|  |
| --- |
| tongji |
| UML课程设计Assignment4 |
| 新型触摸式ATM自助购票系统 |
|  |
| **Team: Boundary**  **组员：尹禹薇 黄啸 杨雨龙 李风忠** |
| **2010-6-8** |

|  |
| --- |
|  |

目录

[一、项目概述 4](#_Toc263783404)

[1.1 项目目标 4](#_Toc263783405)

[1.2 基本功能 4](#_Toc263783406)

[1.3 用例建模 6](#_Toc263783407)

[1.4 软件生命周期 6](#_Toc263783408)

[1.5 软件架构分析 9](#_Toc263783409)

[二、用例模型和分析模型 10](#_Toc263783410)

[2.1 用例模型 10](#_Toc263783411)

[2.1.1系统操作员用例 10](#_Toc263783412)

[2.1.2客户用例 12](#_Toc263783413)

[2.1.3银行用例 19](#_Toc263783414)

[2.1.4票务公司用例 20](#_Toc263783415)

[2.2 分析模型 22](#_Toc263783416)

[2.2.1 登录活动图 22](#_Toc263783417)

[2.2.2 订票活动图 23](#_Toc263783418)

[2.2.3 票务查询活动图 24](#_Toc263783419)

[2.2.4 退票活动图 25](#_Toc263783420)

[2.2.5 余额查询活动图 26](#_Toc263783421)

[2.2.6 转账活动图 27](#_Toc263783422)

[三、设计模型 28](#_Toc263783423)

[3.1 静态类图 28](#_Toc263783424)

[3.2 序列图 29](#_Toc263783425)

[3.2.1 取款序列图 29](#_Toc263783426)

[3.2.2 订票序列图 30](#_Toc263783427)

[3.2.3 转账序列图 31](#_Toc263783428)

[3.2.4 退票序列图 32](#_Toc263783429)

[3.2.5 存款序列图 33](#_Toc263783430)

[3.3 通讯图 34](#_Toc263783431)

[3.3.1 取款交互图 34](#_Toc263783432)

[3.3.2 订票交互图 35](#_Toc263783433)

[3.3.3 转账交互图 36](#_Toc263783434)

[3.3.4 退票交互图 37](#_Toc263783435)

[3.3.5 存款交互图 38](#_Toc263783436)

[四、项目架构文档 38](#_Toc263783437)

[4.1简述 38](#_Toc263783438)

[4.1.1目的 38](#_Toc263783439)

[4.1.2范围 39](#_Toc263783440)

[4.1.3概述 39](#_Toc263783441)

[4.2架构的表示 39](#_Toc263783442)

[4.2.1系统功能划分 40](#_Toc263783443)

[4.2.2系统层次与模块划分 40](#_Toc263783444)

[4.2.3系统物理分部 40](#_Toc263783445)

[4.2.4为实施制定的构架决策 40](#_Toc263783446)

[4.3架构的目标约束 41](#_Toc263783447)

[4.4用例视图 41](#_Toc263783448)

[4.4.1用户 42](#_Toc263783449)

[4.4.2 用例图 43](#_Toc263783450)

[4.5逻辑视图 43](#_Toc263783451)

[4.5.1概览 43](#_Toc263783452)

[4.5.2对架构重要的模型元素 46](#_Toc263783453)

[4.5.2.1数据支持层 48](#_Toc263783454)

[4.5.2.1.1重要类简述 48](#_Toc263783455)

[4.5.2.2业务逻辑层 48](#_Toc263783456)

[4.5.2.2.1重要类简述 49](#_Toc263783457)

[4.5.2.3表现层 51](#_Toc263783458)

[4.6实施视图 51](#_Toc263783459)

[4.7部署视图 51](#_Toc263783460)

[4.8大小与性能 51](#_Toc263783461)

[4.9质量 51](#_Toc263783462)

[五、组员分工 52](#_Toc263783463)

# 一、项目概述

## 项目目标

使用UML思想，通过UML的工具，采用OO的方法对ATM的软件程序进行完整的软件设计。

对于ATM中的流程进行优化，对以前一些存在的弊端进行处理，提高ATM机的稳定性、健壮性、安全性、易操作性等。

在原来传统的ATM登录系统中我们小组新设计了通过触摸式登陆系统，也即是说可以通过指纹代替密码进行登录。同时通过指纹识别功能更加确保安全性。

在功能方面，我们项目中的ATM不仅可以进行ATM机的基本功能，而且和铁路以及民航部门售票系统进行连接，实现自动销售火车票和飞机票。

## 基本功能

新型触摸式ATM自助购票系统是为了克服现有ATM系统不足，而开发的一套新型的，多功能的，高安全性的ATM系统。在实现ATM系统基本功能，如取款，存款，转账，查询等的基础上，新型触摸式ATM自助购票系统进行了如下的改进与扩展：

在普通ATM机现有功能的基础上，增加了用户自主购票功能。用户身份验证后，通过相应功能模块可在ATM机上实现票务的查询，订购，支付，订单查询，退订等业务。将购票的支付业务整合到ATM机上之后，支付的便捷性，安全性有了很大的提高。

新型触摸式ATM自助购票系统将使用动态口令输入来进行用户身份的验证，每次都是随机给出字符，要求客户按照最开始设置对应的密码进行输入。相对于传统的固定密码验证，本系统在安全性上将有很大提高。

同时，本系统添加了指纹识别安全保障，用户可以通过指纹直接进行身份验证。而当用户输入密码超过错误次数限制的时候，系统将要求用户进行指纹验证，通过验证后则进行正常业务，否则自动锁卡。

新型触摸式ATM自助购票系统同时提供传统固定密码验证，动态口令验证，指纹身份验证，用户可根据自己的需要和实际情况选择适合自己的方式进行身份验证。例如，用户如在开户行进行过指纹采集，则可选择使用指纹识别验证；如果用户为跨行使用ATM机且没有进行过指纹采集，则可使用传统的固定密码验证来使用系统。

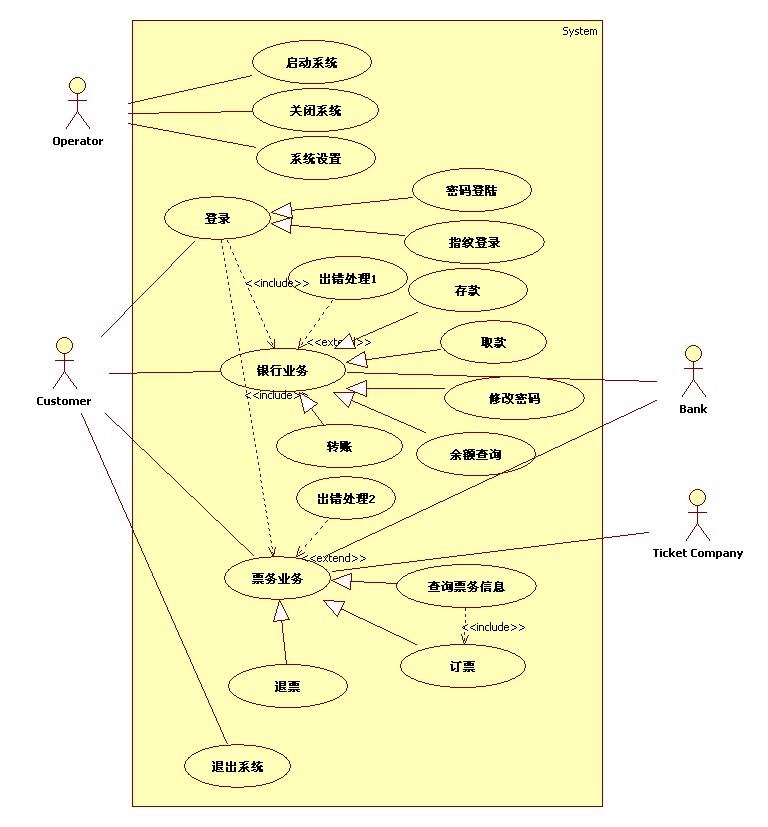
多种选择，多种方式，较传统ATM系统，新型触摸式ATM自助购票系统在安全性方面有了很大的提高。

在服务方面，新型触摸式ATM自助购票系统采用触摸屏与传统键盘结合，用户选择自己喜欢的方式方便的使用系统。系统采用全图形化界面，操作简单，易懂。系统响应速度较传统ATM系统有很大提高。

如果用户因为忘记密码或其他原因而发生吞卡，如果用户进行过指纹采集，可立刻在ATM机上进行指纹验证，通过验证后，被吞卡将退回给用户，用户不必到ATM机管理银行排队领卡，极大的方便了客户。

