高级软件工程

软件需求工程

彭鑫

pengxin@fudan.edu.cn

www.se.fudan.edu.cn/pengxin

内容提要

- 需求工程概述
- 系统与上下文分析
- 需求制品
- 需求工程活动

内容提要

- 需求工程概述
- 系统与上下文分析
- 需求制品
- 需求工程活动

软件开发的目的

- **以软件产品或软件服务的形式为用户（个人或组织）提供现实世界问题的解决方案**
- **传统的软件开发项目**
  - 对某项任务（如文档编辑、税金计算）的自动化处理
  - 对某些业务流程（如订单处理流程）的自动化支持
  - 对于硬件设备（如工业生产设备、家用电器）的自动化控制
- **新型的软件开发项目**
  - 越来越多地面临着实现创新性功能和应用模式的要求
  - 例如：新型的社交网络及电子商务应用、基于嵌入式软件的ABS（防抱死刹车系统）等新型汽车电子应用

软件需求

- **软件需求是对软件产品或服务所需要具备的外部属性的一种刻画：这些属性应当保证所提供的解决方案能够满足用户所需要解决的现实世界问题的要求**
  - 我们在构造任何人工制品或系统（如建造房屋、生产食品、制造汽车等）时都要事先明确目标和要求
  - 软件开发中的需求和需求工程显得尤为重要：与一般的人工制品或系统不同，软件系统是无形的而且相应的软件开发目标具有抽象性

创新性的软件系统

- **传统的软件系统**
  - 以自动化信息处理和自动化控制为目的
  - 软件需求主要是对现实世界中已经存在的业务过程和任务处理逻辑的理解和刻画
  - 软件需求的获取和分析相对容易
- **创新性的软件系统**
  - 软件成为实现创新性应用模式的关键和支撑：社交网络应用、新型电子商务、汽车电子系统...
  - 软件需求则更多地要以创造性、探索性的方式来获取和分析，其难度更高

软件需求的定义

1）用户解决某个问题或者达到某个目标所需要的条件或能力；

2）一个系统或系统组件为了实现某个契约、标准、规格说明或其他需要遵循的文件而必须满足的条件或拥有的能力；

3）对1）或2）中所描述的条件或能力的文档化表示。

IEEE 610.12-1990（软件工程术语）

软件需求示例

财务管理软件系统

- **用户主观期望和要求的系统能力和条件**
  - 系统能力：凭证录入、支出规则控制、财务报表生成等功能
  - 质量要求：负载能力（如可以支持100名用户同时登录进行账务处理）、安全性（如用户密码不能被包括管理员在内的其他用户查询到）等
- **相关的法律、法规、规章、标准、规格说明等所形成的约束**
  - 行业及部门颁布的法规和管理制度：需要上报的财务报表格式、凭证编码规范等
- **对于以上两类需求的文档化形成的需求制品：交流与沟通的媒介、软件维护的重要依据**

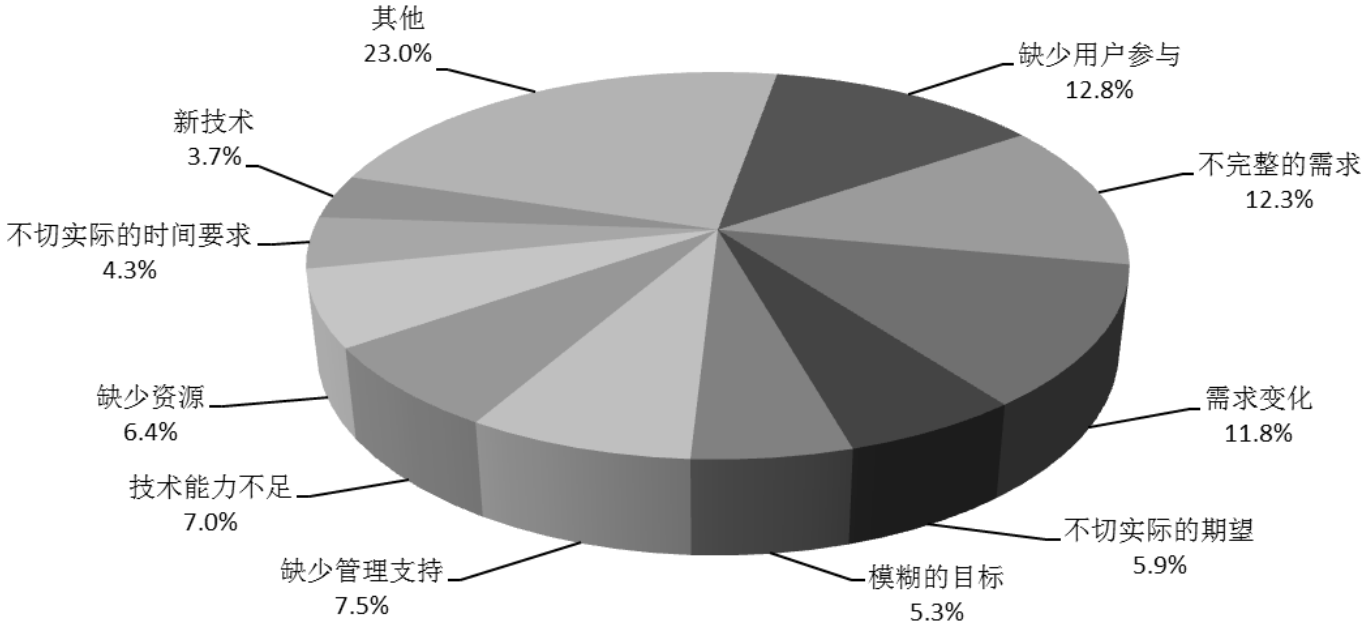
三类软件需求

- **功能性需求：系统应当向用户提供的功能**
  - 系统应提供哪些服务
  - 系统如何响应特定的输入
  - 系统在特定情形下的行为（即应当做什么、不应当做什么）
- **质量需求：待开发系统的质量属性**
  - 系统的性能、可靠性、稳定性等
  - 可以针对系统整体进行定义，对系统产生全局性的影响
  - 也可以针对特定的服务、功能或系统组件定义质量属性
- **约束：针对开发过程或待开发系统自身属性的一种限制**
  - 来自其他组织过程（如来自项目管理的时间或资源约束）
  - 来自系统运行的上下文环境（如所在组织和领域所适用的法律法规等）

软件需求与项目失败

Standish集团研究报告

- 1994-2009年间软件开发项目：完全取得成功的项目总体徘徊在30%左右，而失败或部分失败（如项目严重超支或需求实现不完整）的项目则占到了65%以上
- 造成项目部分失败的各种原因及其分布



伦敦救护车服务系统案例

- 自动化的急救电话处理以及救护车调度
- 项目失败的主要原因就是对于需求的考虑不充分，例如
  - 对救护车通讯设备的用户接口需求考虑不当，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，同时救护人员感到难以操作这些设备
  - 未考虑一些特殊情况，如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
  - 未能充分考虑网络通信的状况，例如该系统完全没有考虑无线电通信盲点的问题
- 系统运行后产生一系列令人无法接受的状况
  - 将未就绪的救护车调度到呼救地点，导致延迟并直接危及人的生命
  - 将超出所需数量的救护车调度到同一个呼叫地点，造成资源浪费

需求相关的问题和困扰

- 具体表现：需求模糊、不确定；需求不完整；相关各方对于需求的理解不一致；需求的频繁变化和管理失控等
- 问题原因
  - 软件自身的复杂性和不可见性决定了软件需求与其他工程化需求相比更加难以准确、全面地获取和刻画以及有效管理
  - 用户、客户与软件开发人员之间在知识背景和思维方式上的巨大鸿沟导致了双方在沟通和交流上的困难
  - 缺少系统化的过程、方法、技术和工具支持，需求的获取、分析、描述和管理等仍然主要依赖于个人能力和经验

需求制品

- **三类主要的需求制品**
  - 目标：早期需求分析的产物，刻画了相关涉众对于待开发系统的期望和意图
  - 场景：通过一系列交互步骤给出了满足或不满足目标的具体实例
  - 面向方案的需求：从解决方案的角度更加具体和一致地描述了待开发软件的数据、功能和行为等方面
- **需求制品的描述方式**
  - 自然语言
  - 概念模型（例如UML模型、数据流模型等）
  - 形式化语言（例如Z语言）

涉众（STAKEHOLDER）

- **涉众：与待开发系统存在利益相关性，同时对系统最终的外部特性存在影响力的人或组织**
  - 用户：直接或间接参与系统使用过程的人或组织，例如系统的直接操作人员、为系统提供信息或从系统接收信息的人员
  - 客户：委托开发方进行系统开发或作为市场销售对象的人或组织
  - 市场人员：开发组织内部或第三方针对待开发系统相关市场的分析人员
  - 领域专家：开发方或客户组织内对相关业务领域有着丰富的经验和业务积累的专家
  - 软件工程师：开发组织内部的软件工程师，包括架构师、开发人员、测试人员和维护人员等

需求工程活动

- 需求获取：识别涉众及其他相关的需求来源，收集并获取初始的系统需求信息，建立起对于待解决问题的基本认识，初步明确待开发系统的范围
- 需求分析：在所收集的需求信息基础上，进行分析、整理和综合，识别并解决其中所隐含的冲突，从系统需求中导出细化的软件需求
- 文档化：针对需求分析所得到的软件需求，通过各种建模语言以及自然语言文本进行规范化地描述和记录，形成需求文档和规格说明
- 需求确认：通过多种手段验证需求文档和规格说明符合相关的格式规范、满足相关质量属性，同时确认所得到的需求与所要解决的问题相符

持续的需求工程

- **阶段性的需求工程：需求工程是一种在软件开发过程初期开展的一次性活动**
  - 缺乏持续性，无法及时反映变化
  - 需求分析过程十分费时费力
  - 不支持系统化的需求复用
  - 关注点十分狭隘，难以容纳创新性的想法
- **持续的需求工程：贯穿整个软件生存周期以及跨越项目和产品边界的持续性过程**
  - 时间维度：伴随着迭代式的软件开发过程持续进行，持续地从此前的迭代过程中获得反馈同时将对于待开发系统更深层次的理解和新的想法融入进来
  - 空间维度：建立起统一的领域或组织范围内的需求库，从而支持跨项目、跨产品的需求制品分析、维护、演化和复用

内容提要

- 需求工程概述
- 系统与上下文分析
- 需求制品
- 需求工程活动

软件密集型系统

- **软件密集型系统（Software-Intensive System）：软件对于该系统整体的设计、构造、部署和演化起着根本性的影响作用**
- **强调系统的整体性以及其中的软件元素与其他非软件元素的共同作用**
  - 传统的ERP软件必须与所属组织机构中各个岗位上的工作人员、规章制度、网络基础设施等非软件部分相结合才能发挥系统的整体作用
  - 汽车电子系统中的软件单元必须与刹车、油门和转向控制等电子和机械单元一起密切配合才能实现防抱死制动、自动巡航等汽车控制功能
- **强调软件所发挥的重要甚至是决定性的作用**
  - 核心功能（如信息管理、分析决策）或质量一般都是由软件实现的
  - 软件正越来越多地成为开发创新性产品和服务的关键

