软件工程

第四部分 面向对象方法学

面向对象分析

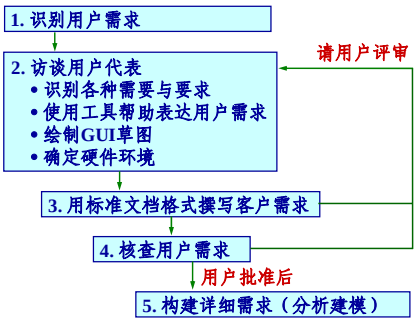
10.1 需求获取
10.2 面向对象分析
10.3 面向对象的分析：
案例分析(略)

10.1 需求获取

- 需求获取的目标是确定用户“需要”什么样的软件产品。就是说，新的软件必须能够做什么。
  - ✓没有专业的系统分析人员，用户很难了解到需要开发什么相关信息和功能；
  - ✓另一方面，没有与用户的交流，系统分析人员也很难弄清客户真正需要什么。
- 发现用户需求的过程称为需求获取。一旦提出了最初的需求，进一步推敲、细化和扩充的过程称为需求分析。

- 需求获取的第一步是理解应用领域，即目标软件的应用环境。如银行、电信公司、书店等。
- 一旦系统分析人员对该领域有了充分了解，就可以建立一个业务模型，描述用户的业务过程，确定用户的初始需求。然后通过迭代，更深入了解应用领域，回过头来推敲业务模型。
- 这种迭代过程直到双方对需求的理解达到共识。
- 需求获取的结果是导出用户可理解的系统规格说明。

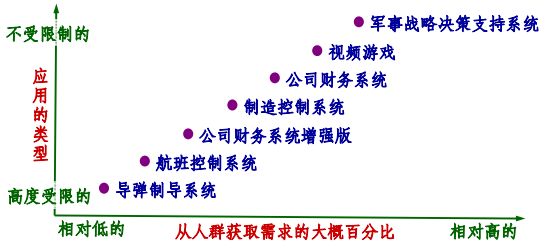
开发用户需求的典型过程



10.1.1 与用户交互

1) 需求的来源

- 不同类型应用能从人员处获取需求的比例：



- 限制是指受客观物理规律的限制。如导弹制导系统更多地受物理运动定律的限制，而非人的决策。视频游戏的大部分需求依赖人，因为它是一个想象出来的产品。
- 应用受到的限制越少，能从人们那里获得的需求比例越大。

2) 识别利益相关者（stakeholder）

- 对项目承担风险和享有利益的人即为利益相关者。他们是应用的“客户”。如公司高层、项目经理、最终用户、系统开发人员等。
- 不同利益相关者之间的利益冲突会导致需求不一致。如果需求冲突不能调和，项目就会陷入困境，最后往往会被取消。
- 即使所有利益相关者的需求一致，也可能由于实现代价高昂，需求不能得到完全满足。

3) 了解客户的需求

- ✓ 一般客户希望得到一个产品，他们需要系统开发人员帮助，明确自己的需要。
- ✓ 例如，有一个客户愿望框架：“Encounter是一个角色扮演游戏，它能模拟被扮演人物的全部或部分活动，应对人们具有相当吸引力。”

- ✓ 完整的客户要求应当记录在需求文档的“概述”部分。但需求中还有一些问题需要由系统分析人员与客户商量，以明确这些需求。
- ✓ 例如游戏是否只允许玩家扮演一个角色还是可以同时控制多个人物？当两个人相遇时会发生什么事情？游戏是否可以联网对战等。

4) 访谈和文档记录

- 大部分需求获取是人与人沟通的活动，这些活动经过精心组织，以准确获得最好的效果。
- 准备和访谈客户的过程如下:

访谈之前

①列出访谈的“客户”对象，并划分客户优先级
最有可能决定项目成败的人

①安排访谈日程，设定开始和结束时间
系统开发人员至少有两人参加访谈
准备录音设备

访谈中

③ 注意倾听
不要处于被动状态：启发和鼓励
- ✓ 理解客户的需要并探索要求
- ✓ 采用用例？或数据流图？状态图？

记录全部访谈内容

④ 安排补充会议

访谈之后

⑤ 根据标准模版撰写软件需求规格说明，打客户需求草稿

⑥ 通过电子邮件征求客户意见

- 对于不同类型的应用，用例方法是一种获取和表达需求的有效方法。
- 某些需求需要通过数据流图或状态图与用户沟通。

10.1.2 描述客户需求

- 需求可以看成是应用与应用的外部代理（如用户）之间的交互。可利用用例作为表达工具。
- 用例描述了系统外的参与者（Actor）与应用之间的交互情况。主要注重用户对系统的看法。
- 描述客户需求的过程如下:
  1) 标识参与者　标识目标系统将支持的不同类型的用户，可以是人、事件或其他系统。
  2) 标识场景　用场景描述目标系统典型功能的活动细节，并与用户沟通，加深开发人员对应用领域的理解。

3) 标识用例　当双方确定了一组场景后，开发人员从该场景抽象出一组用例，描述所有可能的情况。用例表达了系统的范围。
4) 求精用例　细化每一个用例。引入带有出错处理或带有异常处理的用例，描述系统的行为，保证需求的描述是完全的。
5) 标识用例之间的关系　描述用例之间的依赖关系，提取相同功能，建立用例模型。
6) 标识非功能需求　包括系统性能上的约束、文档、使用资源、安全性和质量等需求。

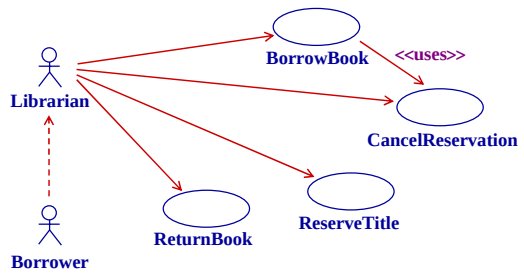
- 需求获取期间，开发人员需要访问一些不同的信息资源:
  - ✓客户提供的与应用领域相关的文档和手册。
  - ✓将被目标系统替代的遗留系统的技术文档。
  - ✓最终用户和客户本人。

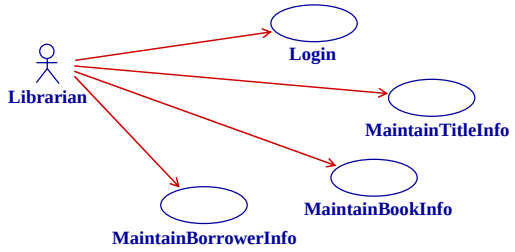
- 以“图书管理系统”为例，首先标识参与者:
  a)Librarian 图书管理员: 创建、修改、删除借阅者信息; 添加、编辑、删除书目信息; 添加、编辑、删除图书信息。
  b)Borrower 借阅者: 借阅、预约、归还图书，以及取消书目预约。
- 图书（Book）是指某种书目（Title）的某一流通中的复本。例如“数学分析教程第二册”的 5 本馆藏复本中的第 3 本。

- 识别用例:
  a) BorrowBook: 借阅图书
  b) ReturnBook: 返还图书
  c) RecerveTitle: 预约某种书目
  d) CancelReservation: 取消预约
  e) MaintainBorrowerInfo: 维护借阅者信息，包括创建、修改、取消借阅者账户
  f) MaintainTitleInfo: 维护书目信息，包括添加修改、删除书目信息
  g) MaintainBookInfo: 维护图书信息，包括添加、修改、删除图书信息
  h) Login: 登录

- 识别参与者与用例之间的关系（场景）
  a) Borrower执行BorrowBook、ReturnBook、ReserveTitle、CancelReservation等用例。
  b) Borrower是通过Librarian完成上述用例的工作。则Borrower与Librarian存在依赖关系。
  c) Librarian还与MaintainBorrowerInfo、Main- tainTitleInfo、 MaintainBookInfo交互。
  d) Librarian还需要与用例Login交互。