本系统能给用户带来全新的ATM机使用体验。

## 用例建模



## 软件生命周期

软件生命周期是软件的产生直到报废的生命周期，周期内有问题定义、可行性分析、总体描述、系统设计、编码、调试和测试、验收与运行、维护升级到废弃等阶段， 这种按时间分程的思想方法是 软件工程 中的一种思想原则，即按部就班、逐步推进，每个阶段都要有定义、工作、审查、形成文档以供交流或备查，以提高软件的质量。但随着新的 面向对象 的设计方法和技术的成熟，软件生命周期设计方法的指导意义正在逐步减少。

软件生命周期(SDLC)的六个阶段

1、问题的定义及规划  
      此阶段是软件开发方与需求方共同讨论，主要确定软件的开发目标及其可行性。

2、需求分析  
      在确定软件开发可行的情况下，对软件需要实现的各个功能进行详细分析。需求分析阶段是一个很重要的阶段，这一阶段做得好，将为整个软件开发项目的成功打下 良好的基础。"唯一不变的是变化本身。"，同样需求也是在整个软件开发过程中不断变化和深入的，因此我们必须制定需求变更计划来应付这种变化，以保护整个 项目的顺利进行。

3、软件设计

此阶段主要根据需求分析的结果，对整个软件系统进行设计，如系统框架设计，数据库设计等等。软件设计一般分为总体设计和详细设计。好的软件设计将为软件程 序编写打下良好的基础。

4、程序编码

此阶段是将软件设计的结果转换成计算机可运行的程序代码。在程序编码中必须要制定统一，符合标准的编写规范。以保证程序的可读性，易维护性，提高程序的运 行效率。

5、软件测试

在软件设计完成后要经过严密的测试，以发现软件在整个设计过程中存在的问题并加以纠正。整个测试过程分单元测试、组装测试以及系统测试三个阶段进行。

6、 运行维护

软件维护是软件生命周期中持续时间最长的阶段。在软件开发完成并投入使用后，由于多方面的原因，软件不能继续适应用户的要求。要延续软件的使用寿命，就必 须对软件进行维护。软件的维护包括纠错性维护和改进性维护两个方面。

二、软件生命周期模型

从概念提出的那一刻开始，软件产品就进入了软件生命周期。在经历需求、分析、设计、实现、部署后，软件将被使用并进入维护阶段，直到最后由于缺少维护费用而逐渐消亡。这样的一个过程，称 为"生命周期模型"（Life Cycle Model）。

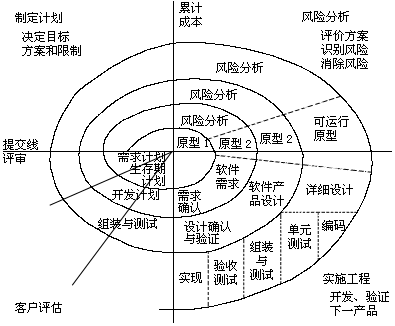
典型的几种生命周期模型包括瀑布模型、快速原 型模型、迭代模型。

瀑布模型（Waterfall Model）首先由Royce提出。该模型由于酷似瀑布闻名。在该模型中，首先确定需求，并接受客户和SQA小组的验证。然后拟定规格说明，同样通过验证后，进入计划阶段。

 迭代式模型是RUP推荐的周期模型，迭代被定义为：迭代包括产生产品发布的全部开发活动和使用该产品发布必需的所有其他外围元素。所以，在某种程度上，开发迭代是一 次完整地经过所有工作流程的过程：需求工作流程、分析设计工作流程、实施工作流程和测试工作流程。

快速原型 （Rapid Prototype）模型。快速原型模型在功能上等价于产品的一个子集。瀑布模型的缺点就在于不够直观，快速原型法就解决了这个问题。一般来说，根据客户的需要在很短的时间内解决用户最迫切需要，完成一个可以演示的产品。这个产品只是实现部分的功能。它最重要的目的是为了确定用户的真正需求。

在本项目开发中，我们采用了传统的瀑布模型，强调文档的重要性，以及对需求分析的反复确认和分析，在每一次进一步的设计中，对需求进行修改与重新确认。模型如下图所示：



## 软件架构分析

该系统采用经典的MVC架构，即把一个应用的输入、处理、输出流程按照Model、View、Controller的方式进 行分离，这样一个应用被分成三个层——模型层、视图层、控制层。

视图(View)代表用户交互界面，该系统使用JAVA中的抽象窗口工具箱（Abstract Window Toolkit，AWT）上实现的，其特点是具体实现由目标平台下的OS来解释，从而导致Java GUI在不同平台下会出现不同的运行效果（窗口外观、字体等的显示效果会发生变化），以及组件在设计时不应采用绝对定位，而应采用布局管理器来实现相对定位，以达到与平台及设备无关。

MVC设计模式对视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理交予模型(Model)处理。比如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

模型(Model)：就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其它层来说是黑箱操作，模型接受视图请求的数据，并返回最终的处理结果。控制(Controller)可以理解为从用户接收请求, 将模型与视图匹配在一起，共同完成用户的请求。

在该系统中使用类库来定义模型以及逻辑控制，类库中包含业务处理逻辑类、数据访问类以及数据对象类，以此来实现控制层的内容。

# 二、用例模型和分析模型

## 2.1 用例模型

### 2.1.1系统操作员用例

本用例描述了系统操作员与系统的交互。系统操作员所能进行的操作包括启动系统，关闭系统以及系统的设置。

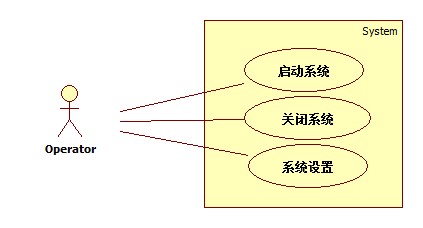


图4-3 Operator Use Cases

**用例描述**

|  |  |
| --- | --- |
| 用例名称 | 密码登录 |
| 主要角色 | 客户 |
| 前提条件 | 系统处于工作状态  客户银行卡可用 |
| 主要流程 | 客户插入银行卡  系统验证银行卡  客户选择通过密码验证还是指纹验证  系统提示客户输入密码  客户输入密码  系统验证密码  登录完成, 客户选择选择服务 |

|  |  |
| --- | --- |
| 用例名称 | 指纹登录 |
| 主要角色 | 客户 |
| 前提条件 | 系统处于工作状态  客户银行卡可用 |
| 主要流程 | 客户插入银行卡  系统验证银行卡  客户选择通过密码验证还是指纹验证  系统提示客户输入指纹  客户输入指纹  系统验证指纹  登录完成,客户选择选择服务 |

|  |  |
| --- | --- |
| 用例名称 | 系统设置 |
| 主要角色 | 系统管理员 |
| 前提条件 | 系统处于打开状态 |
| 主要流程 | 系统管理员选择系统设置  系统管理员输入密码  系统验证系统管理员身份  系统保存当前系统状态  对系统进行设置  保存设置  系统管理员选择退出 |

|  |  |
| --- | --- |
| 用例名称 | 启动系统 |
| 主要角色 | 系统管理员 |
| 前提条件 | 系统处于关闭状态 |
| 主要流程 | 系统加电  系统初始化  系统管理员输入密码  系统验证管理员身份  管理员选择系统启动  系统恢复关机前的参数状态  系统启动完成 |

|  |  |
| --- | --- |
| 用例名称 | 关闭系统 |
| 主要角色 | 系统管理员 |
| 前提条件 | 系统处于打开状态 |
| 主要流程 | 系统管理员选择系统管理员登陆  系统管理员输入密码  系统验证管理员身份  选择系统关闭  系统保存系统当前状态  系统关闭完成 |

