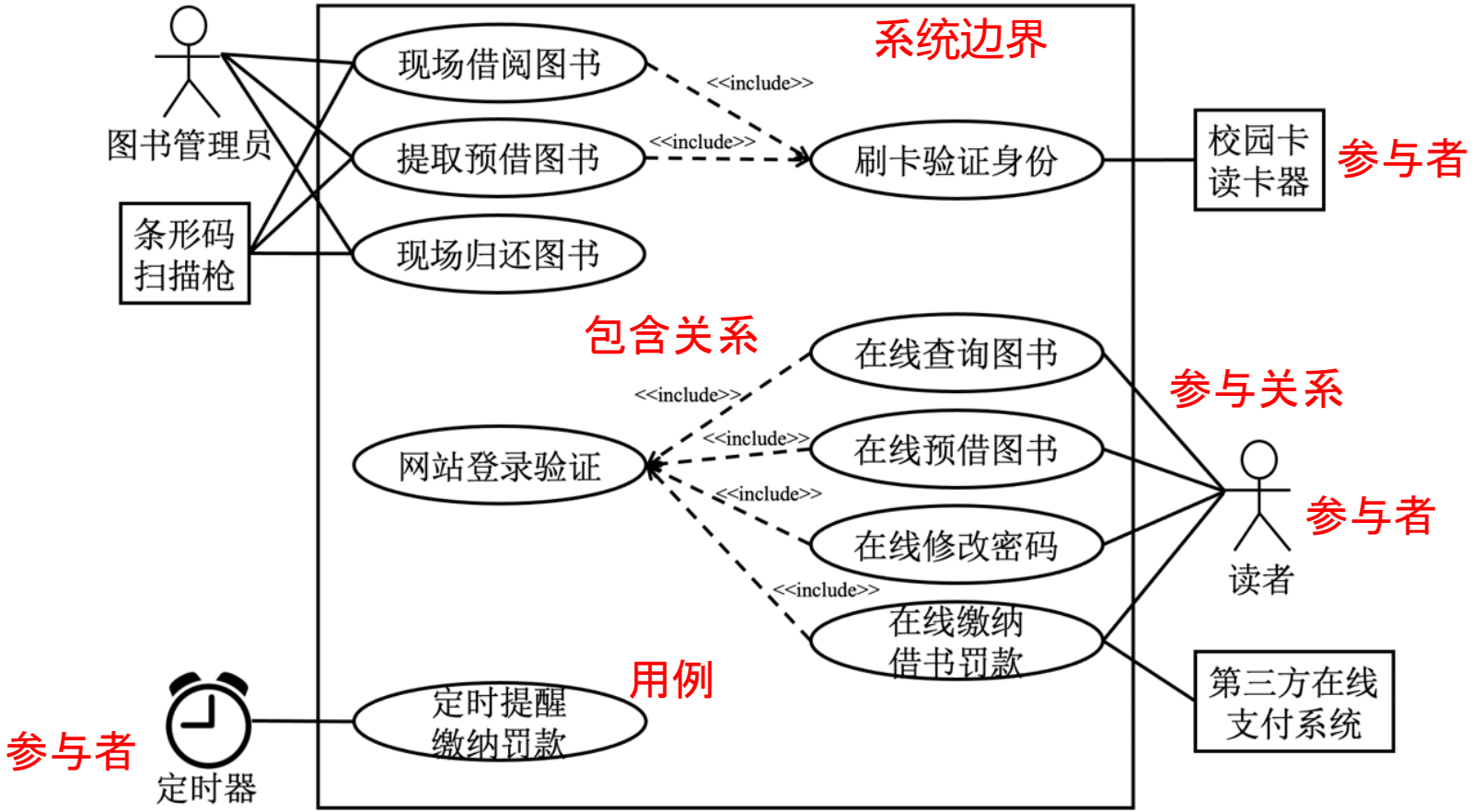
识别用例

- 识别参与者后，我们可以针对参与者考虑其与目标系统可能存在哪些典型的交互，即参与者会在哪些情况下与目标系统打交道
- 其中每一种情形都有可能成为一个需求用例，可以进一步按照以下标准进行判断
  - ✓ 过程明确且完整：用例应当包含明确的交互过程，可以明确识别其中的交互步骤，交互过程有明确的开始和结束
  - ✓ 可独立完成且不可再分：用例可以在不依赖其他用例的情况下独立完成，内部不可再细分得到更小的独立用例
  - ✓ 对于用户有价值：用例所包含的交互过程如果完成对于相关用户有一定的价值