- 画出用例图





- 用例BorrowBook的规格说明

  1.1 前置条件：在此用例开始之前，Librarian必须登录到系统中。

  1.2 后置条件：如果此用例执行成功，在系统中建立并存储一条借阅记录，必要时需要删除预约记录。如果执行不成功，系统状态不变。

1.3 事件流

基本流

✓ 当Borrower借阅某种书目，且Librarian选择“借书”，则此用例启动。

① 提供书目和借阅者信息。

② 检索书目（E-1）。

③ 确定该书目的物理复本（图书）是否在架上（E-2）。

④ 检索借阅者（E-3）。

⑤ 将图书交给借阅者。

⑥ 创建并存储借阅记录。

⑦ 删除预约记录。

候补流

E-1：若该种书目不存在，系统显示提示信息，用例终止。

E-2：若该种图书都已借出，系统显示提示信息，用例终止。

E-3：系统中不存在该借阅者，系统显示提示信息，用例终止。

10.1.3 与用户沟通的其他工具

1) 数据流图

✓某些需求可以很自然地表述为处理元素之间的数据流。

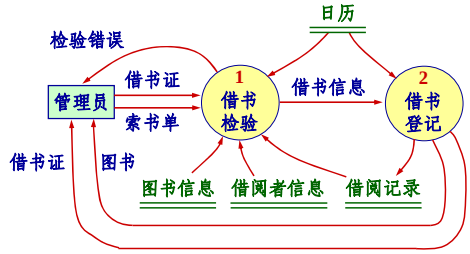
✓顶层图即为系统与外部实体的交互。

2) 状态图

✓有时把应用看作是几个状态下的应用，而在某一确定时刻的应用始终明确地处于某个状态中。这种状态划分对理解系统比较有益。

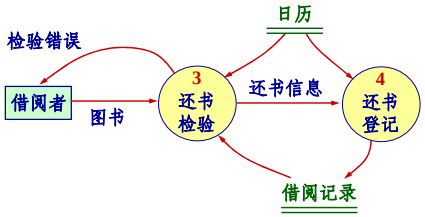
✓状态的具体内容到实现阶段会有确切的定义。

借书过程的数据流图



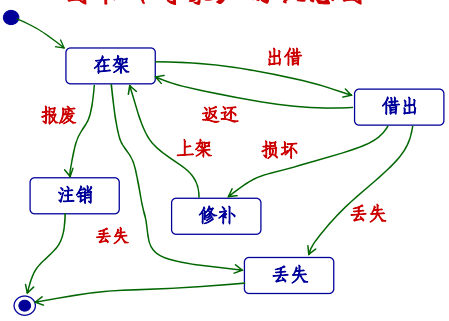
- 外部实体、数据流和数据存储都为候选对象

还书过程的数据流图

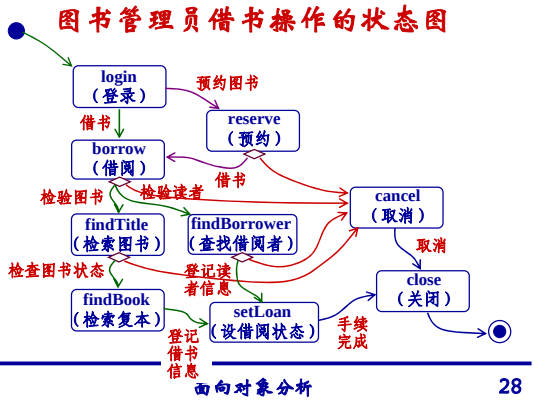


- 系统与外部实体、系统与数据存储的交互，构成系统的接口。相应数据流构成接口数据。

图书（对象）的状态图



图书管理员借书操作的状态图



10.1.4 草拟用户界面和其他接口

- 建立初始用户界面，是原型方法的一种，目的是快速与客户沟通。客户通常在看到应用的图形用户界面（GUI）才能相像到这个应用未来的样子。
- 开发用户界面的步骤如下：