### 2.1.2客户用例

本用例为最核心与最重要的用例之一。用户通过系统，可以进行存取款，查订票等核心业务。同时，用户帐户管理，指纹识别等系统特色功能在本用例也有体现。

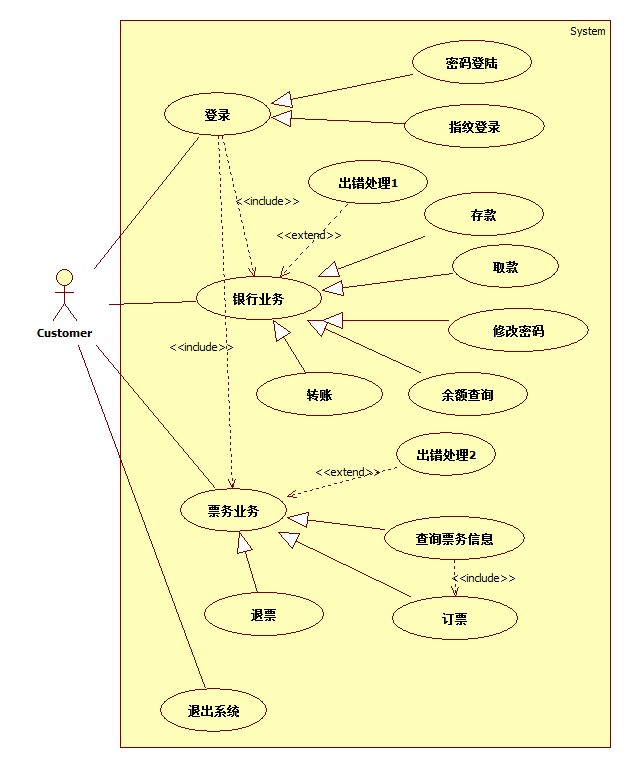


图4-4 Customer Use Cases

**用例描述**

|  |  |
| --- | --- |
| 用例名称 | 存款 |
| 主要角色 | 客户，银行 |
| 前提条件 | 客户已经登录 |
| 主要流程 | 客户通过主界面选择存款的选项，进入存款的页面  ATM打开存款放钞的窗口，客户放入想存的钞票数量  ATM开始检验钞票的真伪并统计钞票数量  显示存款的钞票数量，并且询问库户是否继续放入钞票  若客户选择继续放入钞票，则返回第二步继续放入钞票；若停止放钞则在账户内增加相应数额的金额  提示是否打印凭条或者查询余额  若客户选择其中之一的功能，则进入相关界面，否则退回主界面 |

|  |  |
| --- | --- |
| 用例名称 | 取款 |
| 主要角色 | 客户，银行 |
| 前提条件 | 客户已经登录，ATM机内有现金 |
| 主要流程 | 客户通过主界面选择取款的选项，进入取款的页面  输入取款金额  系统检查ATM内余额是否足够本次取款  若余额不足则弹出错误信息；若余额足够则吐出相应金额的钞票，以及在该客户账户上减去相应数额的金额  提示是否打印凭条或者查询余额  若客户选择其中之一的功能，则进入相关界面，否则退回主界面 |

|  |  |
| --- | --- |
| 用例名称 | 余额查询 |
| 主要角色 | 客户，银行 |
| 前提条件 | 客户已经登录 |
| 主要流程 | 客户通过主界面选择余额查询的选项，进入余额查询的页面  在查询余额的页面上查看自己账户上的余额  提示是否打印凭条  若客户选择打印凭条功能，则进入相关界面，否则退回主界面 |

|  |  |
| --- | --- |
| 用例名称 | 转账 |
| 主要角色 | 客户，银行 |
| 前提条件 | 客户已经登录 |
| 主要流程 | 客户通过主界面选择转账的选项，进入转账的页面  客户输入拟转入的银行账号，并且连续输入两次相同，否则系统认为输入账号有误  输入转账的金额  系统检查客户的账户余额是否足够进行本次转账业务，若账户余额小于转账金额则系统会提示余额不足；若账户金额大于转账金额则拟转入账户余额增加输入的金额，而客户的账户减少相应金额  提示是否打印凭条或者查询余额  若客户选择其中之一的功能，则进入相关界面，否则退回主界面 |

|  |  |
| --- | --- |
| 用例名称 | 出错处理 |
| 主要角色 | 客户，银行 |
| 前提条件 | 客户已经登录 |
| 主要流程 | 客户在前四种银行业务中进行了出现异常的操作  窗口出现错误信息提示，客户根据提示进行操作  根据提示操作后消除了异常，进行正常业务操作 |

|  |  |
| --- | --- |
| 用例名称 | 票务信息查询 |
| 主要角色 | 客户，票务公司 |
| 前提条件 | 客户已通过系统身份验证 |
| 主要流程 | 1：客户在主界面选择票务模块，进入票务模块后选择预定功能。  2：客户选择要预定的票的类型，如电影票或飞机票，进入相应操作界面。  3：客户输入想要预定票的时间范围与档次等信息，进行查询。  4：系统根据客户输入的查询条件列出所有符合要求的票务公司的报价信息。 |

|  |  |
| --- | --- |
| 用例名称 | 票务预定 |
| 主要角色 | 客户，票务公司 |
| 前提条件 | 客户已通过系统身份验证，客户已进行过票务信息的查询 |
| 主要流程 | 1：客户从查询结果中选择票务公司，查看详细信息。  2：客户输入想要预定的票数目，精确日期，档次信息后，提交信息。  3：客户输入领票人信息，包括姓名，证件号，手机号等。提交信息。  4：系统向客户展示客户预定票的所有详细信息，等待客户确认。  5：客户确认信息无误，进入结帐界面。  6：客户确认账户信息无误，输入消费密码。  7：付款成功后，系统提示客户取票地点与程序。如所订票为电子票，系统向客户提供验证码或向手机发送电子票。 |

|  |  |
| --- | --- |
| 用例名称 | 票务退定 |
| 主要角色 | 客户，票务公司 |
| 前提条件 | 客户已通过系统身份验证，客户已进行过票务预定 |
| 主要流程 | 1：客户在主界面选择票务模块，进入票务模块后选择票务退定。  2：系统根据客户信息，按时间顺序自动显示客户所有的订票记录。  3：客户根据显示的记录，选择要退票的订票记录，选择退票。  4：系统显示订票记录详细信息，等待客户确认。  5：客户检查信息无误后，点击确认退票。  6：系统响应客户请求，计算应退还客户的金额，并将相应数目的资金打入客户账户。  7：退款成功后，系统向客户提供成功信息。 |

|  |  |
| --- | --- |
| 用例名称 | 错误提示 |
| 主要角色 | 客户，票务公司 |
| 前提条件 | 客户已通过系统身份验证 |
| 主要流程 | 1：客户在以上三种操作中操作不当，或违反其他限制条件。  2：系统向客户给出错误提示。  3：客户更正错误操作，或系统自动退出当前操作。 |

|  |  |
| --- | --- |
| 用例名称 | 退出系统 |
| 主要角色 | 客户，票务公司，银行 |
| 前提条件 | 客户已通过系统身份验证 |
| 主要流程 | 1：客户在任何一种操作中，点击返回，退回到主页面。  2：客户选择退出系统。  3：如果客户已插卡，系统退卡。  4：系统清除当前客户信息，进入等待服务状态。 |

### 2.1.3银行用例

本用例为最核心与最重要的用例之一。通过对相应的银行用户赋予权限，银行用户通过终端机可进行所有普通用户进行的操作，包括普通银行务以及票务业务。但银行用户与普通银行储户的区别是银行用户不需要进行身份的验证便可以进入系统。