两类软件密集型系统

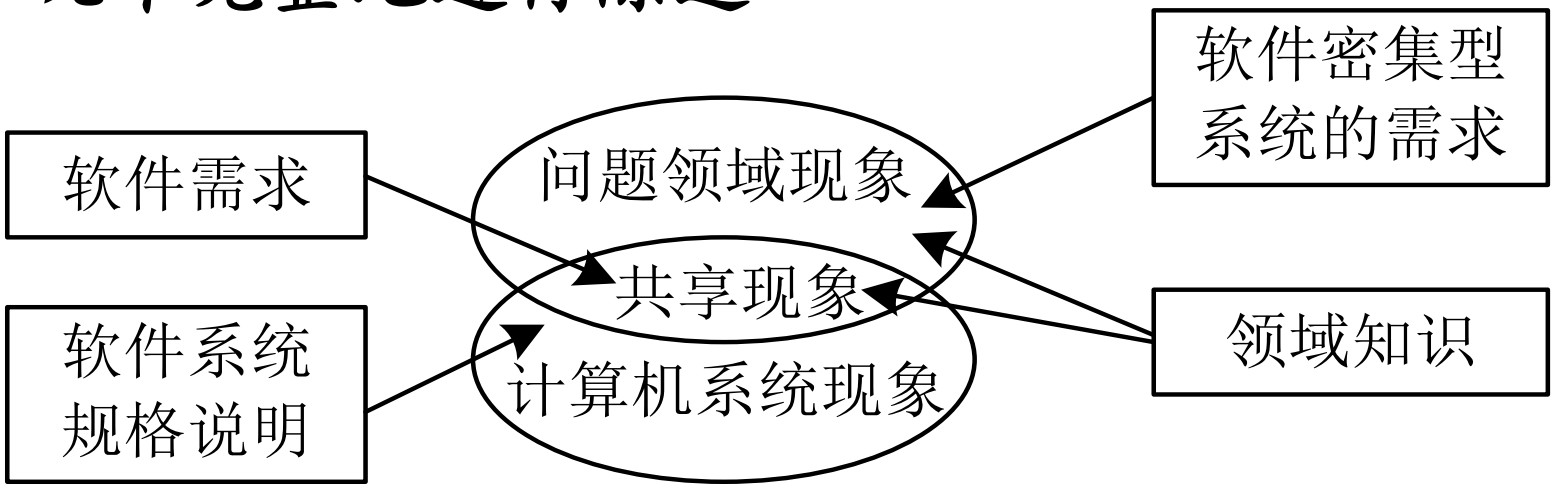
- **信息系统**
  - 对信息或数据进行收集、存储、转换、传输和/或处理，其目标是在适当的时间和地点为用户（人或其他系统）提供他们所需要的信息信
  - 主要由运行在通用计算机上的软件组成
  - 典型实例：银行帐务系统
- **嵌入式软件密集型系统**
  - 软件与硬件之间密切地集成在一起，即待实现系统中的硬件和软件部分之间经常存在复杂交互
  - 软件仅仅是嵌入式软件密集型系统一部分，但却经常是支撑创新性功能和质量特性的重要组成
  - 典型实例：汽车的防抱死制动系统（ABS）

软件密集型系统组成元素

- 硬件设备：软件系统运行所依赖的通用服务器和客户机，以及各种专用的输入/输出设备、通信互联设备、控制设备及其他电子和机械设备等
- 基础软件：支撑应用软件运行的操作系统、中间件（如消息中间件、构件容器）等基础软件
- 数据库：提供持久化信息存储和管理服务的数据库
- 网络基础设施：支持分布式系统通信的骨干网、城域网、局域网等网络基础设施
- 人员：使用和操作系统功能的各种用户
- 制度和规程：定义与系统相关的各种业务处理规则（包括非软件实现的那部分业务）的制度和规程

与问题领域的关系

- 软件密集型系统的需求涉及问题领域中的事件或状态以及这些事件或状态必须满足的条件，这些需求可以在不考虑计算机系统的情况下完整地进行陈述



校园一卡通系统案例

系统需求

需求：卡内余额低于10元时不能消费

方案1：系统自动检查余额并在低于10元时拒绝消费

方案2：工作人员查看卡内余额后发现低于10元则拒绝消费

软件需求

需求：刷卡后如果余额低于10元则拒绝消费请求，同时在屏幕上显示余额不足

软件规格说明

进一步明确软件解决方案的结构和性质，其中会涉及计算机系统的内部现象（例如数据流图）

从系统需求到软件需求

- 软件需求工程开始于一个针对整个系统的目标和愿景，而目的是得到完整、准确的软件需求定义
- 相关涉众需要通过系统工程（系统需求分析和设计）得到系统的总体规划方案
  - 组成系统的软件、硬件、网络、人员等组件和元素
  - 各个组件和元素之间的功能和责任分配
  - 不同组件或元素间的接口及相关的质量约束等
- 因此，软件需求很大程度上取决于不同系统组件和元素之间的功能和责任分配

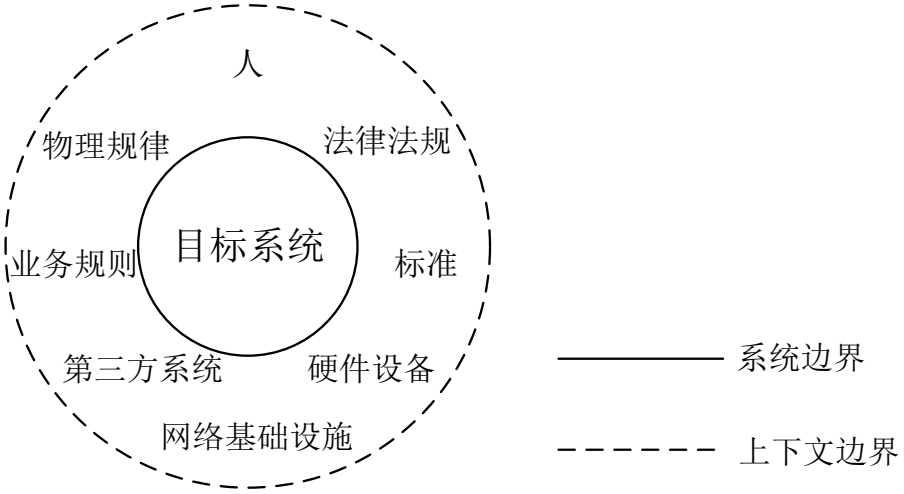
校园一卡通系统案例

系统需求：在多校区之间实现统一的账务管理

校园网络		一卡通管理软件
实现多校区之间校园网络的全连通	→	无需特殊考虑
实现多校区之间校园网络的全连通	→	多校区数据定时同步与合并功能

系统上下文（CONTEXT）

- **始终将软件需求置于系统上下文环境之中进行认识和理解**
  - 系统边界内：将由待开发的解决方案来实现的系统元素，开发团队可以自主规划并构造解决方案
  - 系统边界外（上下文）：开发团队只能做出相应的假设并被动地接受和适应



系统边界的划分

- **边界划分与待开发产品或项目的范围相关**
- **例如，对于一个中学教学管理信息化系统**
  - 如果要求开发团队在学校已有的网络及硬件设施基础上建设该系统，那么学校的网络和硬件基础设施属于系统边界之外的上下文
  - 如果要求开发团队提供一个包含网络和硬件在内的信息化整体建设方案，那么网络和硬件基础设施将位于系统边界之内，开发团队可以对其提出自己的解决方案

上下文与软件需求分析

- 对于软件需求的认识建立在对于上下文要素的一系列假设基础上
- 因此，全面、准确地识别并分析相关的上下文要素对于准确、完整地定义软件需求是至关重要的

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

隐含的上下文假设？

有什么问题？

如何解决？

问题框架（Problem Frame）

- **由Michael Jackson提出的一种软件开发问题结构化分析方法**
  - 研究需求工程必须关注软件问题，关注软件系统所处的环境
  - 强调对上下文环境的刻画，将需求的含义指称到环境的描述上
  - 认为软件问题的解决就是要得到一个能够在给定的环境约束下、满足特定用户需求的软件解决方案
- **包含五种基本的问题框架和四种变体模式**
- **通过复用基本问题框架及变体框架中的软件问题求解模式得到软件问题的解决方案**

问题分析

需求工程原则：直接关注目标软件系统要解决的软件问题，而不是直接进入软件问题解决方案的设计！

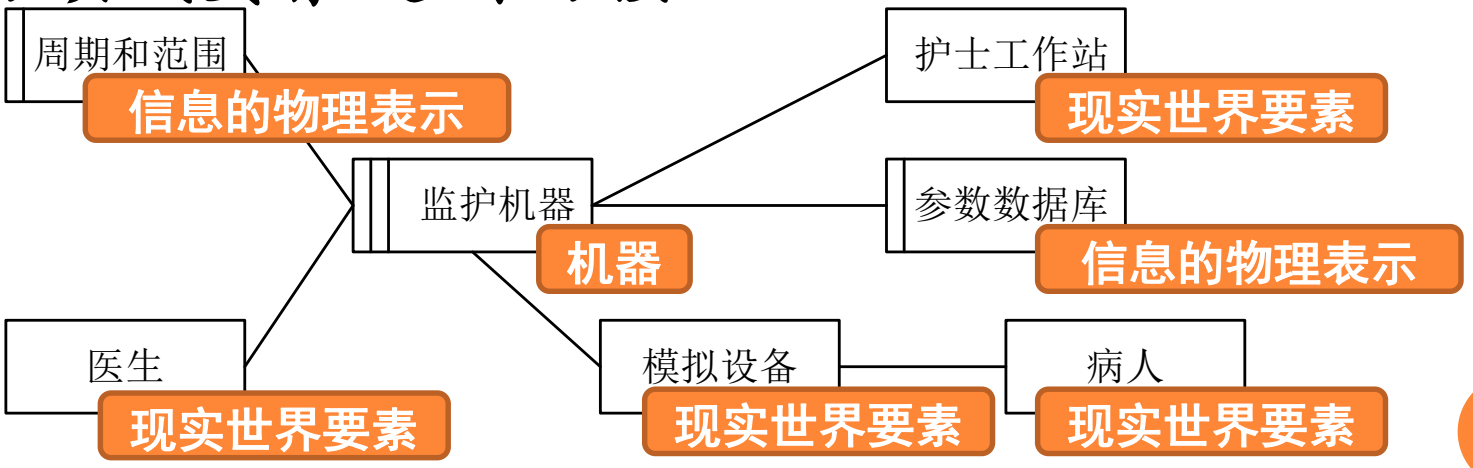
病人监护问题

软件开发人员面临的问题是开发出一个病人监护软件系统。每个病人由一个模拟设备来来监护，模拟设备用来测量病人的脉搏、体温、血压、表面阻力等体征参数。该软件按照医生指定的周期读取模拟设备所获取的数值，并将这些数据存储在数据库中。对每个病人而言，医生指定了每一个监控参数的范围。如果监控到的某个参数超出病人的体征参数超出了安全范围或发生模拟设备失效，则通知护士工作站。

- **为了解决该问题，问题框架方法深入问题所关联的现实世界，考虑如下问题：**
  - 所有的病人都要被监护，还是仅有一部分病人要监护？
  - 所有病人的安全参数都相同，还是每一个病人都有不同的安全参数？
  - 是由医生指定周期和范围，或由其他人指定这些参数？

上下文图（CONTEXT DIAGRAM）

- 分析软件问题时问题框架方法首先关注“问题在哪里”，深入问题所关联的现实世界
- 上下文图展现了用于解决软件问题的计算机系统所处的环境，描述计算机系统与其上下文中的领域要素之间的接口