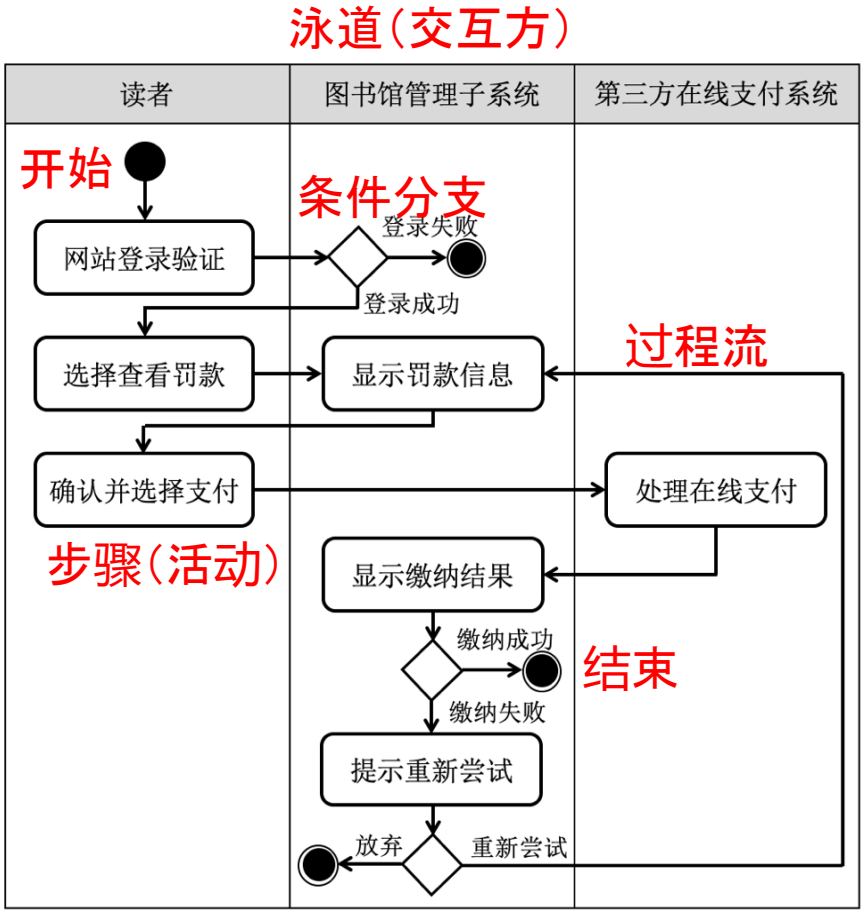
图书馆管理子系统中的几个候选用例

候选用例	原因说明
现场借阅图书	由读者在图书馆现场触发，有完整的交互过程，可独立完成且不可再分，对于读者有价值
现场归还图书	由读者在图书馆现场触发，有完整的交互过程，可独立完成且不可再分，对于读者有价值
在线缴纳借书罚款	由读者在系统网站上在线操作触发，有完整的交互过程，可独立完成且不可再分，对于读者有价值
在线修改密码	由读者在系统网站上在线操作触发，有完整的交互过程，可独立完成且不可再分，对于读者有价值
定时提醒缴纳罚款	由定时器定时触发，有完整的交互过程，可独立完成且不可再分，对于读者有价值
在线预借图书，等待通知后现场取书	在线预借图书和现场取书虽然存在因果关系，但是相对独立的两个交互过程，在线预借图书成功后即使暂不取书也算完成了一件事情
刷卡验证身份	独立完成没有什么意义，往往作为其他包含现场刷卡的用例（如现场借阅图书、提取预借图书）交互过程的一部分出现
网站登录验证	独立完成没有什么意义，往往作为其他在线完成的用例（如在线缴纳借书罚款、在线查询图书信息）交互过程的一部分出现