1) 了解客户　深入了解最终用户的想法。根据用户的层次，提供多种用户界面。

✓知识和经验层次：计算机素养、系统经验、使用类似应用的经验、教育水平、阅读水平、打字技能等。

✓ 用户的生理特征：年龄、性别、左右手习惯、生理障碍等。

2) 理解业务功能　根据应用的整体意图来理解特定用户界面的目的。功能界面出现的顺序通常可以反映用户处理日常业务的方式。

✓ 用户的任务和工作特征：应用的使用方式、使用频率、雇员的流动率、任务的重要性、任务的重复性、对培训的期望、工作类型等。

✓ 用户的心理特征：工作态度、能动性、认知方式等。

3) 理解优秀界面设计的原则　目的是加强视觉效果。
   - ✓ 确保应用的各个界面之间风格的一致性：习惯、步骤、视觉和感觉、位置等。
   - ✓ 描测用户通常开始操作的地点
   - ✓ 导航系统尽量简捷
   - ✓ 使用分组和分层来强调重要性级别
4) 选择合适的窗口类型　五类窗口：
   - ✓ 属性窗口：展示实体的属性
   - ✓ 对话窗口：完成特定任务或命令的信息

   - ✓ 消息窗口：提供信息
   - ✓ 面板窗口：展示一组控件
   - ✓ 弹出窗口：突出显示信息
5) 制作系统菜单　为用户提供一个稳定的、易于理解的使用环境，可以方便地搜寻需要的选项。
   - ✓ 提供一个主菜单
   - ✓ 显示所有相关选择（仅局限于此）
   - ✓ 将菜单结构与应用要完成的任务对应起来
   - ✓ 尽量减少菜单的级数

6) 选择合适的基于设备的控件　提供给用户，向系统发送指示的实际手段，包括鼠标、键盘、触摸屏、绘图板、轨迹球、麦克风等。
7) 选择合适的基于界面的控件　即出现在屏幕上的符号。用户通过这些符号向系统提出他的输入和操作意图，包括图标、按钮、复选框、单选框等。
8) 组织和安排窗口布局　多窗口的排列规则，如平铺、层叠等。
9) 选择合适的颜色　尽量保持简捷和低调。颜色需要和谐。

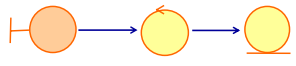
10.2 面向对象分析

- 分析建模的目的是对来自客户的需求形式化。形式化可以导致新的洞察和发现需求错误。
- 1999年NASA损失了一颗价值数亿美元的气象卫星，据调查是因为列在度量表中的控制数据出了问题。不巧的是这个缺陷在灾难发生几天之前才刚发现，如果在需求分析阶段就被识别出来就可避免损失了。
- 避免需求错误或遗漏的第一道防线就是把所有的需求细化，建立分析模型。

- 分析模型由三个独立的模型构成：
  - ✓ 由用例和场景表示的功能模型；
  - ✓ 用类和对象表示的对象模型；
  - ✓ 由状态图和顺序图表示的动态模型。
- 在需求获取阶段得到的用例模型就是功能模型。据此可导出分析对象模型和动态模型。
- 分析中的类可以看作是高层抽象，在后续阶段将使用更多的细节实现。

- 注意：这些模型代表的是来自客户的概念，而非实际软件类或实际构件。如数据库、子系统、会话管理器、网络等，不应出现在分析模型中，因为这些概念仅与实现相关。

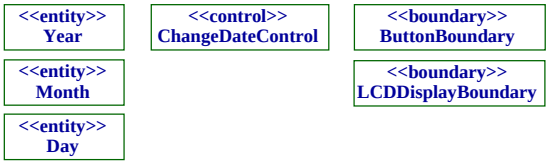
- 在分析对象模型中有三种类型的对象:
  - ✓实体对象表示系统将跟踪的持久信息;
  - ✓边界对象表示参与者与系统之间的交互;
  - ✓控制对象负责用例的实现 。