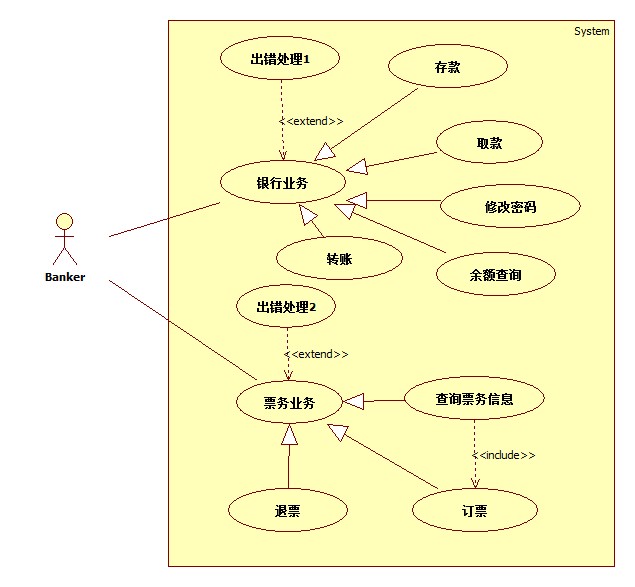


图4-5 Bank Use Cases

**用例描述**

因为银行用户所能进行的操作与普通用户完全相同，所以银行用户用例请见客户用例部分的用例描述。

### 2.1.4票务公司用例

通过给票务公司中相应人员赋予权限，该人员通过系统，可以进行所有普通用户可以进行的票务操作。包括查询，订票，退票，订单查询等操作。

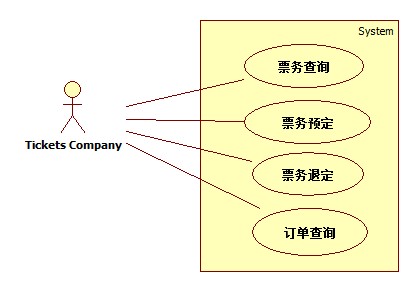


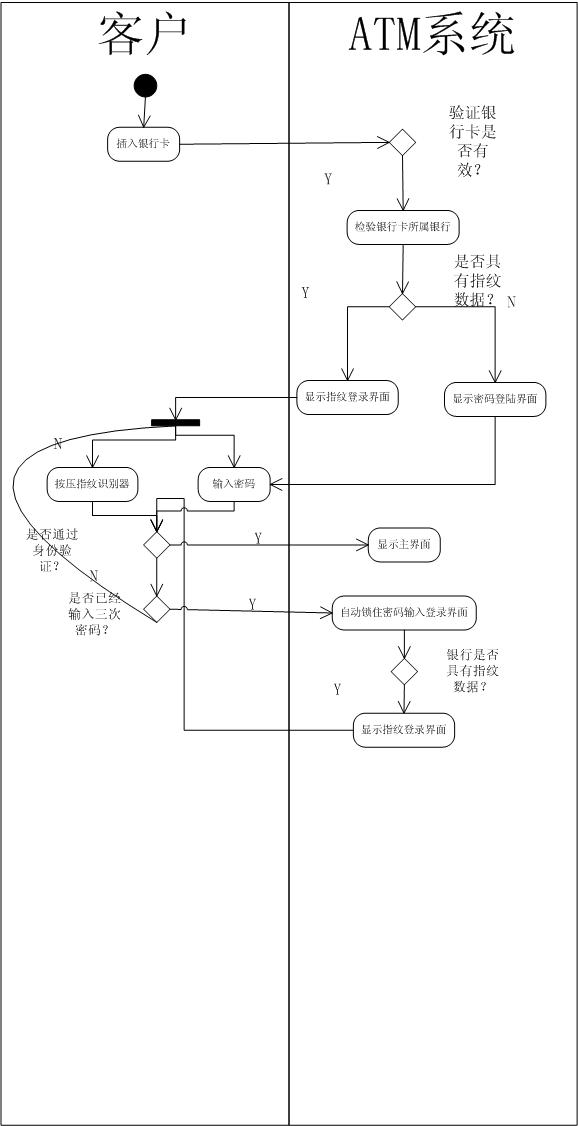
图4-6 Ticket Company Use Cases

**用例描述**

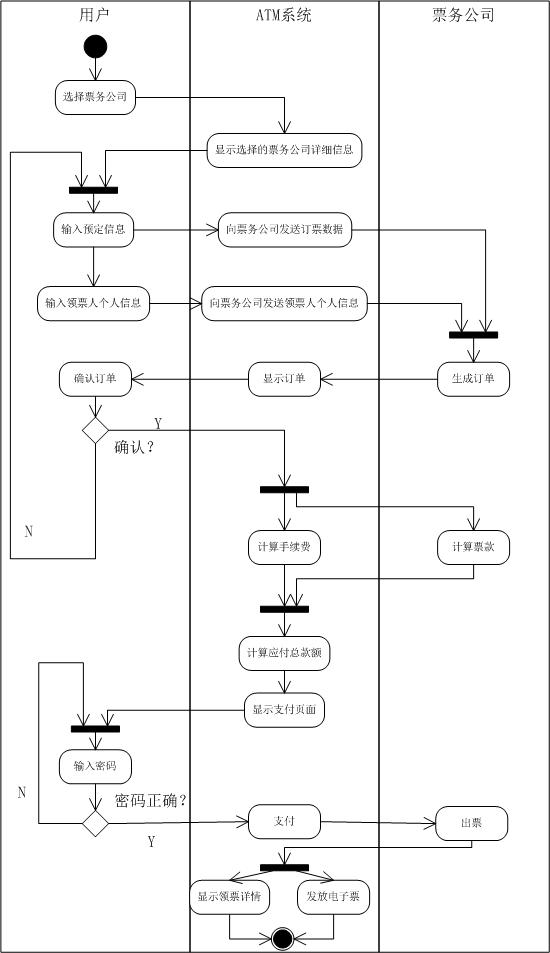
因为票务公司用户所能进行的操作与普通用户所能进行的票务操作完全相同，所以村务公司用例请见客户用例部分的用例描述。