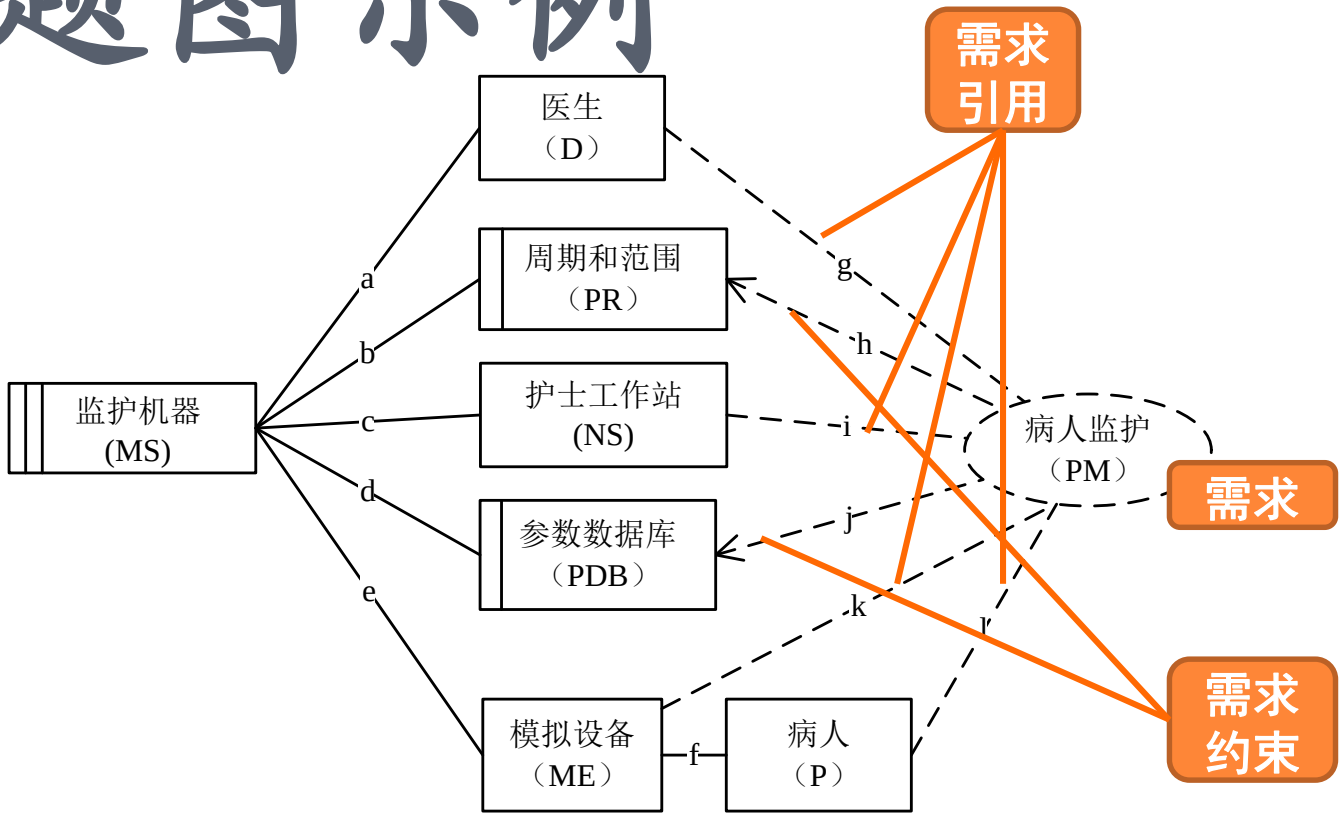


连线（接口）：各个领域要素之间共享事件、状态或数值

问题图（Problem Diagram）

- 问题图描述了如何把用户需求、问题所处的环境、问题的解决方案联系起来
- 问题图引入用以表示需求的图元，界定了需求与物理领域之间的关系
  - 需求引用：描述需求对物理领域的领域现象的引用
  - 需求约束：描述需求对于与之关联的领域现象或行为的期望和要求
- 问题图还精化了领域要素间接口上共享现象的描述，指出了领域对共享现象的控制关系

问题图示例



a: D!{EnterPatientName, EnterPeriod, EnterRange, EnterFactor}
b: PR!{PatientName, Range, Period, Factor}　　c:MS! {Notify}
d! MS{SetPatientName, SetRange, SetPeriod, SetFactor}
e: ME! {RegisterValue}　　f: P!{FactorEvidence}
g: D!{EnterPatientName, EnterPeroid, EnterRange, EnterFactor}
h: PR!{PatientName, Range, Period, Factor}　　i: NS!{NotifyInfo}
j: PDB!{PatientNameValue, PeriodValue, RangeValue, FactorValue}
k: ME!{RegisterValue}　　l: P!{FactorEvidence}

领域控制的现象

问题分解

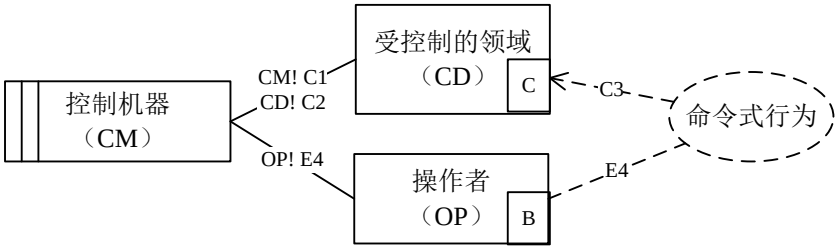
- **问题框架方法通过问题分解和模式匹配来控制问题的规模和复杂性**
  - 利用已有的分析经验（即问题框架）分析待解决的问题
  - 通过对软件问题中领域特性、接口特征等方面的分析，利用匹配基本问题框架方式把面临的问题分解为多个子问题
  - 其中每一个子问题匹配一个基本问题框架或变体框架
  - 在此基础上复用问题框架的框架关注点，诱导出问题的解决方案，同时识别出解决问题时候可能面临的困难（用特定的关注点表示）

五个基本问题框架

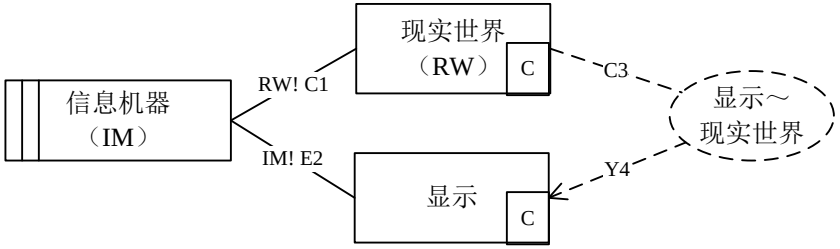
- 需求式行为框架：物理世界的某个部分的行为需要受到控制以满足特定的约束，问题是建立一个机器来施行所需要的控制
- 命令式行为框架：物理世界的某个部分的行为需要按照操作者的命令来控制，问题是建立一个机器来接收操作者的命令并施加相应的行为
- 信息显示框架：物理世界的某个部分的状态或行为的信息需要被显示出来，问题是建立一个机器从物理世界获取信息然后按照要求的格式进行显示
- 工件框架：需要一个工具来支持用户创建或编辑特定类型的、计算机可处理的问题或图形对象
- 变换框架： 存在一个计算机可读的输入文件，其数据必须按照特定的格式变换成要求的输出文件

五个基本问题框架

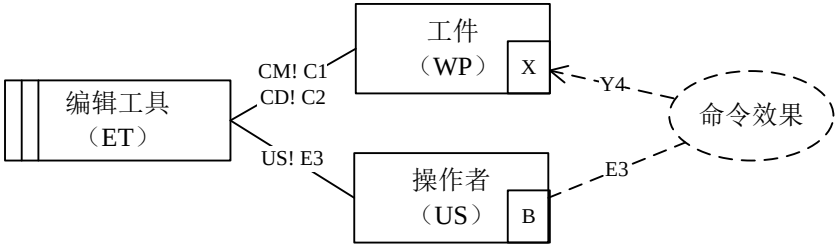
（1）需求式行为框架



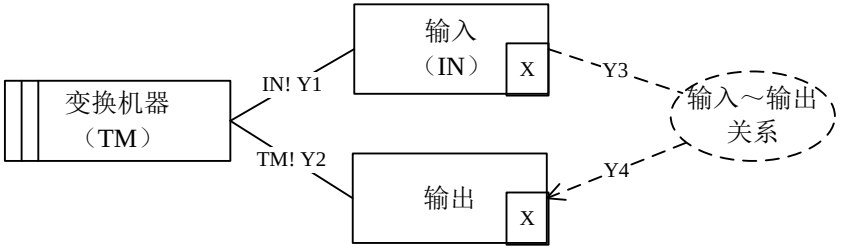
（2）命令式行为框架



（3）信息显示框架



（4）工件框架



（5）变换框架

四个变体框架

- 现实世界中的软件问题经常会发生分解后的子问题难以匹配某个基本问题框架
- 问题框架方法使用框架变体来扩展基本问题框架
  - 描述变体：通过引入描述领域来修改基本问题框架
  - 操作者变体：将操作者引入基本问题框架
  - 连接变体：通过在机器和问题领域之间引入连接领域 来拓展基本问题框架
  - 控制变体：不引入任何新的领域，而通过改变接口现象的控制特性来改变基本问题框架

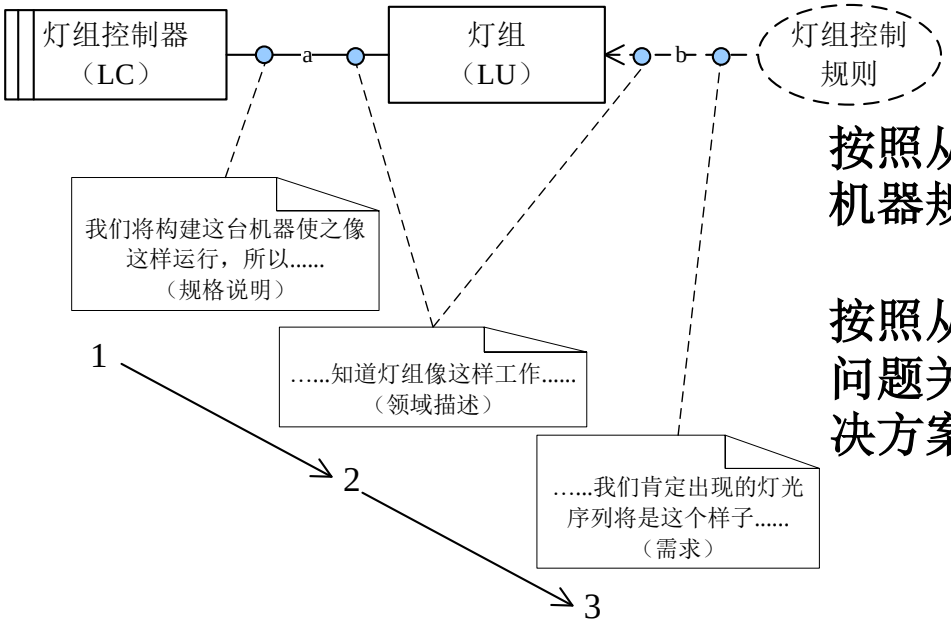
问题框架图

- 表示问题模式的结构，使用框架关注点表示一个问题模式的分析思路，利用特定的框架关注点表示在问题类求解时候可能面临的困难
- 进一步将问题图中的领域按照领域特征划分
  - 因果领域：典型的物理领域，其领域现象之间存在可预测的因果关系
  - 可叫牌领域：通常由人组成，它是物理领域，但是其领域现象之间没有可预测的因果关系
  - 词法领域：数据的物理表示，涉及符号现象