Boundary 边界对象　Control 控制对象　Entity 实体对象

- 可用UML提供的衍型机制，区分不同类型对象。

具有两个按钮的手表的分析类



- 使用实体对象、边界对象和控制对象等概念对系统建模时，常常需要提供一些简单的启发式规则来指导开发人员使用这些概念，可以使用相应的类来跟踪。

- 分析建模活动包括以下步骤。

1) 标识实体对象

✓ 自然语言分析法　利用Abbott启发式准则，将语言成分映射为模型成分。

语言成分	模型成分	示例
专有名词	实例	人员乙
普通名词	类	现场工作人员
Doing动词	操作	创建、提交、选择
Being动词	继承	是…的一种，是…中的一个
Having动词	聚合	有…，由…组成，包括…
情态动词	约束	必须是
形容词	属性	事件描述

✓自然语言分析法主要关注用户术语,依赖:
- ➢识别质量高度依赖人们的书写风格;
- ➢可能会出现许多无关词汇，或同义词。

✓检查每一个用例，标识候选对象
- ➢用例中的连续名词（如借阅事件）;
- ➢系统需要跟踪的现实世界中的实体（如借阅记录、书目信息）;
- ➢系统需要跟踪的现实世界中的活动（如紧急情况操作预案）;
- ➢数据源（如借阅者、管理员）。

2) 标识边界对象

✓在用例图中，每一个参与者至少要与一个边界对象交互。边界对象收集来自参与者的信息，将它们转换为可用于实体对象和控制对象的表示形式。

✓边界对象对用户界面进行粗略的建模，不涉及如菜单项、滚动条等可视方面的细节。

✓ 标识边界对象的启发式准则如下:
- ➢标识用户所需初始用例的用户界面控制;
- ➢标识用户需要键入系统的数据表格;
- ➢标识通知和系统用于响应用户的消息;
- ➢当用例中有多个参与者时，根据构想的用户界面来标识参与者的行为;
- ➢不要使用边界对象对接口的可视方面建模，应使用用户原型对可视用户界面建模;
- ➢使用用户的术语来描述接口，不要使用来自设计和实现的术语。

3) 标识控制对象

- ✓ 控制对象负责协调实体对象和边界对象。
- ✓ 控制对象没有在现实世界中具体的对应物，它通常从边界对象处收集信息，并把这些信息分配给实体对象。

图书管理系统中的实体对象

① **Borrower**：借阅者。他们可以借阅、返还、预约和取消预约。因为名字可能重复，可用借阅者ID识别。

② **Title**：书目。它表明某一种书，通过ISBN号码识别。

③ **Book**：图书。它表明某一种书目的具体物理复本，通过馆藏号码识别。

④ **Loan**：借阅记录。同一个人关于不同图书的借阅记录是不同的。

⑤ **Reservation**：预约记录。

图书管理系统中的边界对象

① **mainWindow**：主窗口。有借书、还书、预约、取消预约、添加书目、修改书目、删除书目、添加借阅者、修改借阅者、删除借阅者、添加图书、删除图书等操作。