## 2.2 分析模型

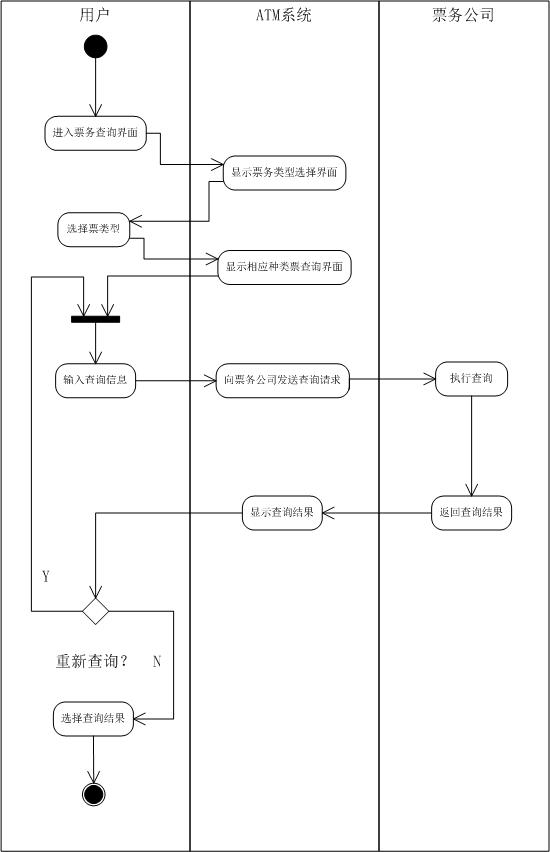
### 2.2.1 登录活动图



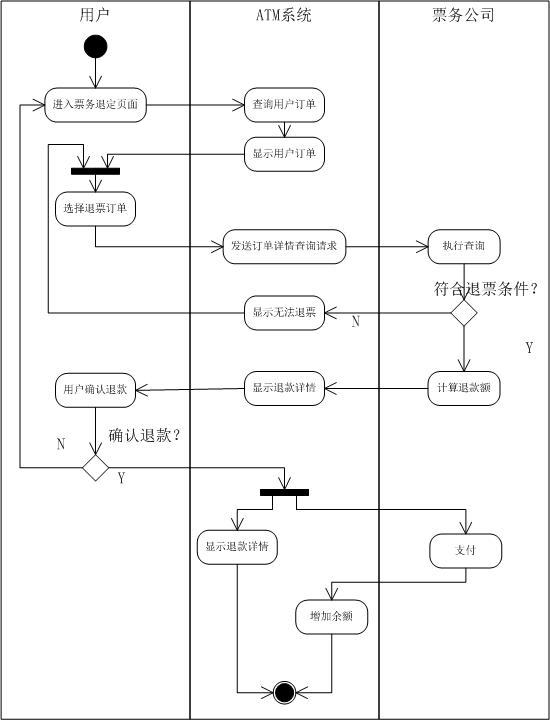
### 2.2.2 订票活动图



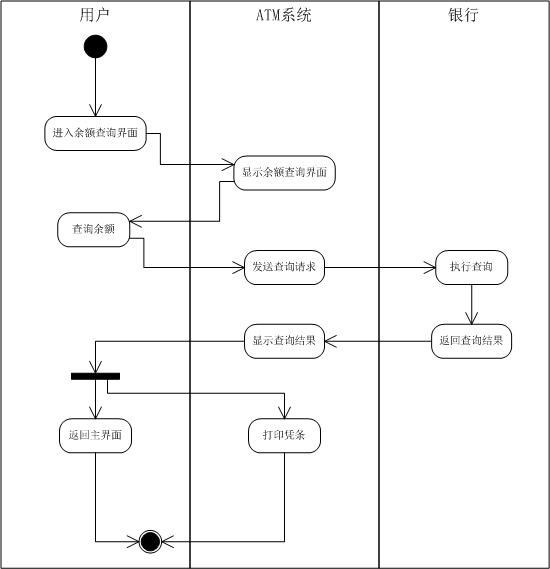
### 2.2.3 票务查询活动图



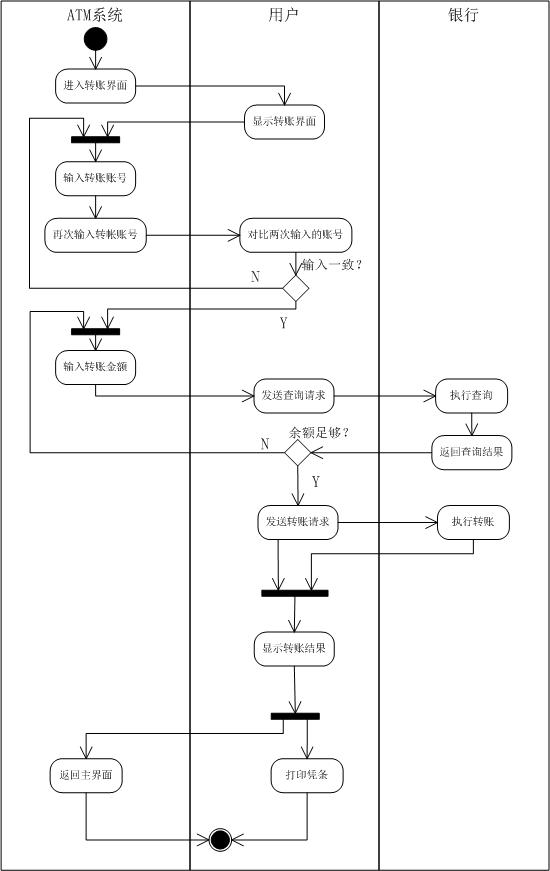
### 2.2.4 退票活动图



### 2.2.5 余额查询活动图

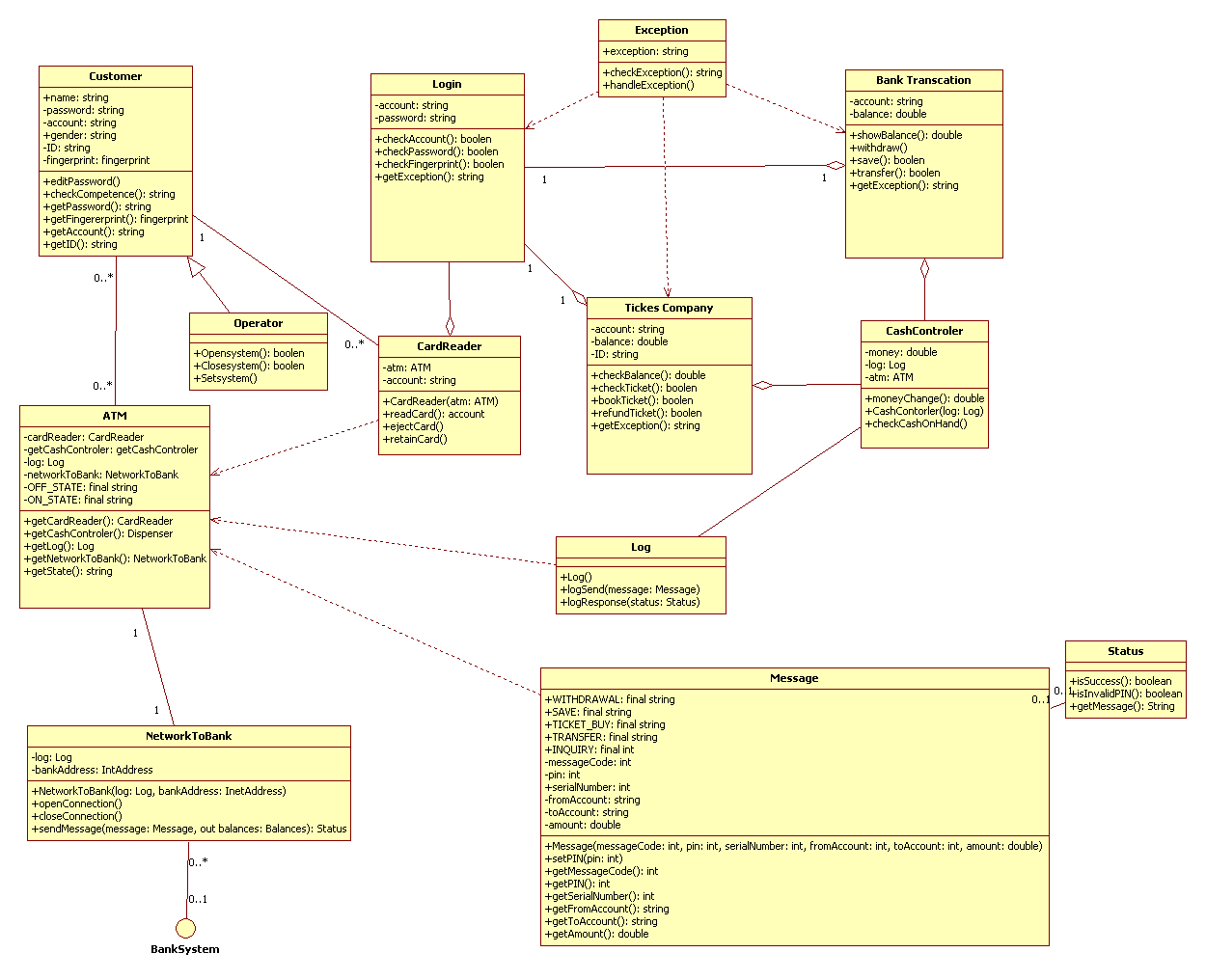


### 2.2.6 转账活动图



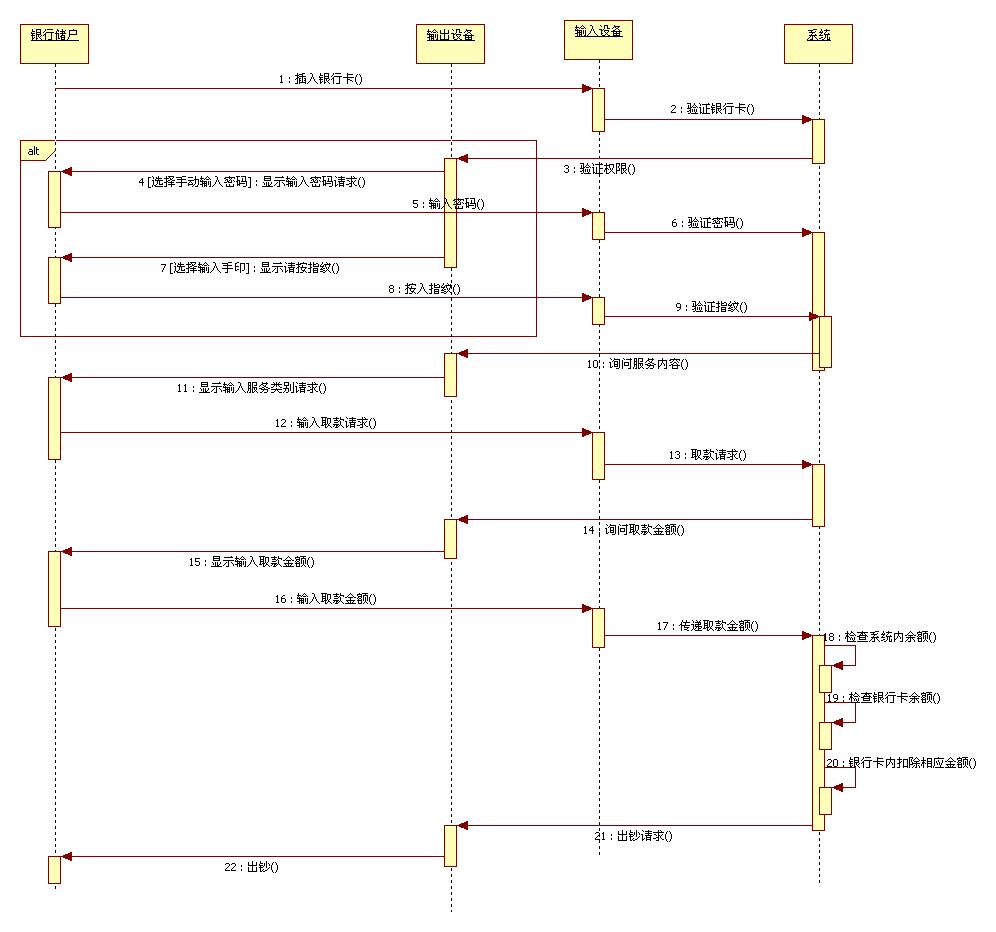
# 三、设计模型

## 3.1 静态类图

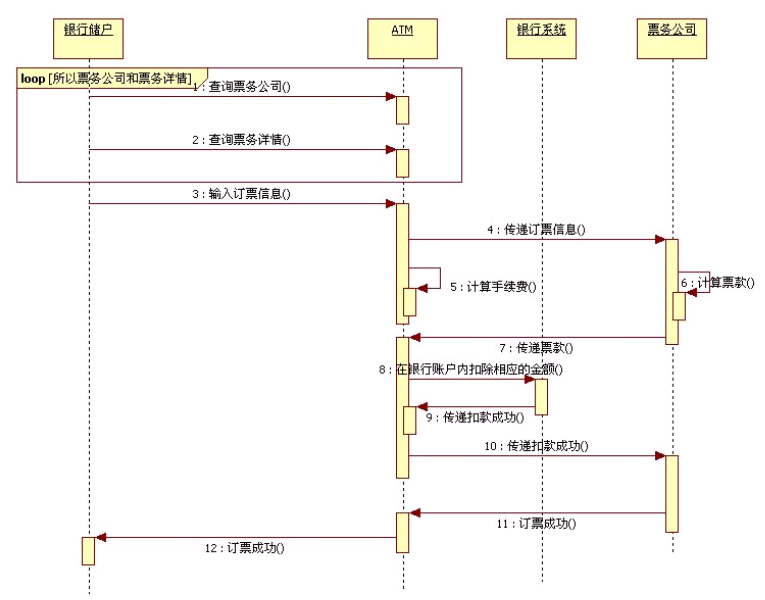


## 3.2 序列图

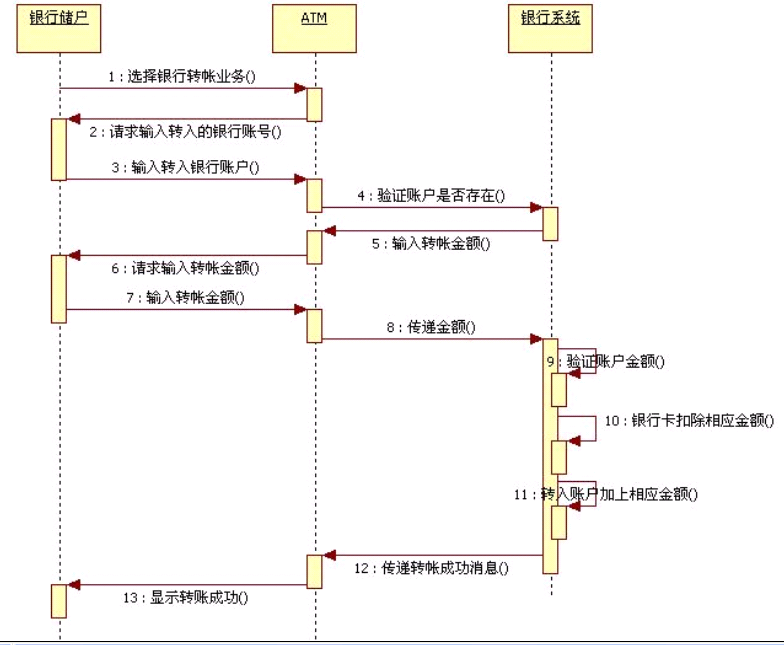
### 3.2.1 取款序列图



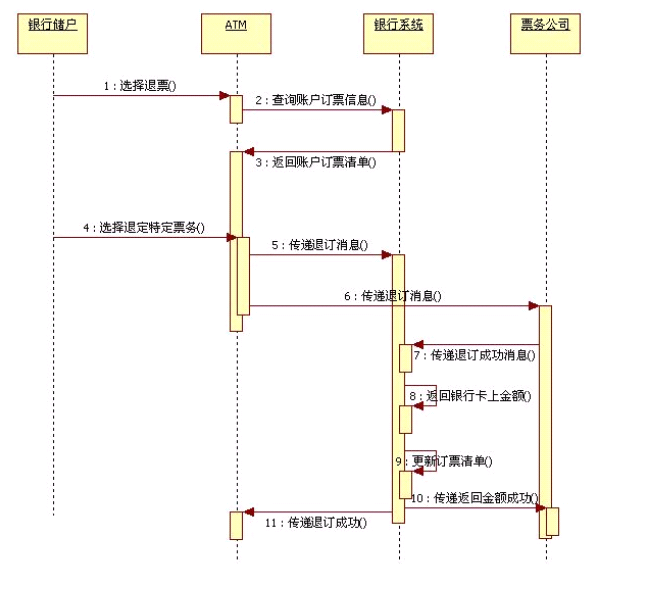