框架关注点

a: LC!{Rpulse[i], Gpulse[C1]} [C1]
b: LU!{Stop[i], GO[i]} [C3]

单行线交通灯控制问题



按照从左到右的顺序分析：可以确认给定的机器规约是否能够确保需求的满足

按照从右到左的顺序分析：澄清用户需求、问题关联的环境约束，为机器规约及问题解决方案的求解提供了启发与指导

单行线交通灯问题的框架关注点

内容提要

- 需求工程概述
- 系统与上下文分析
- 需求制品
- 需求工程活动

三种主要的需求制品

- 目标：在一定的抽象程度上刻画了涉众对于待开发软件系统的期望和意图，用于在早期阶段对需求进行建模
- 场景：通过实例化的方式描述了系统在使用和维护等过程中的典型交互序列
- 面向方案的需求：以面向解决方案的方式刻画了待开发系统在数据、功能和行为等方面的属性

目标和目标模型

- **功能性目标和非功能性目标**
  - 功能性目标描述了涉众期望软件系统所提供的服务，如导航系统应该有定位功能
  - 非功能性目标描述了涉众对于这些服务的质量要求，如定位应该具有较高的准确性
- **目标模型**
  - 系统地描述了涉众的需求及其精化和依赖关系
  - 同时捕捉了满足系统目标的多种实现方式以及相应的质量关注

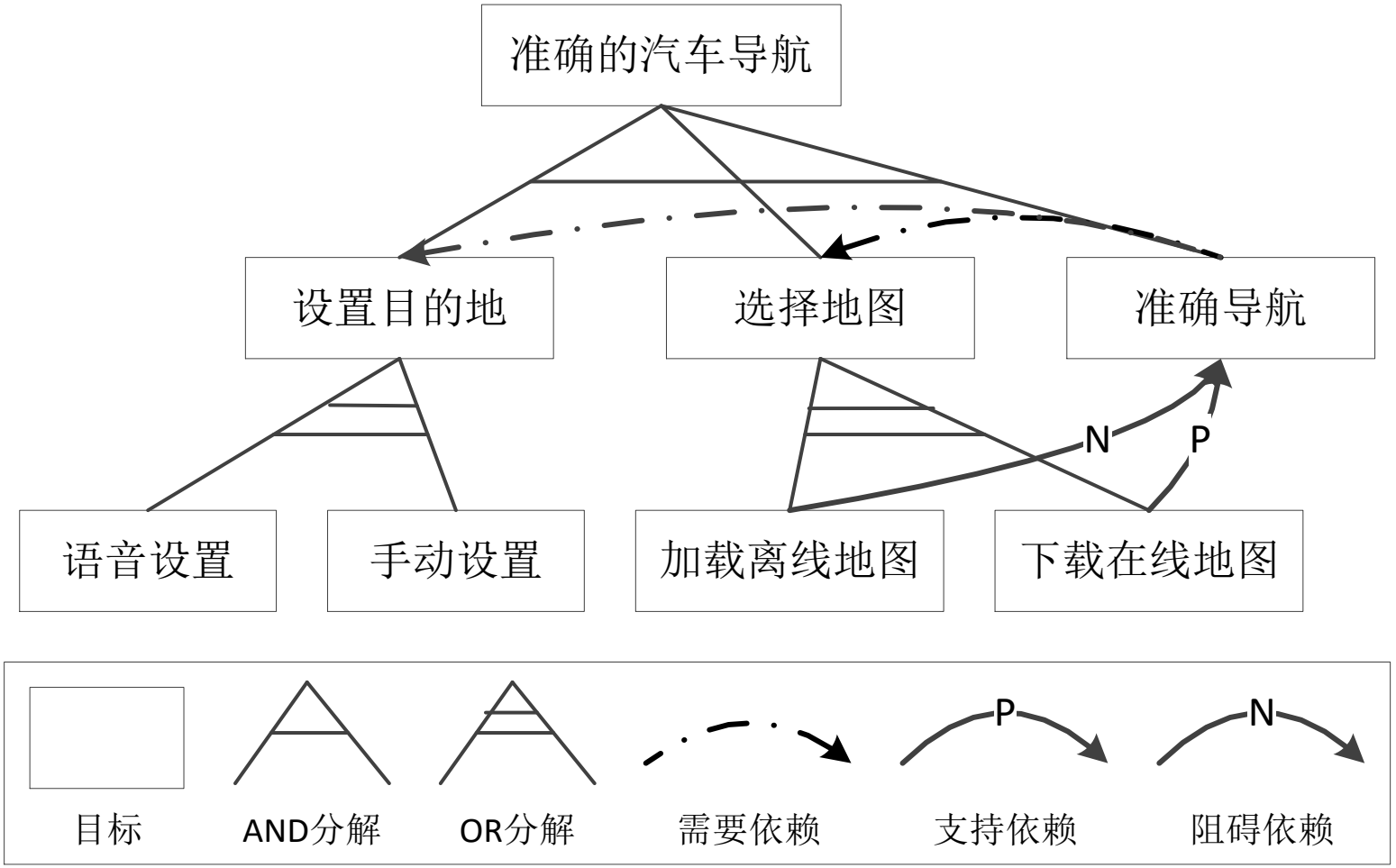
基于AND/OR分解的目标建模

- 主要关注于目标精化和依赖关系的描述
- 目标精化（目标分解）：把相对抽象的高层目标通过子目标分解得到相对具体的底层目标的过程
- 目标精化类型
  - 目标的AND分解：如果父目标G被AND分解为子目标G1、G2、……、Gn，那么必须所有的子目标得到满足才能使得父目标得到满足
  - 目标的OR分解：如果父目标G被OR分解为子目标G1、G2、……、Gn，那么只要任意一个子目标得到满足就能使得父目标得到满足

目标依赖

- 需要：如果目标G1需要依赖于目标G2，那么目标G2得到满足是目标G1得到满足的前提
- 支持：如果目标G1支持依赖于目标G2，那么目标G1得到（部分）满足能够使得目标G2得到（部分）满足
- 阻碍：如果目标G1阻碍依赖于目标G2，那么目标G1得到（部分）满足能够使得目标G2（部分）不满足
- 冲突：如果目标G1冲突依赖于目标G2，那么目标G1得到满足将使得目标G2不满足，或者目标G2得到满足将使得目标G1不满足

基于AND/OR分解的目标建模示例



面向主体的目标建模框架I*

- **主要关注于参与者之间的依赖关系以及参与者的内部结构描述，包含两类模型**
  - 策略依赖模型：刻画参与者之间的协作关系
  - 策略原理模型：刻画参与者的内部原理结构
- **参与者可以是与待开发软件系统相关的人、组织或者系统**
- **四种策略依赖关系：目标依赖、任务依赖、资源依赖和软目标依赖**
  - 依赖者不关心也不指定被依赖者如何实现特定的目标、完成特定的任务、提供特定的资源或者实现特定的软目标
  - 被依赖者也可以自由地选择适当的方式来实现特定的目标、完成特定的任务、提供特定的资源或者实现特定的软目标

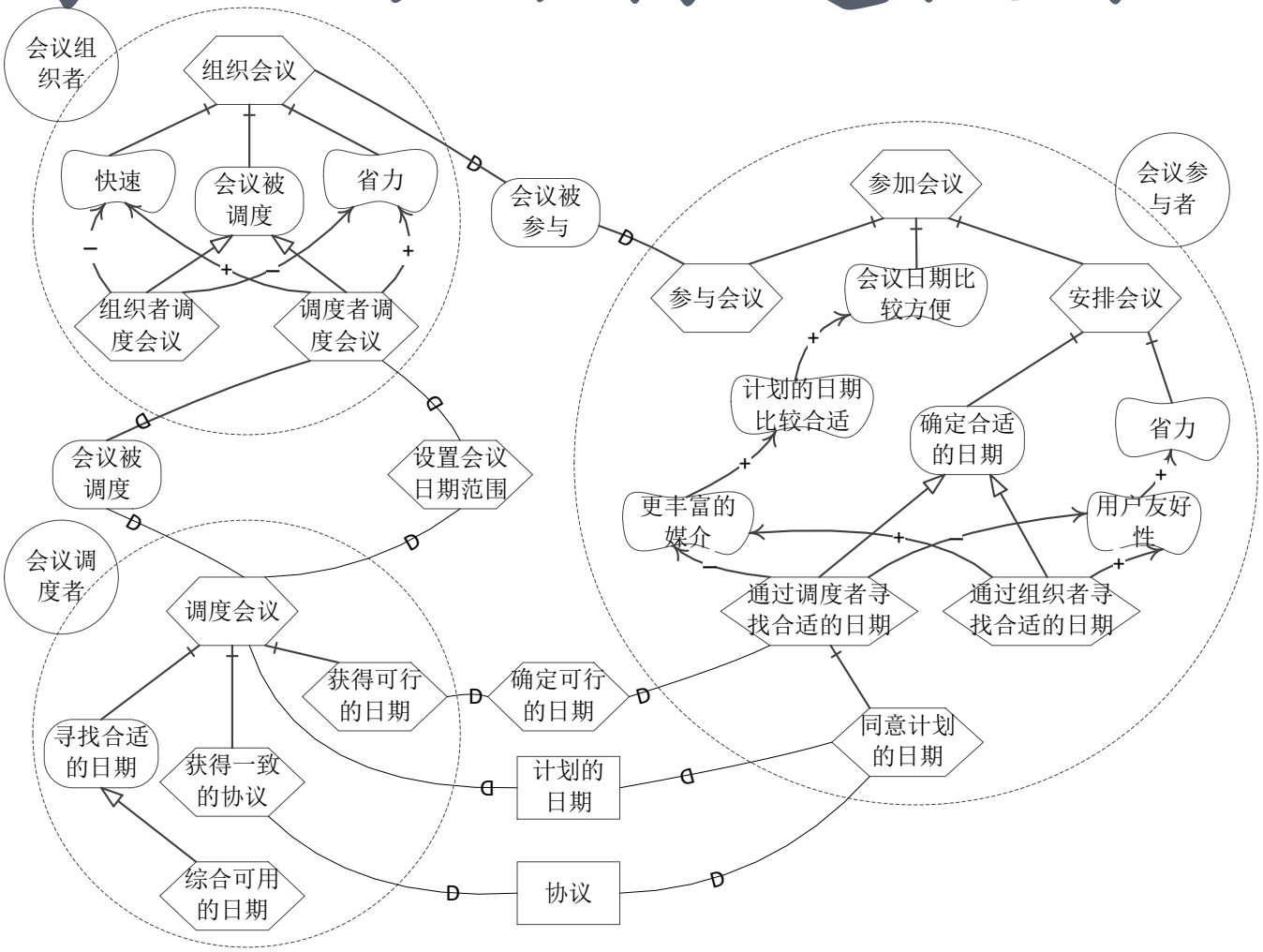
四种策略依赖关系

- 目标依赖：一个参与者依赖于另一个参与者来实现特定的（硬）目标
- 任务依赖：一个参与者依赖于另一个参与者来完成特定的任务
- 资源依赖：一个参与者依赖于另一个参与者来提供特定的物理或者信息资源
- 软目标依赖：一个参与者依赖于另一个参与者完成某个实现特定软目标的任务

参与者的原理结构关系

- 目的—手段链接：一个目标和一个实现这个目标的任务之间的关系（可以描述实现一个目标的多种可选方案）
- 任务分解链接：一个任务和完成这个任务的组成部分之间的关系（子目标、子任务、资源或者软目标的任意组合）
- 贡献链接：一个目标、一个任务、一个资源或者一个软目标对实现另一个软目标的影响程度(正面影响、负面影响和未知影响)