② **BorrowerDialog**：借阅者对话框。有添加借阅者、修改借阅者、删除借阅者等操作。

③ **FindBwrDialog**：弹出对话框。有根据借阅者ID查找借阅者的操作。

④ **TitleDialog**：书目对话框。有添加书目、修改书目、删除书目等操作。

⑤ **FindTDialog**：弹出对话框。根据图书的ISBN号码查找书目。

⑥ **BorrowDialog**：借书对话框。根据书目的ISBN号码和借阅者信息，执行借阅动作，创建和保存借阅记录。

⑦ **ReturnDialog**：还书对话框。根据图书的馆藏号码，执行还书动作，删除借阅记录。

⑧ **ReserveDialog**：预约对话框。根据书目的ISBN号码和借阅者信息，执行预约、取消预约动作。

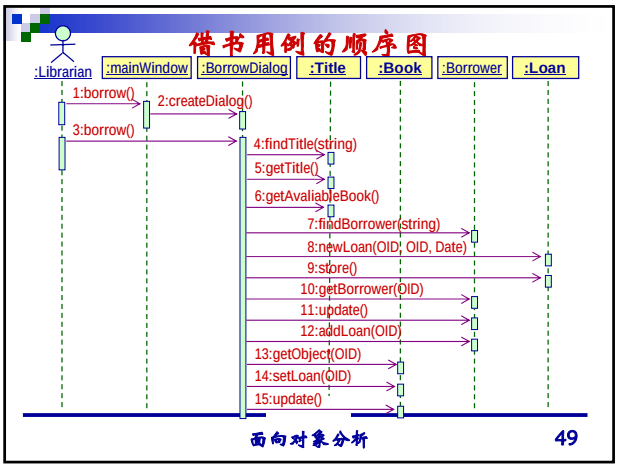
⑨ **MessageWindow**：显示提示信息窗口。

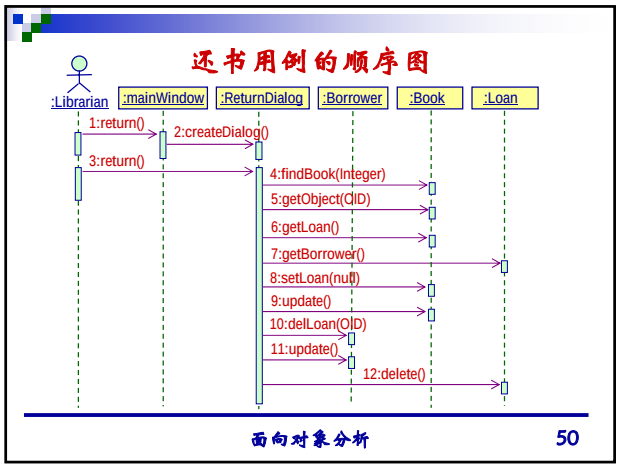
⑩ **LoginDialog**：输入用户名和密码的窗口。

4) 使用顺序图将用例映射为对象

- ✓ 顺序图将用例与对象联系起来，直观地描述了用例（场景）行为在其参与对象之间是如何实施的。
- ✓ 顺序图对用例中各参与对象之间的交互序列进行建模。每一个消息从一个对象（或参与者）发送给另一个对象（或参与者）。消息的接受就触发了一个操作。
- ✓ 通过顺序图，将责任以操作集合的形式分配给每一个对象。如果一个对象参与到多个用例，则其操作应为这些用例共享。

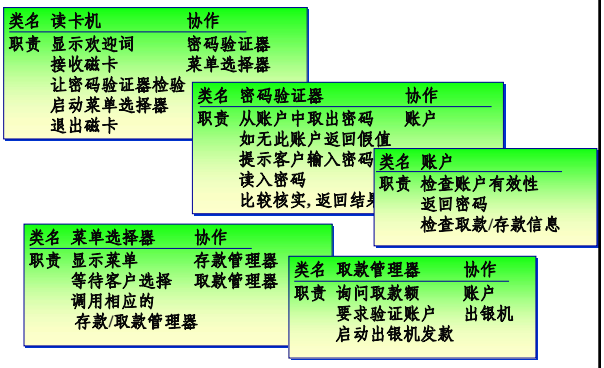
- ✓ 画顺序图的启发式准则如下：
  - ➢ 顺序图第一栏对应激活该用例的参与者；
  - ➢ 顺序图第二栏是边界对象；
  - ➢ 顺序图第三栏是管理用例中其他参与对象的控制对象；
  - ➢ 通过边界对象来初始化用例，并创建控制对象；
  - ➢ 通过控制对象可创建其他边界对象；
  - ➢ 实体对象允许边界对象和控制对象访问；
  - ➢ 实体对象不能访问边界对象和控制对象；