用例概览描述（UML用例图）



用例详细描述（UML泳道图）



针对“在线缴纳借书罚款”用例的详细描述，展开描述用例内部的交互序列

UML泳道图是活动图的一个变体，此外还可以用UML顺序图来描述

用例详细描述（自然语言模板）

- 目标：相关参与者希望在用例中实现的目标
- 前置条件：用例发生所需要满足的前提条件
- 触发条件：触发用例执行的事件及需要满足的条件
- 主场景：在多数情况下会发生且满足相关目标的成功场景，具体描述为一系列交互步骤
- 其他场景：可能发生的在部分步骤上与主场景有所差异的其他场景，可以有多个
- 异常场景：可能发生的反映异常情况及相应的异常处理方式的场景，可以有多个
- 发生频率：当前用例发生的频率

用例详细描述（自然语言模板）示例

用例名称：在线缴纳借书罚款

参与者：读者、图书馆管理子系统（目标软件系统）、第三方在线支付系统

目标：读者希望能方便快捷地完成缴纳罚款的过程，同时罚款原因、金额、产生时间等明细信息公开透明，缴纳方式安全可信，结果能够准确、及时反馈；图书馆管理子系统希望读者能够及时缴纳罚款，同时第三方支付稳定可靠，支付结果准确无误。第三方在线支付系统希望单位时间内的支付请求量不要过高，以免对系统造成较大的压力。

前置条件：当前用户拥有图书馆读者卡且读者卡处于正常可用状态。

触发条件：读者登录网站并选择缴纳借书罚款。

主场景：读者一次性完成借书罚款缴纳

1）读者进行网站登陆验证。
2）读者登录成功后选择查看罚款。
3）系统显示罚款信息。
4）读者确认罚款信息并选择支付。
5）第三方在线支付系统处理在线支付。
6）系统显示罚款缴纳成果结果。

其他场景：

1. 登录失败

1）读者进行网站登陆验证。
2）系统显示登录失败。

2. 罚款缴纳失败后再次尝试成功

按照主场景执行到显示缴纳结果之后
1）系统显示缴纳失败并提示重新尝试。
2）读者选择重新尝试。
3）系统显示罚款信息后按照主场景继续执行，最终缴纳成功。

3. 罚款缴纳失败后读者选择放弃

按照主场景执行到显示缴纳结果之后
1）系统显示缴纳失败并提示重新尝试。
2）读者选择放弃。

异常场景：等待第三方在线支付结果超时

按照主场景执行到处理在线支付之后
1）系统等待第三方在线支付结果超时。
2）系统提示读者稍后电话联系图书馆服务中心核对支付结果。

发生频率：早上9点到晚上12点期间平均每小时10次，高峰期（如学生毕业离校手续办理截止日前夕）每小时可达500次。

类分析与描述

- 关注于对象属性以及相互之间的关系
- 理解场景、功能等需求描述的基础，也是设计阶段考虑数据结构（如数据库表）设计的基础
- 基于面向对象的类分析方法
  - ✓ 类是属性和操作的统一封装体，代表着同一类对象或实体的共性抽象
  - ✓ 类包含一组所有对象或实体都具有的属性和操作
  - ✓ 类之间存在多种不同类型的关系：继承（Inheritance）、聚合（Aggregation）、关联（Association）、依赖（Dependency）等
- 关心分析类：不考虑设计和实现方案的情况下分析需求的过程中可以识别出来的类

不同类型的分析类

- 外部实体：产生或使用目标软件系统信息的外部实体，例如人员、设备、其他系统
- 事物：目标软件系统所管理的各种事物，例如订单、发票、商品
- 事件：发生的事件，一般具有时间戳，例如报警、来访
- 角色：由人扮演的各种角色，例如读者、管理员
- 组织单元：人类社会的各种组织单位，例如公司、部门、学校、院系
- 场地：现实世界的场地空间，例如房间、楼宇
- 结构：各种结构体，例如车辆、电器