基于I*的目标建模示例


会议组织者
组织会议
快速
会议被调度
省力
组织者调度会议
调度者调度会议
会议被参与
参加会议
会议参与者
参与会议
会议日期比较方便
安排会议
计划的日期比较合适
确定合适的日期
省力
更丰富的媒介
用户友好性
通过调度者寻找合适的日期
通过组织者寻找合适的日期
会议被调度
设置会议日期范围
会议调度者
调度会议
获得可行的日期
确定可行的日期
寻找合适的日期
获得一致的协议
计划的日期
同意计划的日期
综合可用的日期
协议
目标
任务
资源
软目标
目的—手段链接
任务分解链接
贡献链接
目标依赖
任务依赖
资源依赖
参与者

基于KAOS框架的目标建模

- 主要关注于通过目标的形式化规约来进行目标模型的半形式化和形式化的推理分析
- 提供了两种层次的目标建模方法
  - 基于图形的方法
  - 基于一阶时序逻辑的方法

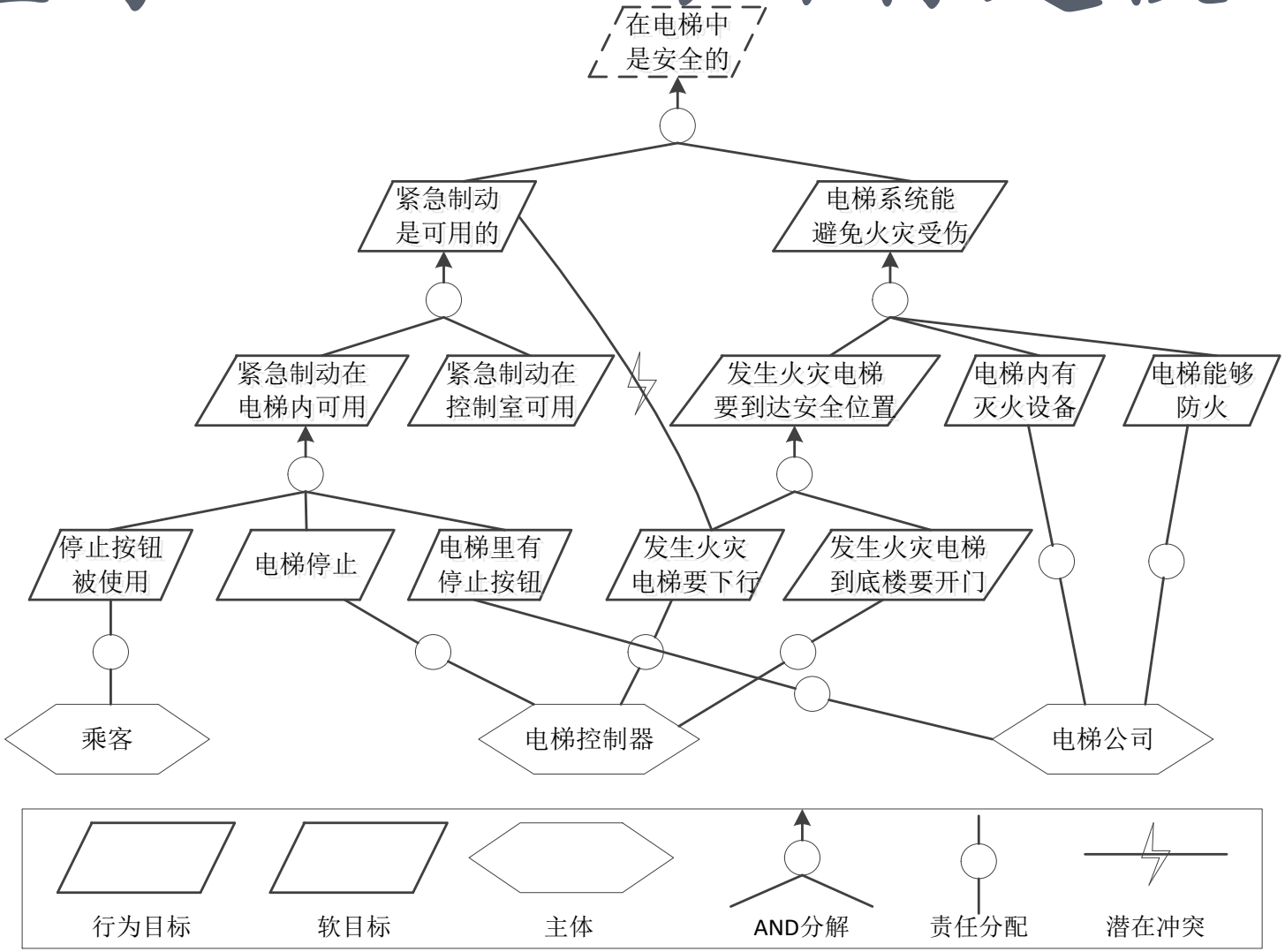
KAOS的对象元素

- 行为目标：描述了一组期望的、所允许的系统行为，用来推导系统的操作
  - 达成目标（要求所定义的性质最终必须成立）
  - 维持目标（要求所定义的性质一直保持成立）
  - 避免目标（要求所定义的性质一直不成立）
- 软目标：描述了一个期望系统满足的质量属性，用来比较多个可选的系统行为
- 操作：描述了系统状态的转变，由主体来执行
- 主体：一个人、一种设备或者是一个系统组件

KAOS的关系元素

- AND分解：如果一个父目标被AND分解为多个子目标，那么必须所有的子目标都得到满足才能使得父目标得到满足
- OR分解：通过将多个AND分解关联到同一个父目标来实现的，其中每一个AND分解表示满足这个父目标的一个可选方案，只要其中任意一个得到满足就能使得父目标得到满足
- 潜在冲突链接：一个目标与另一个目标之间的潜在冲突链接表明满足一个目标在一定条件下会阻碍另一个目标的满足
- 操作化链接：一个操作化链接将一个目标和一个操作关联起来，表明了为了实现这个目标需要执行这个操作
- 责任分配链接：一个责任分配链接将一个目标和一个主体关联起来，表明了这个主体有实现这个目标的责任

基于KAOS的目标建模示例



场景与用况

- **通过实例化的方式描述了系统在使用和维护等过程中的典型交互序列**
  - 有利于涉众间的需求交流
  - 可以在多个抽象层次上以不同的方式进行描述，因此有利于作为软件需求的中间层抽象
- **用况将相关的场景组织在一起，包括**
  - 一个主场景
  - 若干可替换场景
  - 若干例外场景组成

场景与目标的关系

- **目标以一种抽象的方式刻画了相关涉众的期望和意图**
  - 适合于在高层把握系统的总体需求并进行自顶向下的需求精化
  - 但因缺少细节而不利于在涉众间进行需求的交流
- **场景以实例化的方式从外部用户的视角描述了与系统之间的交互过程**
  - 场景描述了一个目标（或者一组目标）被满足或者未被满足的一个具体实例，提供了一个或多个目标的具体细节
  - 场景通常定义了一系列为满足目标而执行的交互步骤，并将这些交互步骤与系统上下文联系了起

场景与目标的关系

驾驶辅助系统包含一个防追尾（子）系统。这个系统包含（①）距离传感器，这个传感器（②）一直检测本车与前车的距离（③）以避免追尾。（④）如果系统发现与前车距离低于安全距离但仍然在临界范围之外，就会发出声音警报，（⑤）或者在汽车驾驶舱中的驾驶员显示器上显示相应的信号或信息。如果驾驶员在2秒内没有做出反应并且与前车距离持续拉近，（⑥）那么系统会降低车速。（⑦）如果距离（单位：米）在任何时刻降低到车速（单位：千米每小时）的四分之一，那么系统将启动紧急刹车。

解释：

（①）静态/结构性方面
（②）功能性方面
（③）解释性方面
（④）行为性方面
（⑤）探索性方面
（⑥）可替换场景的步骤
（⑦）例外场景的条件

将参与者、上下文信息与系统的功能和行为等关联起来

场景描述的抽象层次

- **用况和场景的一个突出特点是可以在多种不同的抽象层次上进行描述**
  - 非常接近于现实世界的描述方式，例如通过录像记录的用户场景
  - 以接近现实世界的方式但去除一些无关细节之后的描述方式，例如通过动画描述的用户场景
  - 通过自然语言“讲述”的用户场景
  - 抽象的概念模型描述，例如使用UML用况图、顺序图和活动图建立的用况和场景模型
- **适合用于需求获取和分析过程中**
  - 作为反映现实世界的具体需求信息到抽象的基于概念模型的需求描述的中间层和过渡
  - 例如，可以首先通过录像、动画或自然语言描述与用户交流需求场景，然后在此基础上通过抽象获得场景的概念模型描述，并得到关于目标以及面向方案的需求等其他抽象需求。

场景分类

- 当前状态场景和期望状态场景
- 正面场景、负面场景和不当使用场景
- 描述性场景、探索性场景和解释性场景
- 实例场景、类型化场景和混合场景
- 交互场景、上下文场景和系统内部场景

实例/类型/混合场景-1

- **实例场景（用户故事、具体场景）**
  - 描述了具体参与者之间具体的（现有的或想象的）交互序列，其中的输入和输出信息也是在实例层次上描述的
  - 例如，描述选课过程的实例场景会涉及具体的参与者和输入、输出信息，如“计算机系三年级本科生李磊”、“软件工程课程”、“主校区2101教室”等
- **类型化场景**
  - 对于特定交互序列中具体的参与者、输入和输出进行了抽象。类型化场景中使用的是活动者类型（例如人的角色），其中的输入和输出信息也是在类型层次上描述的
  - 例如，描述选课过程的类型化场景会使用抽象的参与者角色和输入、输出信息类型，如“学生”、“课程”、“教室”等

实例/类型/混合场景-2

- **混合场景**
  - 既包含实例层次的交互信息又包含类型层次的交互信息
  - 混合场景在实践中使用的较为广泛，一般可以遵循下列原则
    - ✓场景的重要内容应该在实例层次上详细描述
    - ✓未完全理解的内容在实例层次上描述，避免由于早期抽象时一些未完全理解的方面导致的错误
    - ✓冲突或可能发生冲突的内容在实例层次描述，以支持涉众间的交流和冲突解决

交互/上下文/内部场景-1

- **交互场景描述了系统边界上的交互，即系统和外部参与者（人或系统）之间的交互**
  - 以一种面向使用的方式描述了系统如何嵌入到上下文之中发挥作用
  - 例如，选课系统中的选课场景是描述用户（学生）与系统之间交互序列的交互场景
- **上下文场景描述了与系统使用或系统自身相关的附加上下文信息，例如参与者以及系统的非直接用户之间的交互**
  - 虽然可能完全是系统边界之外的交互过程，但其中的上下文信息可以为系统的需求定义提供附加的说明
  - 例如，对于选课系统，描述任课教师通过填写打印好的纸质登分表向教务员提交课程成绩的场景属于上下文场景，但该场景有助于相关涉众理解系统为教务员提供的课程成绩录入场景（该场景属于交互场景）中的成绩来源

交互/上下文/内部场景-2

- **系统内部场景或场景片断描述了系统如何通过系统内部交互序列实现一些与系统上下文的交互**
  - 只关注系统内部的交互，即发生在系统边界之内的交互
  - 描述了系统决定对于外部事件（来自上下文的输入）或时间事件的响应（对上下文的输出）的内部处理和决策过程
  - 通常在体系结构设计过程中使用
  - 例如，针对选课系统中选课过程的一个内部场景可以描述系统内部选课子系统、学费管理子系统、成绩管理子系统等子系统之间的交互过程