5) 使用CRC卡片对对象之间的交互建模

- ✓ CRC是类、职责和协作的缩写。每一个类可用一张CRC卡片表示。
- ✓ 建立CRC卡片有以下几个步骤:
  - ① 识别类和职责: 首先识别类或对象，然后从客户需求说明中寻找有关行为的描述，以发现职责。
  - ② 将职责分配到类：记录在相应的卡片上。
  - ③ 找寻协作者: 依次检查每一类承担的责任，看是否需要其他类的帮助，找寻与每个类协作的伙伴，并记录在相应卡片上。



④ 细化: 模拟在执行每个基本功能时系统内部出现的场景，以此推动细化工作的进行。

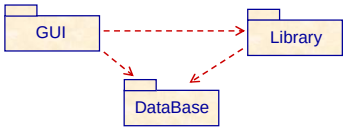
- ✓ 在模拟一个场景的过程中，每当一个类开始“执行”时，它的卡片就被拿出来讨论，当“控制”传送到另一个类时，注意力就从前一张卡片转移到另一张上去了。不同的场景，包括例外和出错状况，都应逐一加以模拟。
- ✓ 在这个过程中可以验证已有的定义，不断发现新的类、职责以及伙伴。
- ✓ 在模拟不同的场景中会发现某些职责需要重新加以分配。这些都导致进一步的开发工作。

6) 标识关系（结构）

- ✓ 使用类图，能够表示对象之间的关系。
- ✓ 关联表示了两个或多个类之间的关系。标识关联的启发式准则如下:
  - 检查指示状态的动词或动词短语;
  - 准确地命名关联和角色;
  - 尽量使用常用的修饰词标识出名字空间和关键属性;
  - 消除可导出其他关联的关联;
  - 在关联集合稳定之前不必关心重复性;
  - 过多的关联使得一个模型不可读;

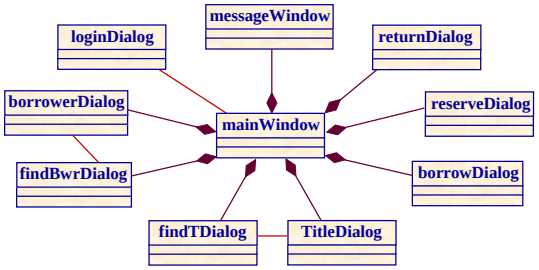
a) 建立系统的包图（主题）

- 建立包图是为了降低复杂性，目的是控制可见度及指引读者的思路。
- 对于面向对象分析模型，主题表示此模型的整体框架。可以是一个层次结构。
- 通过对主题的识别，可以让人们能够比较清晰地了解大而复杂的模型。