识别分析类：选取标准

- 包含系统运行所需要的信息
  - ✓ 分析类应当包含一些系统运行所不可或缺的信息
  - ✓ 某个概念或事物在现实世界中可能具有很多信息，但并不都是当前软件系统所关心的，这取决于系统需求和关注点
  - ✓ 分析类可以通过属性来表达相关的有用信息，一般也同时包括改变这些属性的操作
- 包含多个而不是单个属性
  - ✓ 有些候选分析类仅包含单个属性，那么可以作为其他分析类的一部分，而不用单独作为分析类
  - ✓ 例如，图书作者应当作为分析类还是图书类的属性取决于系统需要记录和处理的作者信息有哪些：如果只需要记录姓名以进行展示或关键字查询，那么可作为图书类的一部分；如果需要从多方面记录作者信息（如个人简介、作品等）从而支持详细查询和分析，那么应当作为分析类

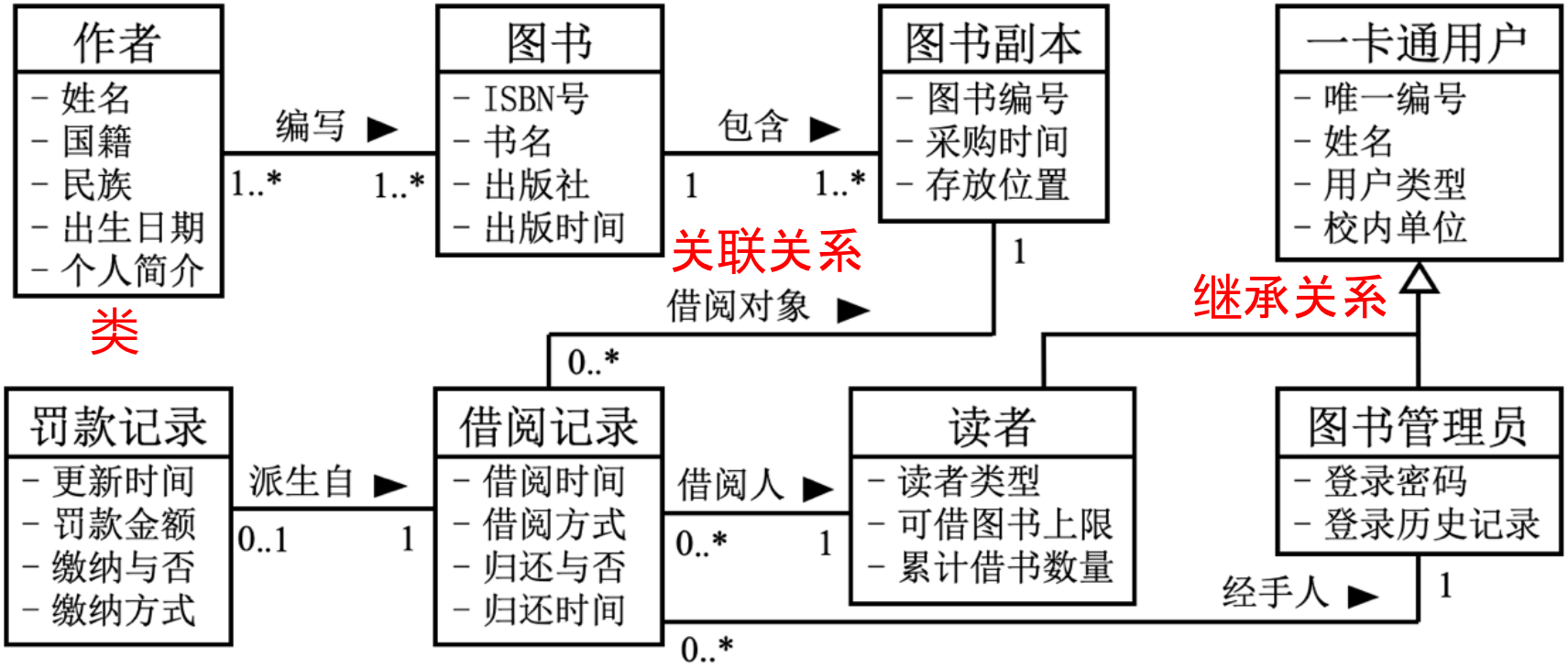
图书馆管理子系统中的候选分析类

候选	原因说明
读者	系统的主要服务对象，与图书借还、罚款缴纳等业务直接相关，是一卡通用户且包含读者类型、可借图书上限、累计借书数量等多种信息
图书管理员	参与现场借阅图书等需现场处理的业务过程且需要记录参与操作的管理员信息，包含登录密码、登录历史记录等多种信息
作者	在图书查询和图书推荐等业务过程中都需要查询和分析图书作者，包含姓名、国籍、民族、出生日期、个人简介等多种信息