场景描述-1

- **叙述性的场景描述：使用自然语言，以叙述性的方式描述场景的发生过程**
  - 优势：容易使用、容易理解，可以灵活调整不同部分描述的详细程度并将不同类型的场景信息结合起来
  - 不足：描述随意不规范、遗漏重要的场景信息、交互步骤及步骤间关系不明确等
- **基于结构化模板的场景描述：一般以用况为单位，使用基于模板的自然语言描述场景及相关的其他信息**
  - 给出了场景描述所要求的各个描述项及其描述方式或要求
  - 在保持自然语言的易读性和易理解性基础上进一步提高了场景描述的规范化和结构化程度

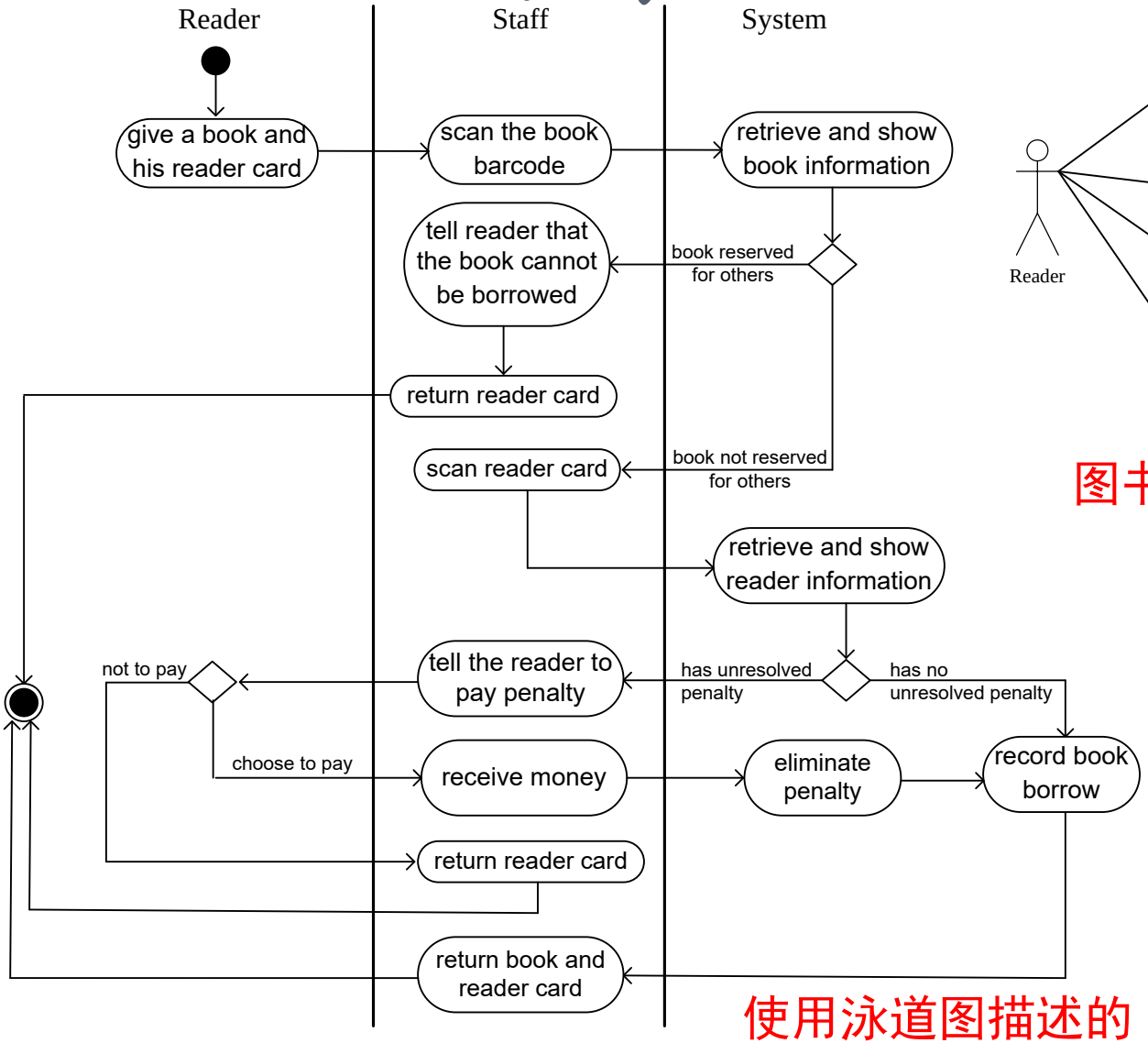
场景描述-2

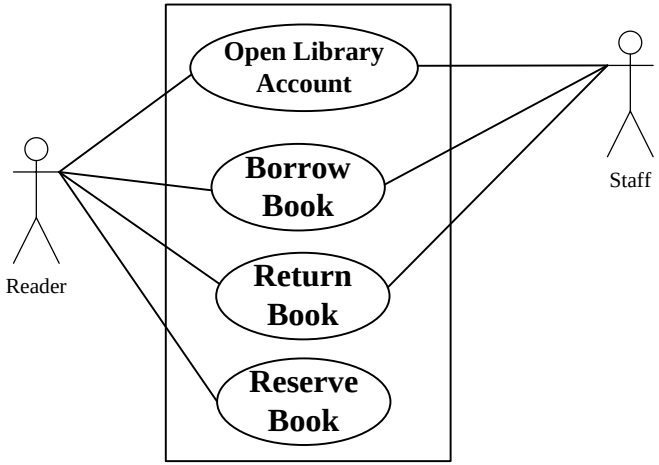
- **基于UML模型的场景描述：通常结合使用UML用况图、顺序图及活动图**
  - 用况图描述一个系统中所涉及的外部参与者、所包含的用况、参与者与用况以及用况之间的关系
  - 单个用况的交互序列则可以使用顺序图或活动图（或其变体泳道图）描述
    - ✓UML顺序图强调的是系统与一系列用户随着时间的交互序列
    - ✓活动图则侧重于描述在多个场景（例如属于同一用况的场景）之间的控制流

场景描述模板示例

描述项	说 明
用况名称	简短、易理解的用况名称，一般以动词开头。
参与者	参与当前用况交互过程的外部参与者。
涉众及关注点	与当前用况相关的涉众，以及这些涉众对于该用况不同的关注点和目标。
前置条件	当前用况可以执行之前应当满足的条件。
后置条件	当前用况成功结束后应当满足的条件。
触发事件及条件	触发当前用况执行的事件以及应当满足的条件。
发生频率	当前用况发生的频率，此属性对于与该用况相关的性能和可用性等质量设计具有较大的影响。
主场景	当前用况的（唯一）主场景，一般可以通过顺序编号的方式描述场景的每一个交互步骤。
可替换场景	当前用况的（零个或多个）可替换场景。可以以类似于主场景的方式逐一描述每个可替换场景的交互步骤，也可以在主场景基础上描述每个可替换场景的差异性部分（与主场景有区别的那些步骤）。
例外场景	当前用况的（零个或多个）例外场景。对于每个例外场景都要描述例外条件和相应的例外处理步骤。
质量需求	与当前用况执行过程相关的质量需求及其要求。例如，如果用况中包含与敏感信息传输相关的步骤，那么该用况与保密安全性相关；如果包含可能危及人身安全的步骤，那么该用况与安全性相关。
其他问题	关于当前用况所存在的待解决的问题，例如需要相关涉众进一步澄清或协商的地方。

UML场景描述示例





图书馆管理系统用况图

使用泳道图描述的“借书”用况

综合应用多种描述方式

- **场景描述通常以一种增量和演化式的方式进行**
  - 自顶向下的分解：从抽象的总体场景到细化的局部场景
  - 从黑盒到白盒场景：先描述系统与环境之间的交互以及环境中相关对象之间的交互，充分理解后再将这些黑盒场景扩展为包含系统内的子系统间交互的白盒场景
  - 从非形式化到形式化：先使用叙述性的自然语言描述，然后施加模板结构以完善描述并发现不一致及其他问题，再逐渐转换为更加形式化的描述（如顺序图）
  - 增量的场景开发：随着对需求场景的确认或基于其他领域模型进行检查，场景定义的质量可以随着知识的不断扩充和修订不断改进

面向方案的需求

- **需求工程早期的需求制品主要以目标和场景为主，然后随着对于需求问题理解的不断加深逐渐导出面向方案的需求**
  - 以面向解决方案的方式刻画了待开发系统在数据、功能和行为等方面的属性
  - 直接刻画了待开发系统的属性和特征，是软件开发人员设计和实现系统的直接依据
  - 应当描述统一的需求视图、需求定义更为详细且不包含冲突：这意味着确定面向方案的需求之前相关涉众应当针对所有已识别的需求冲突达成共识