### 3.2.2 订票序列图



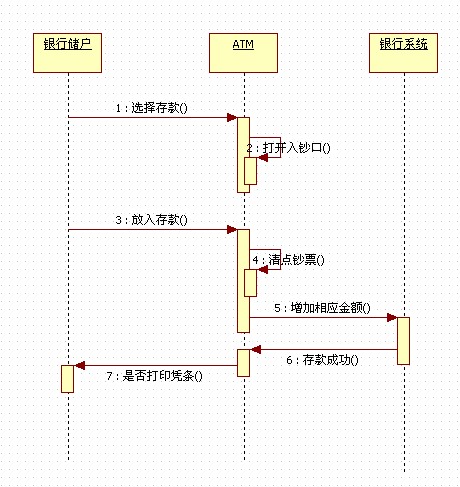
### 3.2.3 转账序列图



### 3.2.4 退票序列图

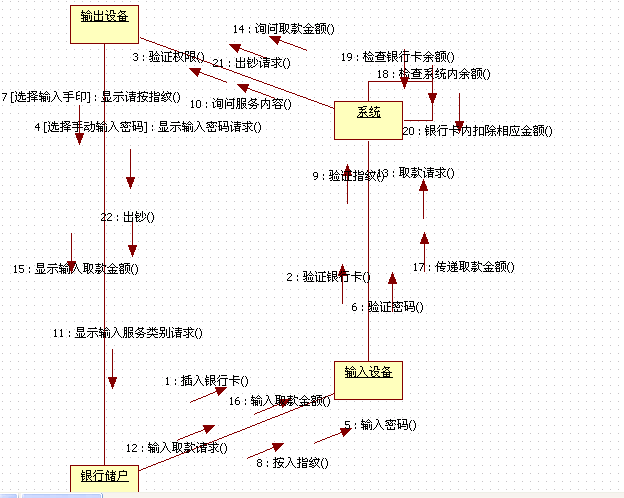


### 3.2.5 存款序列图

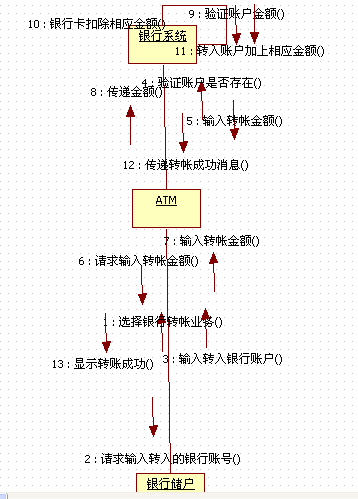


## 3.3 通讯图

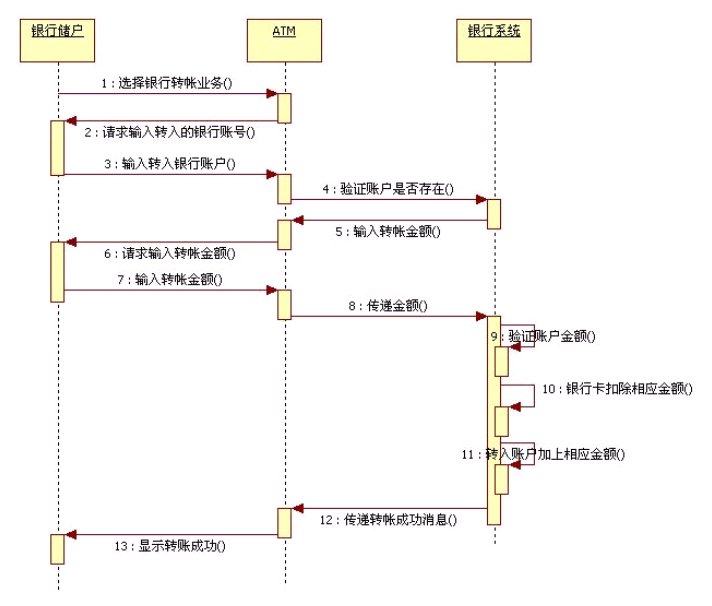
### 3.3.1 取款交互图



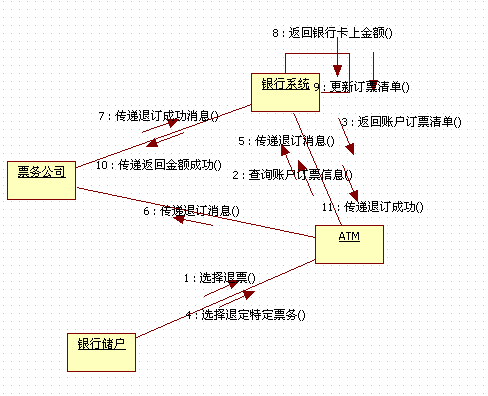
### 3.3.2 订票交互图



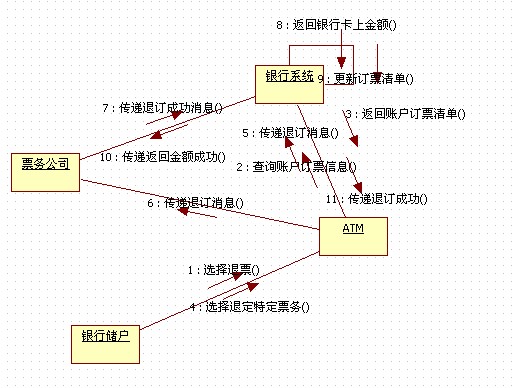
### 3.3.3 转账交互图



### 3.3.4 退票交互图



### 3.3.5 存款交互图



# 四、项目架构文档

## 4.1简述

### 4.1.1目的

本部分将从架构方面对系统进行综合的描述。本部分将使用不同的架构视图来描述系统的各个方面，同时记录表述已对系统架构方面做出的重要决策，向开发者传达正在构建的系统蓝图。