图书	各种与图书相关的业务过程中都会涉及，包含ISBN号、书名、作者封面图片、出版社、出版时间等信息
图书副本	各种与图书实物相关的业务过程中都会涉及，包含图书编号、采购时间、存放位置等信息
图书馆员工	当前系统仅关注于与图书及借还管理相关的业务，所涉及的图书管理员仅是一部分员工，普通员工的人事信息（如年龄、籍贯）等不在当前系统关注的范围内
图书采购单	当前系统仅关注图书副本进入图书馆后的相关服务和管理业务，图书采购不在系统关注的范围内
出版社	出版社信息在图书查询和图书信息展示过程中都有所涉及，但仅需要以文本字符串的方式存在，属于单一属性，可以作为图书类的一部分

分析类的职责划分

- 软件系统实现各个用例和场景的能力要求需要分配到各个分析类上，成为分析类的职责要求
  - ✓ 完整覆盖：所有类的职责分配总体上应当完整覆盖目标软件系统的整体需求以及在各个用例和场景中的能力要求
  - ✓ 合理均衡：职责应当在相关的类之间合理、均衡分布，每项职责都应当分配给与之相关性最高的类
- 类的职责通过将自身能力（属性和操作）与其他类的协助相结合来实现
  - ✓ 类的操作与属性密切相关：查询实例对象状态和信息、操纵实例对象数据、基于实例对象数据的计算操作、监控实例对象中控制事件的发生
  - ✓ 类之间的协作关系主要表现为一个类请求另一个类所拥有的属性和操作：借助于类之间已有的继承、聚合或关联关系来实现或者在类之间建立依赖关系