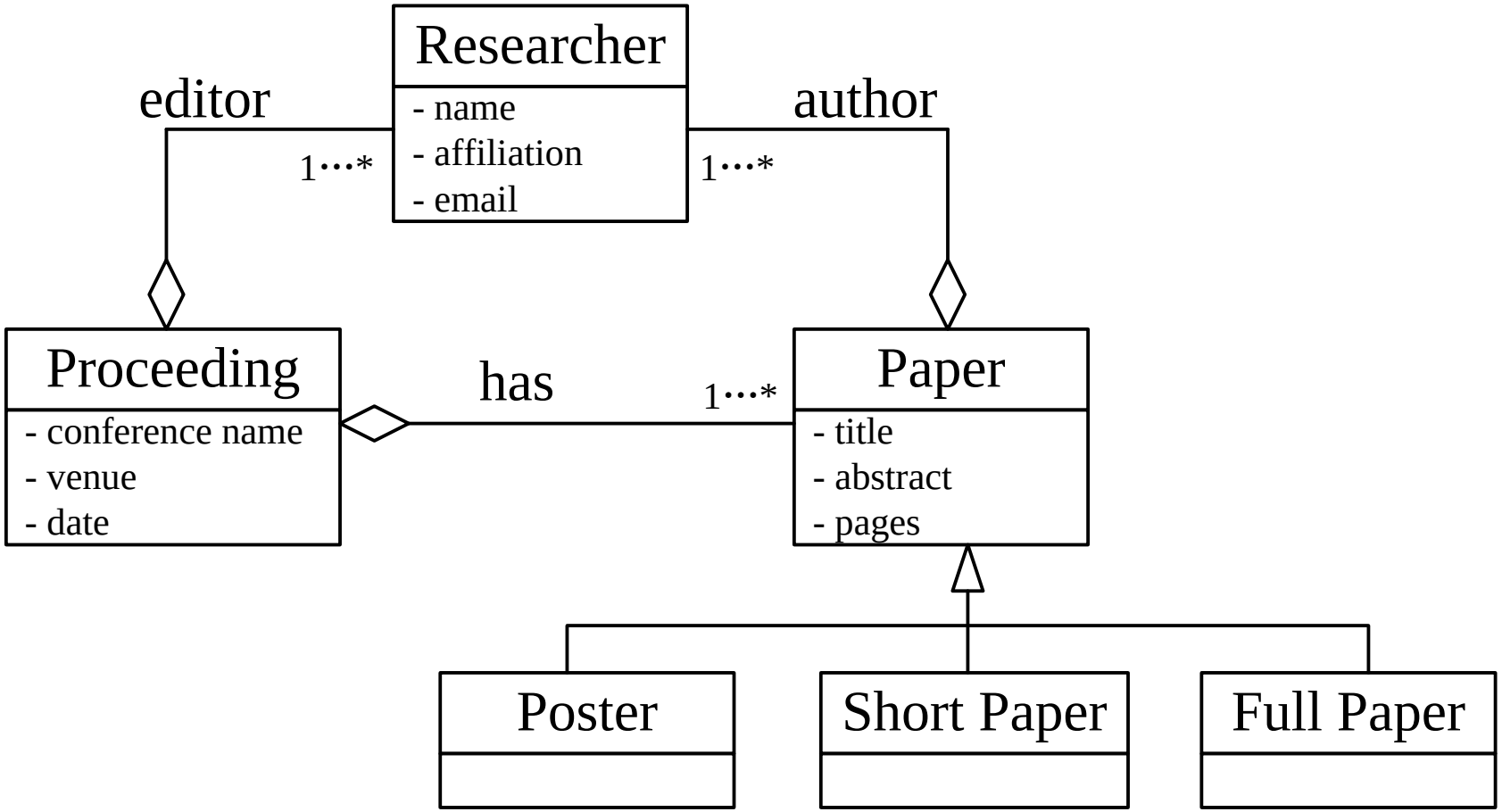
面向方案的需求视图

- **三种视图**
  - 数据视图描述与系统相关的数据和信息实体、属性及实体间关系
  - 功能视图描述系统应当提供的功能和处理过程，包括功能和处理过程间的数据输入输出关系以及相互间的精化关系
  - 行为视图描述了系统对于外部和内部激励的响应以及在此过程中的状态转换
- **三种视图从不同视角描述了待开发系统的属性和特征，但又密切相关**
  - 数据视图中定义的数据和信息实体可以作为输入输出数据出现在功能视图中
  - 功能视图中定义的功能和处理过程可以在行为视图中作为动作进行引用

数据视图

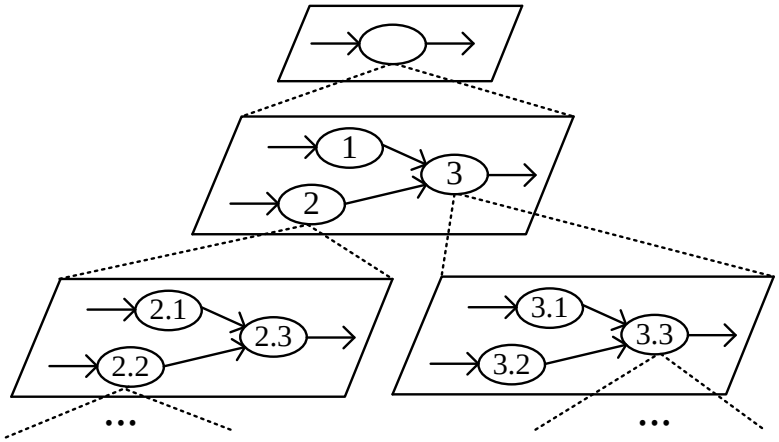
- 数据视图中的需求模型（即数据模型）是一种静态模型
- 在类型层次上描述了与待开发系统相关的数据和信息实体、实体属性及实体间关系
  - 其中的实体、属性和关系一般可以通过考虑上下文信息中的主体刻面进行识别
  - 数据模型一般可以使用实体关系模型（ER模型）或UML类模型进行描述

数据模型示例

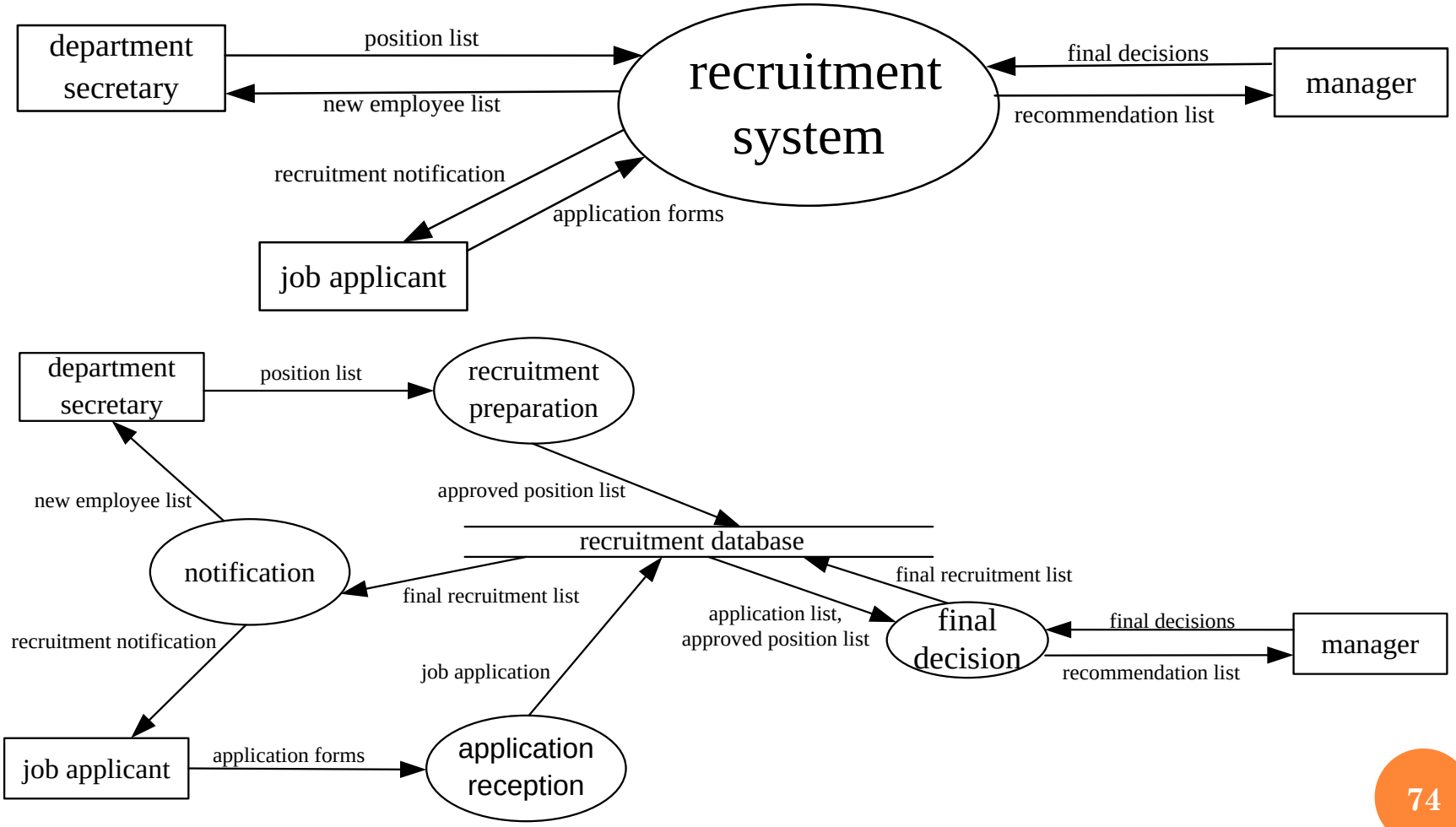

Researcher
- name
- affiliation
- email
editor
1…*
author
1…*
Proceeding
- conference name
- venue
- date
has
1…*
Paper
- title
- abstract
- pages
Poster
Short Paper
Full Paper

功能视图

- 功能视图具体描述了待开发软件系统应当提供的功能和业务处理过程
- 由于软件的功能一般可以表示为对于信息的加工和处理过程，因此功能视图通常可以使用数据流模型来描述
  - 主要元素：外部实体、加工、数据存储和数据流
  - 数据流模型以一种逐层精化的方式支持复杂系统功能视图的描述



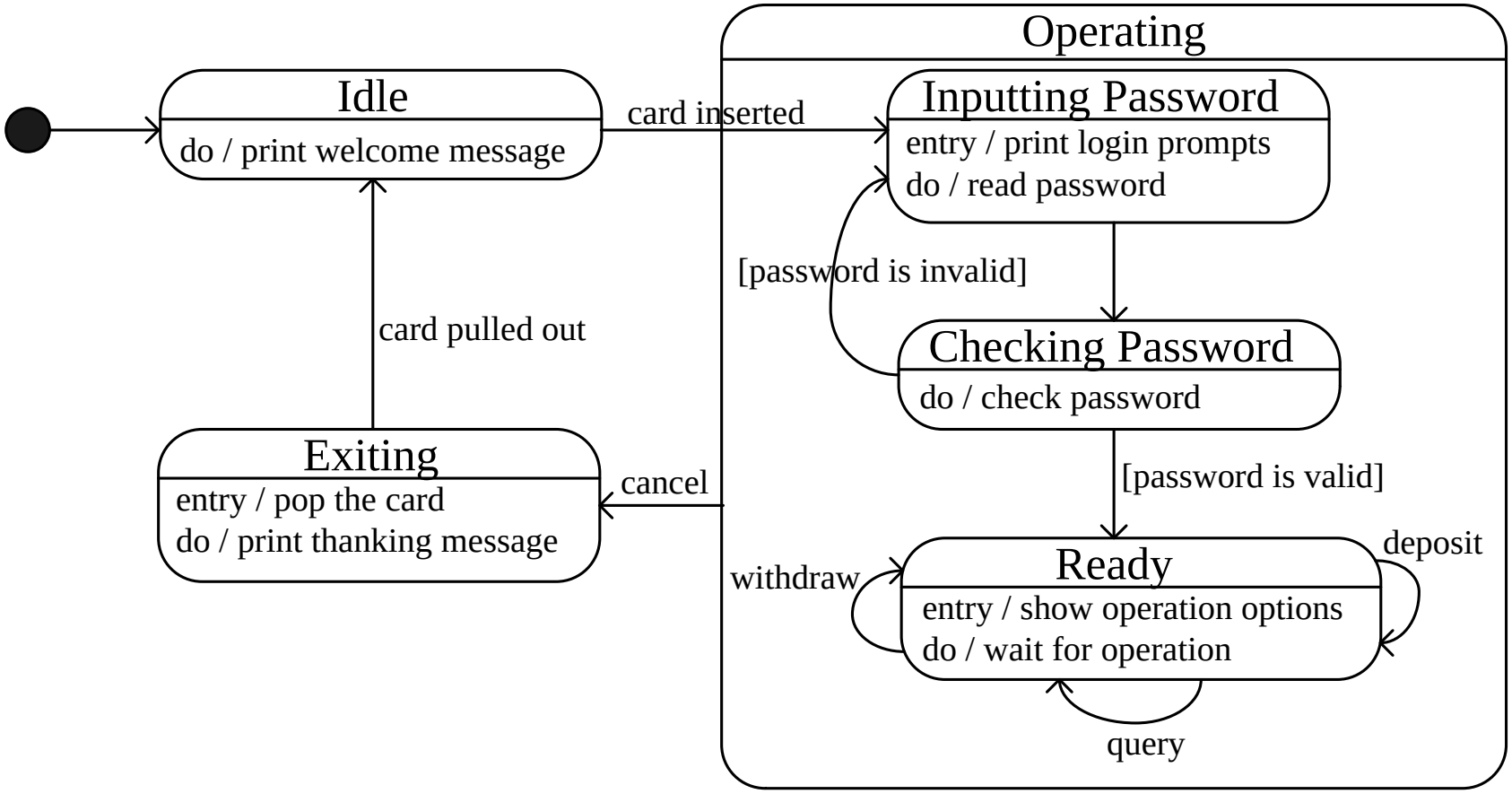
功能（数据流）模型示例


department secretary
position list
new employee list
recruitment system
final decisions
recommendation list
manager
recruitment notification
application forms
job applicant
department secretary
position list
recruitment preparation
new employee list
approved position list
recruitment database
notification
final recruitment list
final recruitment list
application list, approved position list
final decision
final decisions
recommendation list
manager
recruitment notification
job application
job applicant
application forms
application reception