通过编写不同视图的文档说明并捕获每个部分的开发，本部分向开发团队和业务及 IT 参与者传达有关该不断发展的系统的信息。软件架构具有一组其预期要满足的业务和工程目标。体系结构部分可以向参与者传达这些目标将如何实现。

### 4.1.2范围

本部分针对新型触摸式ATM自助购票系统，从软件架构方面对系统做出全面的描述与设计。整个触摸式ATM自助购票系统按层划分为数据层，业务逻辑层与表现层，层与层之间提供服务与接口。每层之中包含子层与包，按功能将系统划分为具体的模块。本部分文档针对本系统的这种架构从架构视图的不同方面对整个系统做出了不同的描述。

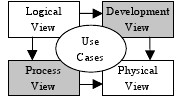
### 4.1.3概述

在本部分文档中，首先概要的描述了本文档的目的与使用范围。然后描述了系统对架构有重要影响的需求与目标。然后分别通过用例视图，逻辑视图，部署视图，实现视图等方面来描述系统的架构。最后从安全与性能方面对系统做了进一步的约束。

## 4.2架构的表示

新型触摸式ATM自助购票系统，按表现，逻辑，数据三层部分将整个系统划分为数据支持层，业务逻辑层，表现层三个大的层次。每个大层中又划分了不同的子层与不同的包。我们按照UML建模的方法，使用面向对象的思想，将整个系统按层次，按模块，从上到下，从粗到细的一步步细化与明确。

本系统的架构具体遵循了“4+1”的框架，此框架提供了软件架构的一组视图。



本系统软件架构文档以“4+1”框架对整个文档时行了组织。具体见下表：

|  |  |
| --- | --- |
| **4 +1 视图** | **文档中对应部分** |
| 用例 | 用例视图 |
| 逻辑视图 | 逻辑视图 |
| 进程视图 | 无 |
| 实施视图 | 实施视图 |
| 物理视图 | 部署视图 |

### 4.2.1系统功能划分

本部分使用用例视图来进行表述。用例视图是外部用户所能观察到的系统功能的模型图。用例模型列出了新型ATM触摸售票系统中的用例和参与者，并显示哪个参与者参与了哪个用例的执行。用例视图显示了在构架方面具有重要意义的用例模型的子集，以及用例和主角的子集。随需求之后进行的分析、设计和实施等活动都以构架这个概念为中心。

### 4.2.2系统层次与模块划分

本部分使用逻辑视图来时行表述。逻辑视图是为了便于理解新型ATM触摸售票系统中设计的结构与组织。系统只有一个逻辑视图，该视图以图形方式说明关键的用例实现、子系统、包和类，它们包含了在构架方面具有重要意义的行为。本部分包含了构架方面具有重要意义的部分，即类、子系统、包和用例实现的子集。

### 4.2.3系统物理分部

本部分使用部署视图来进行表述。部署视图是为了理解新型ATM触摸售票系统在一组处理节点上的物理分布。系统只有一个部署视图，它以图形方式说明了处理活动在系统中各节点的分布，包括进程和线程的物理分布。

### 4.2.4为实施制定的构架决策

本部分使用实施视图来时行表述。本部分用于用于：估算要开发、修改或删除的代码数量；阐明大规模复用的理由；考虑发布策略。

## 4.3架构的目标约束

本系统中，对整个系统的架构有重大影响的软件需求与目标有以下所示：

**安全性**：新型ATM触摸自助购票系统对安全性的要求很高。用户的帐户信息，密码信息，指纹数据等都有很高的价值，而购票信息涉及隐私，所以，系统最重要的功能与特性就是用户数据的安全性。安全性的涉及与关切贯穿了系统架构设计的整个过程。

**数据接口：**因为系统与总行，分行的数据库时行交互，同时系统还可能进行跨行的操作。同时，售票模块还要与售票公司的数据库进行交互，所以整个系统的数据接口很多，这就决定了系统要将与数据处理有关的层次但列出来，向应用层提供一套统一的接口来进行数据的处理，而消息的转发决策等具体处理过程则在数据服务层中实现。

**性能要求：**系统响应快速与简洁是新型ATM触摸自助购票系统的核心设计理念之一，所以，系统的响应速度必须达到很高的水平。这就要求系统的架构设计要较大比重的考虑性能问题。

**ATM功能与售票功能交互与共存：**本系统不是单纯的ATM机或者是售票机，而是将两种功能集成到了一台终端机上。而两种功能并不是简单的共存，而是有业务上的交叉与数据上的共享与交互。比如用户购票付款时的帐户默认为ATM系统当前登录的用户。怎样在保证功能实现的前提下，将数据安全性，保密性，性能等问题很好的解决也是架构设计的主要约束条件之一。

## 4.4用例视图

本节展示用例模型中的一些用例。这些用例体现了本ATM系统最重要，最核心的功能。这些用例使用了许多的架构元素。

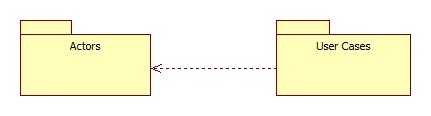


图4-1 Top Level Package

### 4.4.1用户

本小节列出了在所有用例场景中与系统交互的用户类型。

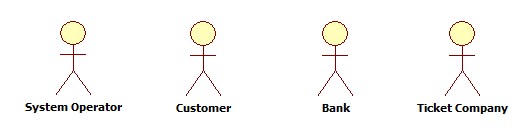


图4-2 Actors

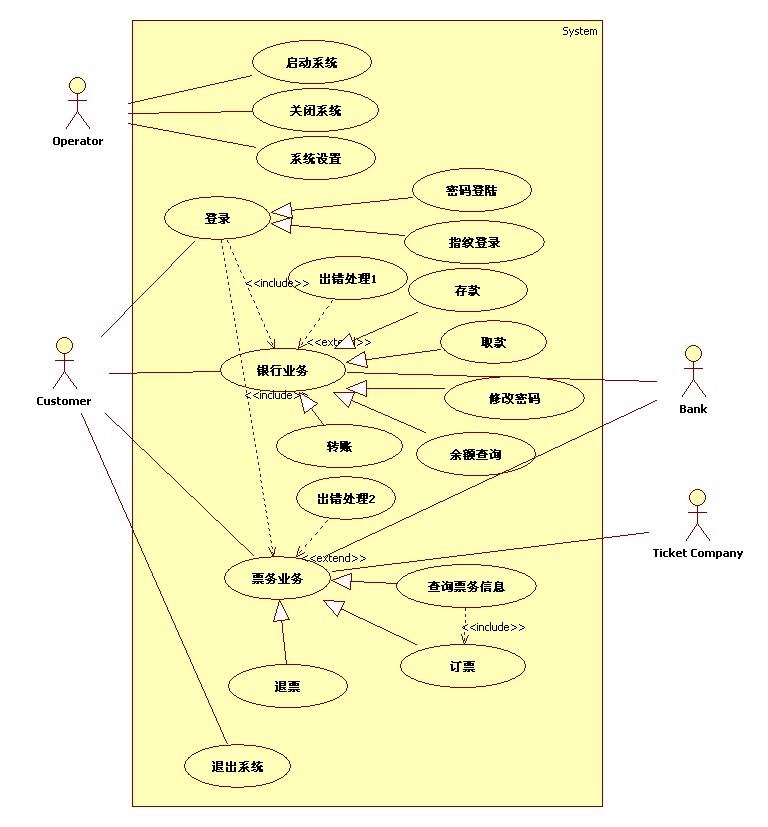
**System Operator:** 系统操作员。负责系统的启动，关闭，系统设置等功能。