类模型描述（UML类图）

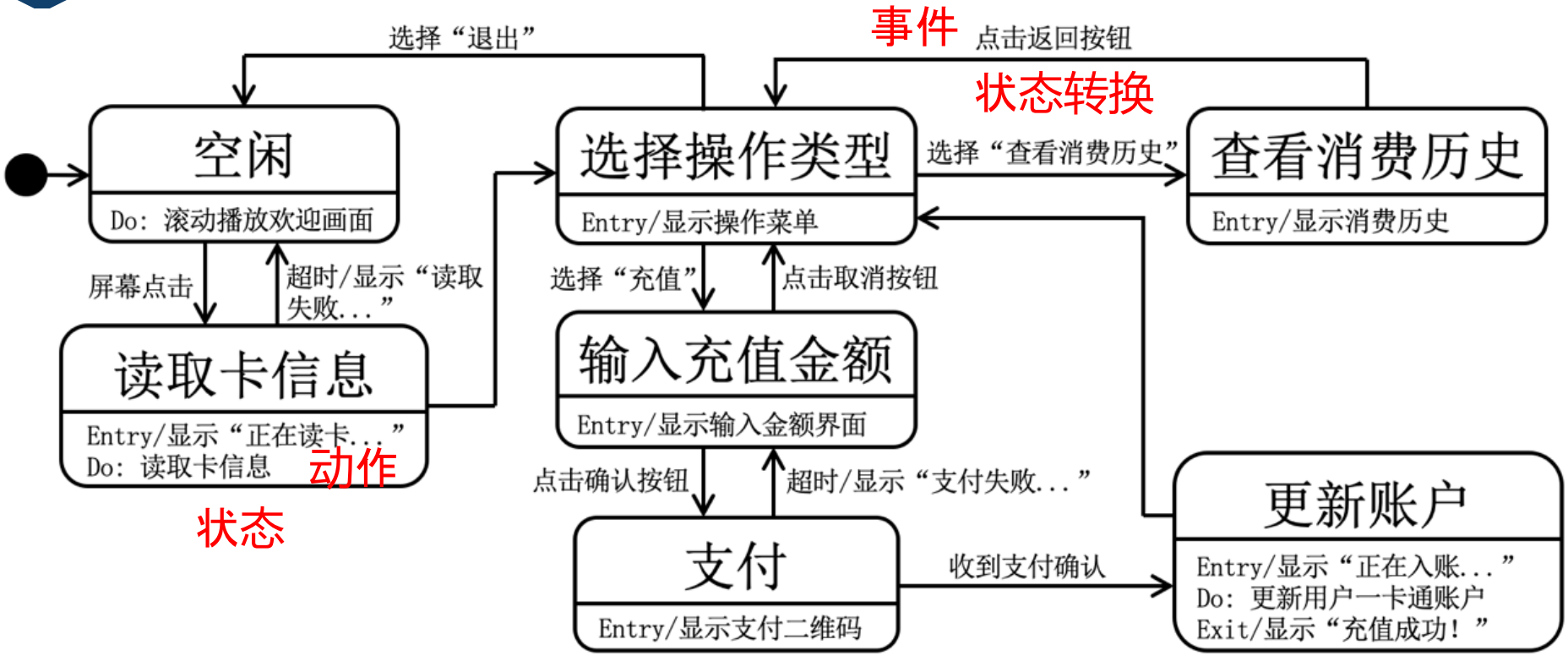


校园一卡通系统图书馆管理子系统部分类模型

行为分析与描述

- 行为：系统或其一部分响应外部的事件激励而执行一些动作，同时伴随自身状态的变化
- 行为分析从系统本身而非场景分析中的外部参与者的角度了解到系统如何响应外部事件进行状态转换并执行各种动作的
- 行为分析与描述的基本内容
  - ✓ 状态：行为主体所处于的某种情形，一般会维持一段时间（短则几秒甚至更短，长则几小时甚至更长）
  - ✓ 事件：影响行为主体行为的外部或内部事件，推动行为主体状态的不断转换和各种动作的执行
  - ✓ 状态转换：行为主体各种状态之间的转换，一般是在事件的激励下发生的，有时也会附加相应的条件要求
  - ✓ 动作：维持某个状态期间或状态转换过程中执行的动作

行为模型描述（UML状态机图）



校园一卡通系统自助充值子系统行为模型

需求文档

- 需求分析完成后需要形成一份需求文档作为后续开发活动（如设计、实现、测试）的基础
- 主要通过自然语言进行书写，同时穿插一些图表和其他形式化描述，例如用例图、泳道图、类图、状态机图、规则表格、计算公式等
- 一般都会参考规范的需求文档模板
- 使用自然语言描述需求的优缺点
  - ✓ 优点：易理解，表达能力强，使用灵活，适用范围广，很容易描述高层意图、需求目标、功能和非功能性要求等需求内容并根据需要调整描述的抽象程度和粒度
  - ✓ 缺点：表达方式随意，容易出现二义性和模糊性，即不同的人针对同一个需求表述可能会产生不同的理解和解读