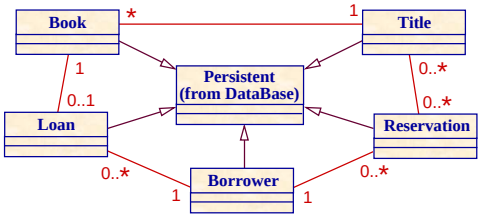
b) 建立边界类的类图

- 标明类之间的关系，包括关联、泛化等。

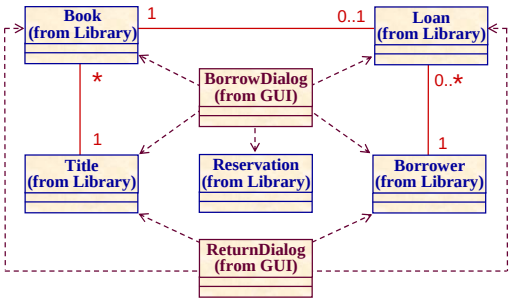


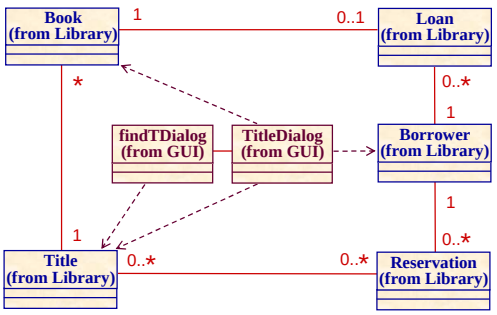
c) 建立实体类的类图

- 这些类与数据库相关，为了操作方便，以它们为子类，建立一个持久类（Persistent）作为它们的父类，共享与数据库相关的操作。



d) 建立边界类与实体类之间关系的类图





7) 标识属性

- ✓ 类的属性所描述的是状态信息，每个实例的属性值表达了该实例的状态值。
- ✓ 标识属性的启发性准则如下：
  - 每个对象至少需包含一个属性；
  - 属性取值必需适合对象类的所有实例；
  - 对象继承的属性必须与泛化关系一致；
  - 系统的所有存储数据必须定义为属性；
  - 对象的导出属性应当略去。例如，“年龄”是由属性“出生年月”导出，不能作为基本属性存在。

8) 对每一对象的与状态有关的行为建模

- ✓对象收到消息后所能执行的操作称为它可提供的服务。
- ✓对每个类的增加、修改、删除、选择等服务有时是隐含的，在图中不标出，但实现类和对象时有定义。
- ✓其它服务则必须显式地在图中画出。
  - ①首先标识在每个类中封装的服务;
  - ②再比较服务与类的属性，验证其一致性。所标识的类属性，它必然关联到某个服务，否则该属性就形同虚设，永远不可能被访问;

③画出对象之间的消息通信路径，协调系统的行为。

- ➢ 自底向上方法：找出每一对象在其生存周期中的所有状态。每一状态的改变都关联到对象之间消息的传递。从对象着手，逐渐向上分析。
- ➢ 自顶向下方法：一个对象必须识别系统中发生或出现的事件，产生发送给其他对象的消息，由那些对象作出响应。所以对象应具有能够接收、处理、产生每个消息的服务。它是从系统行为着手，然后逐渐分析到对象。

④ 使用顺序图或协作图，标识和描述对象之间的相互通信，并进行运行走查。

9) 分析模型评审

有关模型正确性的问题:

- ✓ 对实体对象的分类，用户是否能够理解?
- ✓ 抽象类是否对应到用户层的定义?
- ✓ 是否所有的描述均符合用户的定义?
- ✓ 是否所有的实体对象和边界对象都使用了有实际含义的名词短语进行了命名?
- ✓ 是否所有的用例和控制对象都使用了有意义的动词短语进行了命名?
- ✓ 是否所有的错误用例都已经描述和处理?

有关模型完备性的问题:

- ✓ 每一个对象是否有用例用到它？创建、修改或删除该对象的用例是哪些?
- ✓ 每一个属性的类型是什么？它应进行修饰吗?
- ✓ 每一个关联何时被用到？其重复性的选择原则是什么？该关联使用那一种连接?
- ✓ 每一个控制对象是否具有必要的关联，以连接到用例中相关的其他对象?

有关模型一致性的问题:

- ✓ 是否有多个类或多个用例同名?
- ✓ 名字相近的实体（如类、对象、属性）能够相互区别吗?
- ✓ 在同一泛化层次是否存在相似属性和关联的对象?

有关模型可实现性的问题:

- ✓ 在该系统中性能需求和可靠性需求是否满足?
- ✓ 在选定硬件上运行原型是否可以确定需求?

10.3 面向对象分析:实例分析

- 项目背景

空间太阳望远镜卫星系统(简称SST)是国家战略性的重大投资项目，旨在研制高分辨率的空间太阳望远镜，提高天文观测与研究能力。

SST系统由火箭、发射系统、卫星平台、测控、有效载荷等五个子系统组成，其中有效载荷子系统是整个项目的核心。