**Customer:** 系统用户。通过与系统交互，进行银行业务与票务业务。

**Bank:** 银行用户。负责处理本系统与银行方面交互的接口业务。

**Ticket Company:** 票务公司。负责处理本系统与票务方面交互的业务。

### 4.4.2 用例图



## 4.5逻辑视图

### 4.5.1概览

本系统使用面向对象的方法，对系统进行分层，定义逻辑接口。由上至下呈现了从数据层到业务逻辑层到表现层属性及其关系。本系统顶级分层架构如下所示：

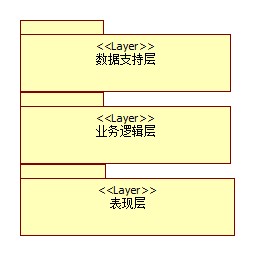


图5-1 系统顶级分层图

**数据支持层**：



本层负责将业务逻辑层传来的数据传递给各数据库，并从各数据库中读取数据。同时，本层还负责消息转发的决策。总行数据库包负责系统与总行数据库的交互。支行数据库负责系统与支行数据库的交互。票务数据库包负责与票务公司的数据库进行数据的交互。而消息转发决策系统则负责将从业务逻辑层传来的数据根据其目的地转发给相应的管理子层包。

**业务逻辑层**：



本层负责与数据支持层交互，时行数据的传递。同时负责与表现层时行数据的交互。而用户身份验证，银行业务，票务业务等核心功能的业务逻辑都在本层实现。业务功能子层中，业务的受理包具体负责业务的验证与受理。而业务数据交换则负责将上层或下层的数据转发出去。而指纹分析则负责将采集到的指纹数据与帐户对应的指纹数据进行比对。在业务接入层中，密码登录，指纹登录负责具体与密码登录，指纹登录相关的事务与逻辑。而个性服务与出钞服务负责票务处理，密码修改，钞票的验证与出钞等具体业务。

**表现层：**



本层负责将系统提供的各种功能以简明，易懂，安全的方式与用户交互。同时与业务逻辑层进行数据的交互。数据采集层的各包负责采集各种数据。而UI层的各包负责界面的表现与描述。

### 4.5.2对架构重要的模型元素

本系统静态类图如下所示：



#### 4.5.2.1数据支持层

改包的功能是建立数据库和应用程序之间的联系，而我们本项目采用的是Java开发，故在本层中我们采用了Hibernate 的架构模式。通过Hibernate的配置文件与数据库中的表进行对应，从而是数据库中的表转换成程序中的一个对象进行操作。而在银行数据库方面，我们采用接口的形式与银行数据库进行连接，根据对方的数据库类型自动配置本地数据库参数，从而实现本地数据库连接与银行数据库解耦合。

##### 4.5.2.1.1重要类简述

(1)数据管理子层：

1.本层中最重要的类是NetworkToBank，其详细的方法与属性介绍在静态类途中已经说明。其主要职责是负责将数据库中的表转换为程序中的对象。



数据服务子层：

1.本层中重要的是一个Hibernate.xml文件，负责数据库数据流向的配置，是银行数据库中的数据持久化到程序中的对象，并且配置该对象的范围，名称，类型等等。

#### 4.5.2.2业务逻辑层

改包的功能是处理前台传过来的数据，并且根据业务流程来处理数据，然后根据处理后的数据传到数据库进行交互以及接收数据库中的数据。改包主要采用了Struts和Spring的框架模式进行架构，使数据可以以对象的方式注入到其他对象中，以及通过Struts—Config.xml文件和前台数据进行绑定，是前台数据绑定到对象中，再通过Spring注入到数据库的对象中，最终通过Hibernate传入到数据库中。而数据库传过来的数据通过相反的操作进行和前台绑定。

##### 4.5.2.2.1重要类简述

(1)应用平台子层：

1.本层中的最重要文件是3个配置文件，主要用来配置Hibernate，Struts，Spring3个框架的信息，是程序中的类可以按照正常的业务逻辑进行实例化以及值的传递。

(2)业务功能子层：

1.本层重要的类是CashContorler，其详细的方法与属性介绍在静态类途中已经说明。主要功能是用户在操作界面提交操作信息以后，通过该类对数据进行判断其是否能执行该操作，以及对数据进行业务逻辑操作。最后以消息的形式通知业务流程下一个对象接受数据。



2.本层重要的类是CardReader，其详细的方法与属性介绍在静态类途中已经说明。主要功能是对用户持有卡的信息进行处理，通过对卡的信息中的用户名，密码，指纹信息等信息进行业务逻辑判断与数据的处理，并以信息的形式传递业务逻辑的处理结果通知业务流程下一个对象接受数据。



3.本层重要的类是Message，其详细的方法与属性介绍在静态类途中已经说明。主要功能是该系统中各个类之间进行业务流程处理的时候，通过该类传递一些信息，以提示下一个处理的流程的类名，处理方法，数据类型，数据表等等重要信息，是整个业务流程能连贯的执行下去。



(3)业务接入子层：

1.本层重要的类是BankTrascation，其详细的方法与属性介绍在静态类途中已经说明。其主要功能是与用户操作界面数据绑定的类进行交互，利用Message中传递的信息识别自己要交互的类名，数据名，数据类型等信息，然后和该对象交互数据，是的前台数据可以传入后台进行处理，同时是后台处理好的数据能传到前台进行显示。



#### 4.5.2.3表现层

改包的功能是用户通过设计好的UI界面执行操作与数据的录入，而系统接收到数据进行最初的数据封装。在本包中，我们用户界面采用了Java中的GUI开发，是用户可以形象的通过按钮和数字键盘等形式输入指令和数据。而后台通过相应的JavaBean文件对数据进行接收和对后台反馈的数据进行接收。然后通过Message与业务逻辑层进行交互。

数据采集层:

1.本层重要的类是BankTrascationUI，其详细的方法与属性介绍在静态类途中已经说明。该类就是用户通过设计好的交互页面，使得用户输入的指令绑定到该类中，然后该类通过消息的形式通知业务逻辑层中相应的类接收数据，并且从业务逻辑层传回到页面的数据也在该类中绑定。

## 4.6实施视图

## 4.7部署视图

## 4.8大小与性能

**响应时间：**用户点击按键或触摸屏时，平均响应时间<0.1s。执行查询，取款，订票等需要数据交互的业务时，平均响应时间<3s。

**并发性能：**系统数据服务部分支持1000台终端机同时操作。

**安全性能：**系统保证用户帐户数据的绝对安全性。用户的个人信息与身份验证信息不会泄露或者遭到破坏。

**易用性：**系统界面简洁，操作方法通俗易懂。

## 4.9质量

**数据接口复杂主工的解决：**因为系统与总行，分行的数据库时行交互，同时系统还可能进行跨行的操作。同时，售票模块还要与售票公司的数据库进行交互，所以整个系统的数据接口很多。所以在本系统的架构设计中，我们单独列出了一层数据支持层来进行数据的处理，此层将数据处理的细节与业务逻辑层剥离开来，而只提供了一组通用的接口。这样，业务逻辑层在编程时就不需要考虑复杂的信息转发决策与及与不同数据库之间交互时的细微差别。而在数据支持层中，我们将消息的转发决策功能与数据库交互功能又进行了划分。

**安全性在架构设计中的关切：**新型ATM触摸自助购票系统对安全性的要求很高。用户的帐户信息，密码信息，指纹数据等都有很高的价值，而购票信息涉及隐私，系统架构中，使用专门的模块对用户身份验证进行处理，此模块与其他模块几乎没有数据的交互。所有用户个人信息与帐户信息都相当于在一个黑盒之中处理，不仅对外不公开，对系统中其他的模块也是不可见的，这样保证了安全性。

**架构设计对性能要求的考量：**系统响应快速与简洁是新型ATM触摸自助购票系统的核心设计理念之一，所以，系统的响应速度必须达到很高的水平。在系统架构的涉及中，分层，分模块，层与层之间，模块与模块之间的数据交互量很少，且频率比较底。而模块之间的耦合度比较底，降低了实现时的复杂度。而在性能上也有了保证。

**ATM功能与售票功能交互与共存对架构的要求：**本系统不是单纯的ATM机或者是售票机，而是将两种功能集成到了一台终端机上。而两种功能并不是简单的共存，而是有业务上的交叉与数据上的共享与交互。系统分层分模块的将票务业务与ATM业务剥离开，两种不同功能划分到了不同的模块之中，而模块间没有直接的交互，数据的共享是通过另一个管理模块来实现，这样保证了数据独立性与完整性。

# 五、组员分工

这次文档是工作量最大的一次，主要是SAD部分。我们团队仍然采取分别学习、集中讨论的形式，使得团队中每一个成员对每一个模块的设计都非常的了解，在经过反复商讨和确认每一部分的内容后，我们采取平均分工的原则，完成了整个文档的撰写，并从中受益匪浅。