行为视图

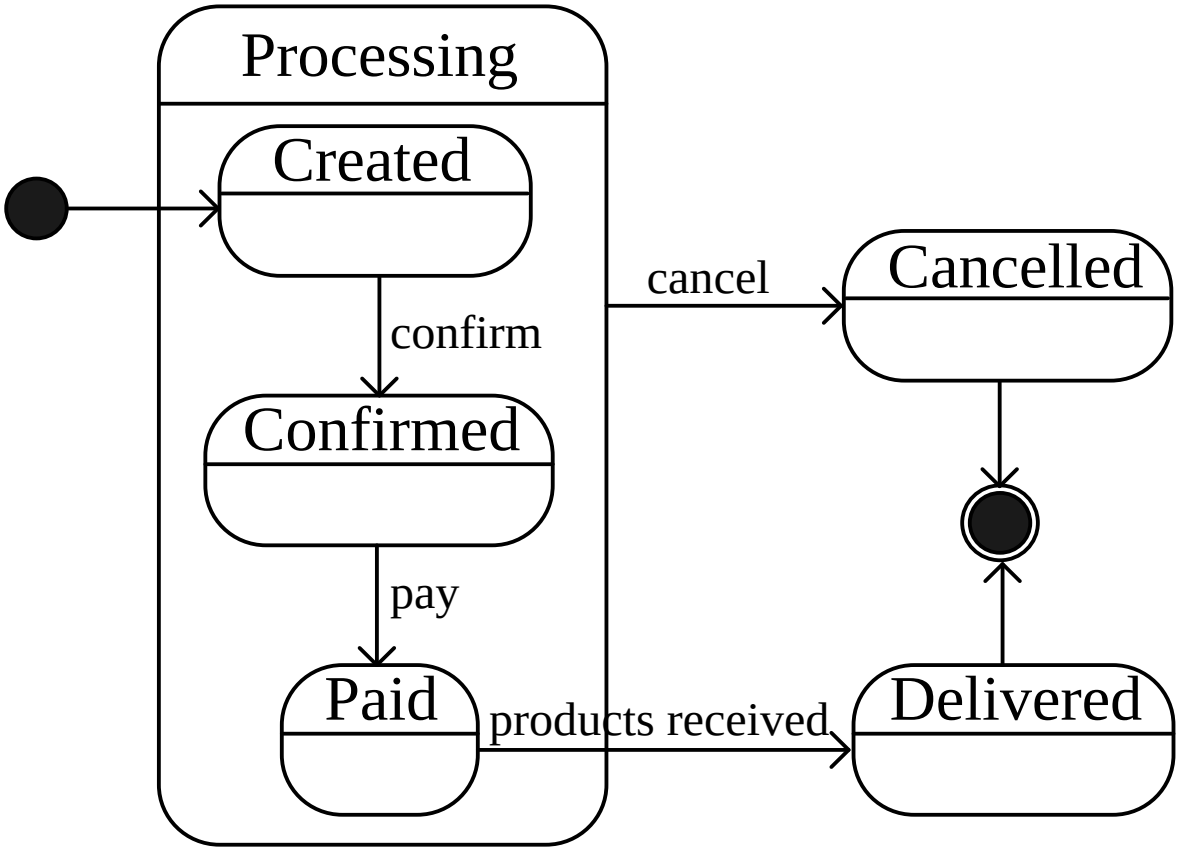
- 行为视图中的需求模型（即行为模型）反映了软件系统在外部的事件或激励作用下的反应性行为
- 具体表现为系统的状态转换以及在此过程中执行的动作等
- 可以在两个层次上进行建模
  - 系统层次上的行为模型描述系统的整体状态转换及其动作
  - 类层次上的行为模型则针对特定的分析类描述类及其对象实例的状态转换及其动作
- 可以使用有限自动机和状态图等方式来描述

行为模型示例-1



使用UML状态机图描述的银行ATM机系统行为模型

行为模型示例-2



使用UML状态机图描述的订单类（对象）行为模型

内容提要

- 需求工程概述
- 系统与上下文分析
- 需求制品
- 需求工程活动

需求工程的目的

- **通过一种协作式的、不断迭代以及增量的过程实现三个维度上的目标**
  - 内容：使得所有相关的需求都在所需要的细节层次上得到清晰的认识和理解
  - 共识：在相关涉众间建立起对于系统需求的充分共识
  - 文档化：所有需求都依照相关的格式和规范进行了文档化和规格说明描述

需求工程活动-1

- 需求获取：识别相关涉众及其他需求来源，通过各种手段从这些需求来源那里获取各种需求信息，并建立对于待开发软件所要解决的问题的初始理解
- 需求分析：确定待开发软件系统的上下文边界及其与环境的交互关系，在此基础上从系统整体需求中导出并细化软件需求，同时发现并解决需求中所存在的冲突和不一致性

需求工程活动-2

- 文档化：根据相关的文档和规格说明规范，对各项需求工程活动中所得到的需求信息进行文档化描述和记录
- 需求确认：检查需求文档是否满足相关的需求质量准则，以及是否完整、准确地反映了相关涉众的期望和要求
- 需求管理：对于需求制品和需求变更的管理，广义的需求管理还包括对于需求工程全过程的管理

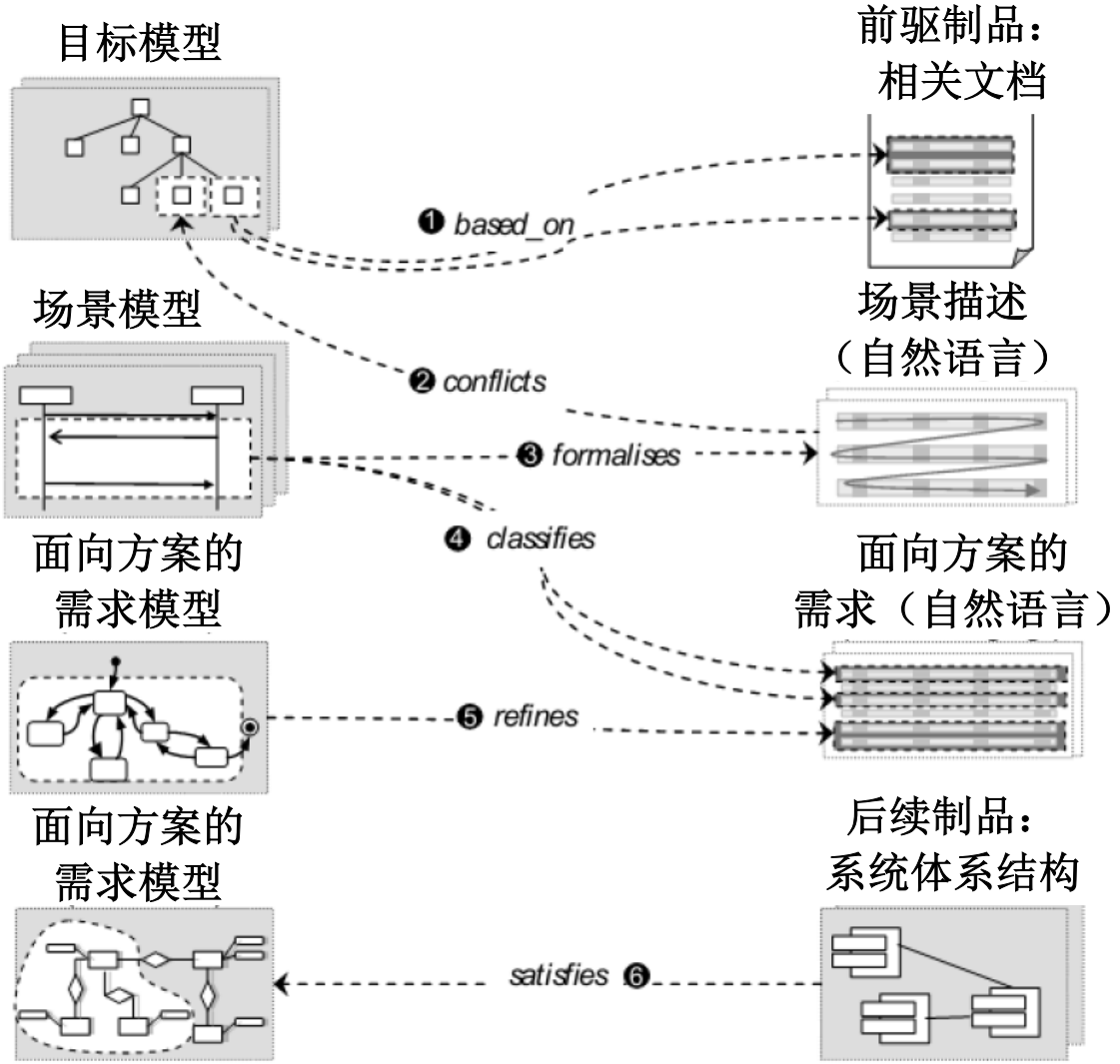
需求追踪管理

- 需求追踪（traceability）是指在前向和后向两个方向上描述并且跟踪需求的存在形式的能力
  - 从需求的来源、开发和规格说明，直到后续的部署和使用
  - 以及在这些阶段中不断精化和迭代的过程中
- 需求追踪关系还包括不同类型的需求元素之间、同种类型但不同抽象层次上的需求元素之间的追踪关系

需求追踪的意义

- 辅助理解
- 需求确认
- 变更管理
- 软件复用
- 项目管理

需求追踪关系示例


目标模型
前驱制品：
相关文档
❶ based_on
场景模型
场景描述
（自然语言）
❷ conflicts
❸ formalises
❹ classifies
面向方案的
需求模型
面向方案的
需求（自然语言）
❺ refines
面向方案的
需求模型
后续制品：
系统体系结构
satisfies ❻

需求追踪关系的管理

- **表示方式**
  - 自然语言文档及需求模型中的引用和超链接
  - 追踪关系矩阵、追踪关系图
  - 基于数据库的需求管理工具（例如IBM Rational RequisitePro）
- **追踪关系管理的困难**
  - 追踪关系的数量和种类非常多
  - 追踪关系非常复杂（多对多、横切）
  - 追踪关系的持续演化和更新

需求优先级管理

- 意义：支持权衡决策、制订增量开发计划
- 基本的需求优先级等级
  - 基本需求：支撑目标软件产品实现基本功能及提供基本服务的需求，如果不能实现则目标软件产品将无法满足用户及客户的基本使用要求
  - 期望需求：用户、客户及其他相关涉众所提出的不属于基本需求的其他需求，如果实现的话可以提高用户及客户的满意度
  - 兴奋型需求：用户、客户及其他相关涉众并未意识到或明确提出的需求，这些需求可能会改进目标软件产品并为用户带来便利，但其对于提高用户及客户满意度的作用还有待评估

需求变更管理

- 分析评估：变更控制组对变更请求进行分类以及时间和成本估计
- 变更决策：对变更请求做出接受或拒绝的决策，对于接受的变更请求确定其优先级
- 变更实施：安排开发人员实现变更请求、进行测试验证并与目标软件产品进行集成
- 变更发布：根据产品发布策略确定新的产品版本的发布时间和方式

高级软件工程

软件需求工程

The End