IEEE推荐的需求文档组织结构

章	内容概述
前言	明确文档的目标读者人群，描述文档的版本历史（每个版本的修订时间、修订原因以及内容修改的概述等）
引言	这部分描述系统的意义和必要性，即为什么需要开发这个系统。需要简要描述系统的功能以及该系统如何与其他相关系统集成和一起工作，同时描述该系统如何服务于委托开发方的总体业务或战略目标。
术语表	定义文档中所使用的专业术语，帮助缺少相关专业经验和知识的读者理解相关术语。
用户需求定义	描述为用户提供的服务以及相关的非功能性系统需求。这些描述可以使用客户能够理解的自然语言、图形化或者其他表达法。同时还要描述当前项目开发必须遵循的产品和过程标准。
系统体系结构	描述所预计的系统体系结构的高层概览，显示各个系统模块上的功能分布。系统体系结构中复用的组件应当进行明确和强调。
系统需求规格说明	更详细地描述系统的功能和非功能性需求。可以根据需要进一步增加非功能性需求中的细节信息，并定义与其他系统的接口。
系统模型	提供图形化的系统模型，用于描述系统组件之间的关系以及系统与环境之间的关系。可能的模型包括对象模型、数据流模型、或语义数据模型等。
系统演化	描述考虑当前系统需求时所基于的基本假设，以及预计由于硬件演化、用户需求变化等原因可能导致的变化。这部分信息对于系统设计很重要，因为系统设计者了解这些信息可以避免在做出设计决策时对未来可能发生的变化造成不必要的限制。
附录	描述与所开发的应用相关的一些特定的详细信息，例如硬件和数据库描述。硬件需求明确说明系统所要求的最低配置和优化配置。数据库需求明确说明系统所使用的数据的逻辑组织以及数据之间的关系。
索引	提供相关的文档索引。除了常规的字母序索引外，还可以包括文档中图的索引以及需求中功能定义的索引。

自然语言需求二义性的主要原因

- 描述不充分
  - ✓ 需求描述缺少必要的细节，从而导致不同的人可能会做出不同的细节假设，从而产生对同一需求的不同理解
  - ✓ 例如，“如用户访问记录存在异常则拒绝登录”这一需求描述中的“访问记录存在异常”缺乏相关的细节描述
- 模糊的术语
  - ✓ 需求文档中使用的术语未经过严格定义，不同的人可能对其含义产生不同的理解
  - ✓ 例如，“高年级学生选课时系统应当提醒学生核对毕业要求”这一描述中的“高年级学生”存在模糊性
- 自然语言固有的二义性
  - ✓ 自然语言所固有的词法、语法、指代等方面的问题都可能导致语义上的不同理解，从而造成二义性
  - ✓ 例如，“系统显示最近5笔充值和消费记录”中的“5笔”可能导致不同理解

自然语言需求描述的指导原则

- 定义需求文档术语表并在需求描述中统一使用标准术语
- 使用规范化的需求文档模板和需求描述句式，例如主动语态以及“系统应当…”、“系统可以…”这样的句式
- 使用带编号的句子来描述基本需求条目并保持原子性（每个句子表达一条不可再分的原子需求）
- 采取多种手段以避免引入二义性
- 在适当的地方结合规则表格、公式、UML模型等其他更精确的描述方式
- 结合使用半结构化自然语言，即定义标准的自然语言表格或模板并明确其中每个字段的含义

敏捷开发中的需求工程

- 认为需求变化非常之快，因此不使用正式的需求文档，而是经常增量地收集需求并以用户故事（User Story）的形式写在卡片或白板上
- 用户故事
  - ✓ 一种从用户视角出发表述的端到端的细粒度功能，是对用户或客户有价值的功能点的简洁描述
  - ✓ 用户故事之间是解耦的，每个用户故事都可以独立交付，是敏捷迭代交付的基础
  - ✓ 一次敏捷迭代的时间周期一般是1-2周，因此进入迭代的需求粒度不能太大
  - ✓ 大的用户故事一般被称为史诗故事（Epic）
- 用户故事可以按照对于用户的价值和紧迫性进行优先级排序，从而决定在下一次迭代计划

用户故事的INVEST原则

- I（Independent）：独立，能够单独进行实现
- N（Negotiable）：可协商，要求不能定得太死，开发过程可变通
- V（Valuable）：有价值，对于客户和用户有意义
- E（Estimable）：可估算工作量及进度（不能估算往往意味着无法作出相对准确的计划，一般是因为粒度还不够小）
- S（Small）：足够小，能在一个迭代内完成，不能跨越多个迭代
- T（Testable）：可测试，能够对应设计测试用例来验证用户故事是否实现

用户故事典型的描述格式

作为一个<角色>，我想要<活动>，以便于<商业价值>。

As a <Role>, I want to <Activity>, so that <Business Value>.

例如，在图书馆管理系统中可以有如下用户故事：

- ✓ 作为一个“读者”，我想要“在读者所预约的图书归还到馆后立刻短信通知读者”，以便于“读者及时借到想看的图书”
- ✓ 作为一个“图书管理员”，我想要“统计每月借阅的热门书籍”，以便于“图书馆梳理待选购的新书清单”

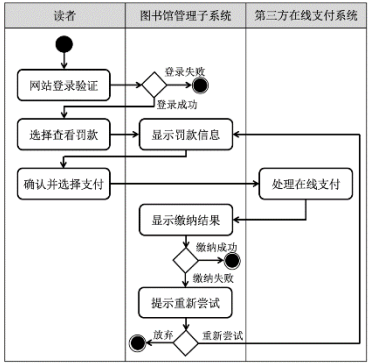
用户故事不能使用技术性的语言来描述，而是要使用用户可以理解的业务语言来描述

用户故事与需求用例的关系

- 二者都可以从用户角度陈述需求
- 用户故事与需求用例的区别
  - ✓ 从范围上看，需求用例覆盖的范围一般更大一些，用户故事需要适应敏捷迭代开发的需要，因此一般需要对所覆盖的范围和内容进行限制，以便进行任务安排和调度
  - ✓ 从生命周期上看，需求用例通常是永久性的工作制品，而用户故事具有短时性，往往仅在所属的迭代中发挥作用
  - ✓ 从目的上看，用例是为了让开发人员与客户或用户代表一起进行讨论并就需求达成一致，而用户故事的目的是为了发布迭代开发和交付计划并引导客户和用户提供更多细节
- 在敏捷开发中需要基于INVEST原则将一些较大的需求用例转化为多个用户故事

用户故事分解示例

针对“在线缴纳借书罚款”用例，根据INVEST 原则拆分得到若干用户故事



作为一个“图书馆管理方”，我想要“读者在查看并缴纳罚款之前通过身份验证”，以便于“确保系统安全和用户隐私保护”。

作为一个“读者”，我想要“在缴纳罚款之前查看并确认罚款信息”，以便于“确保系统所收取的罚款都是合理的”。

作为一个“读者”，我想要“通过在线支付缴纳罚款”，以便于“实现业务便捷办理”。

作为一个“读者”，我想要“通过在线支付缴纳罚款后系统立即告知支付结果，如不成功允许再次支付”，以便于“系统方便使用”。

通过用户故事拆分，不同类型用户所关心的特性及价值变得更明确了，更细粒度的需求表达为迭代开发计划制订和执行提供了便利

例如，“在线缴纳借书罚款”用例可以通过多次迭代实现：首先实现一个登录后显示罚款金额并直接支付的版本，完成最基本的在线缴纳借书罚款功能；在下一次迭代中增加罚款信息确认的特性，使读者在缴纳罚款前可以查看罚款明细并确认；在后续迭代中继续完善并实现了在线支付缴纳罚款后的系统结果告知以及重试的特性。

本章小结-1

- 全面、准确的需求理解以及规范化描述是成功的软件开发项目的重要基础
- 软件并不是存在于真空之中，而是与网络、硬件、设备、人工处理等构成完整的系统
  - ✓ 需要从高层客户愿景和用户需求出发通过逐层分解和精化得到更具体的系统需求
  - ✓ 需要在系统需求的基础上，根据软件与其他非软件系统组成部分之间的职责划分确定软件需求
- 软件需求是一个非常复杂的整体，需要从不同的角度去观察和分析
  - ✓ 场景分析、类分析和行为分析等方法从外部参与者及交互序列、业务对象及关系、系统外部行为等不同方面分析
  - ✓ 同时提供了不同的UML模型作为需求描述手段

本章小结-2

- 与传统软件开发不同，敏捷方法强调持续增量地收集用户需求并描述为用户故事，以有价值且独立的用户故事作为迭代计划制订和执行的基本任务单元

COMP130015.02
软件工程

End

8. 软件需求