## 一、实验目的及要求

一个银行系统的需求建模、需求分析、架构设计、详细设计的全过程，掌握客户机/服务器模式的软件系统分析和设计方法。熟练使用 UML 语言及其相关建模方法，熟练使用 StartUML 建模工具进行相关设计开发。

## 二、实验设备（环境）及要求

实验环境：

1．PC 机最低配置：2G Hz 以上 CPU；1G 以上内存；1G 自由硬盘空间

2．StartUML

实验要求：

1． 详细阅读实验步骤，动手使用 StartUML 绘制相关模型图；

2． 按照实验步骤概要地介绍实验过程，并粘贴用 StartUML 绘制相关模型图；

3．按思考题的要求，进行相关实验；

4．将实验报告（word 文档）命名为：学号+姓名+ex*.doc。

例如：2013308888 张三 ex1.doc

5．将实验报告上传至 ftp://10.13.0.10/课件/软件架构/实验提交

## 三、实验内容与步骤

实验内容：

1．安装 StartUML；

2．对“银行系统”开发用例模型，详细了解用例编档的方法；

3．面向问题域的静态建模；

4．识别系统中的各类对象；

5．动态建模；

6．子系统划分；

7．子系统设计；

8．数据库设计；

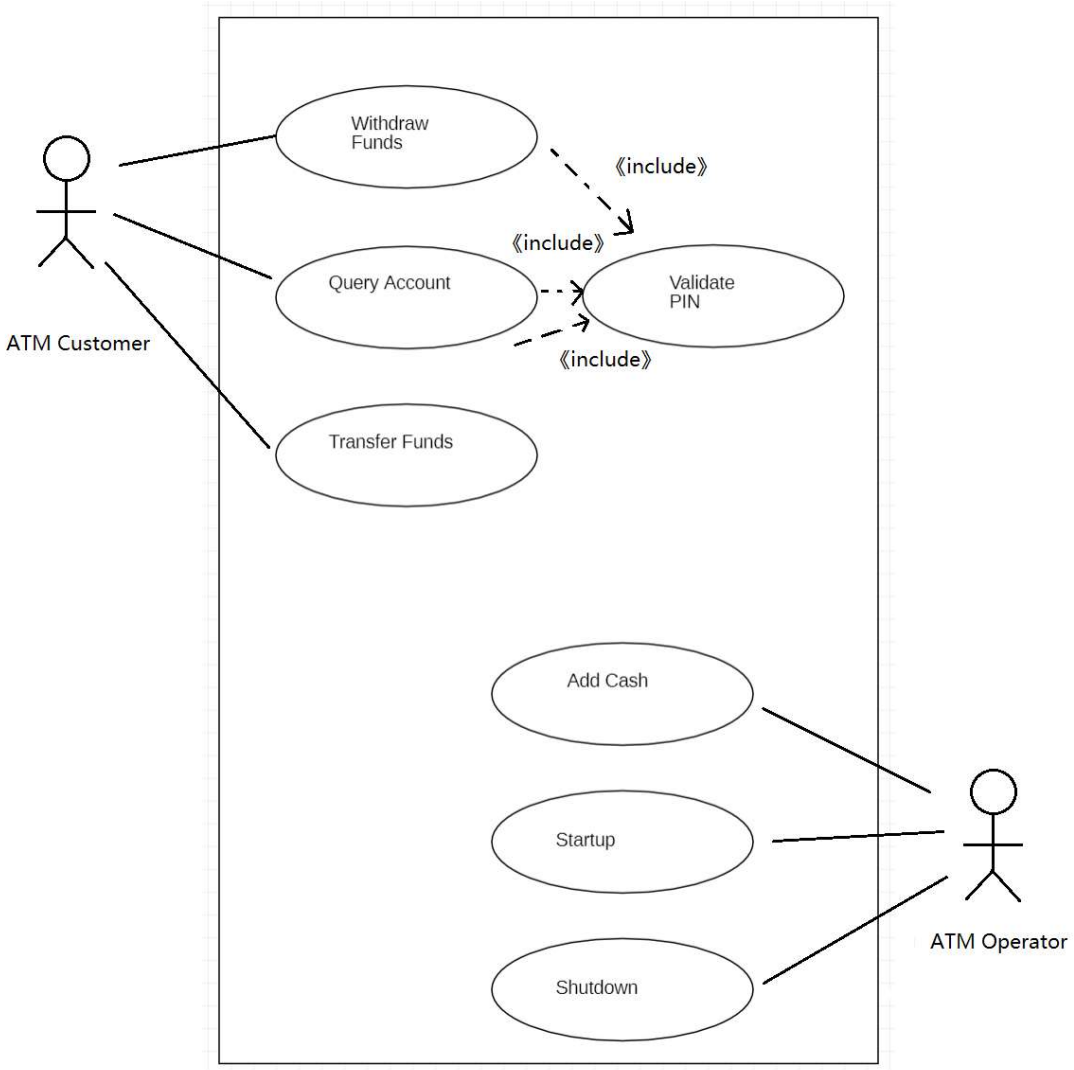
9．详细设计；

10．部署方式设计。

四、　　实验结果与数据处理

1.对“银行系统”开发用例模型，详细了解用例编档的方法；

将操作员用例分为三个用例。如图所示：

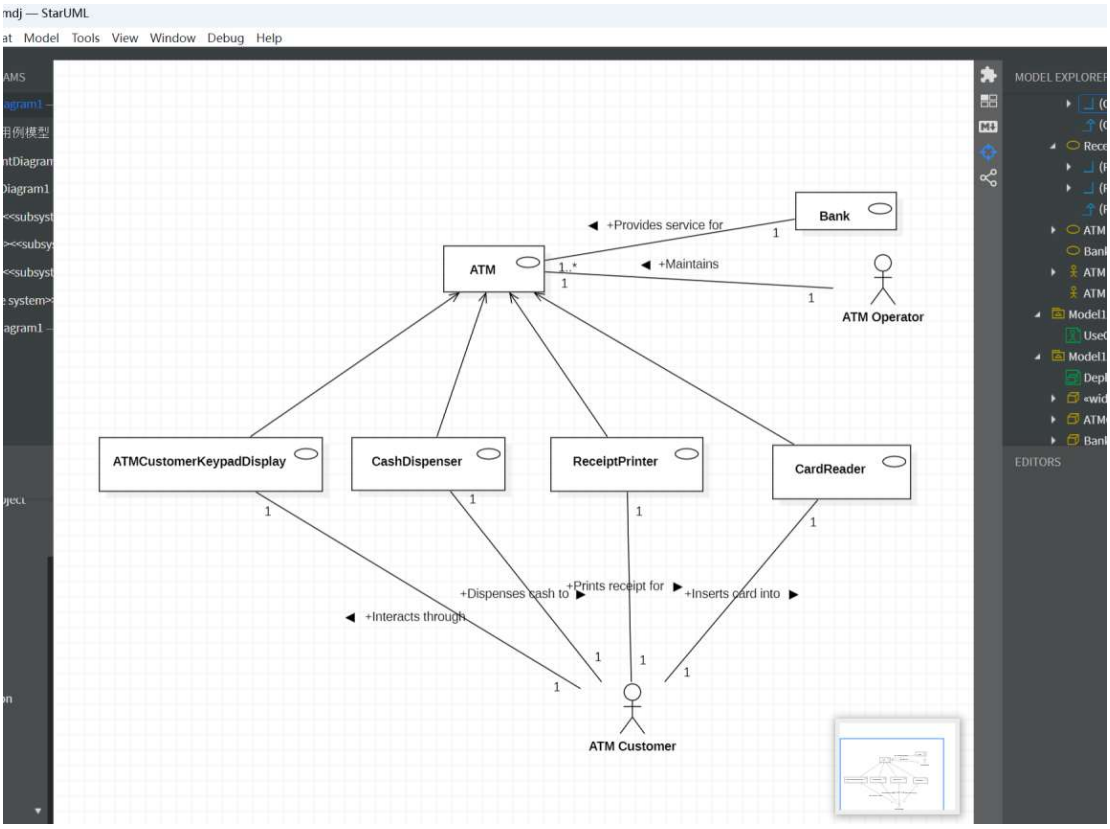



该用例模型中，参与者为 ATM Customer 和 ATM Operator。对于 ATM Customer 有 Withdraw Funds、Query Account、Transfer Funds 三个用例，其中，这三个用例具有一个公共部分 Validate Pin 为包含用例。ATM Operator 有三个用例，Add

Cash、Startup 和 Shutdown。

2.面向问题域的静态建模；

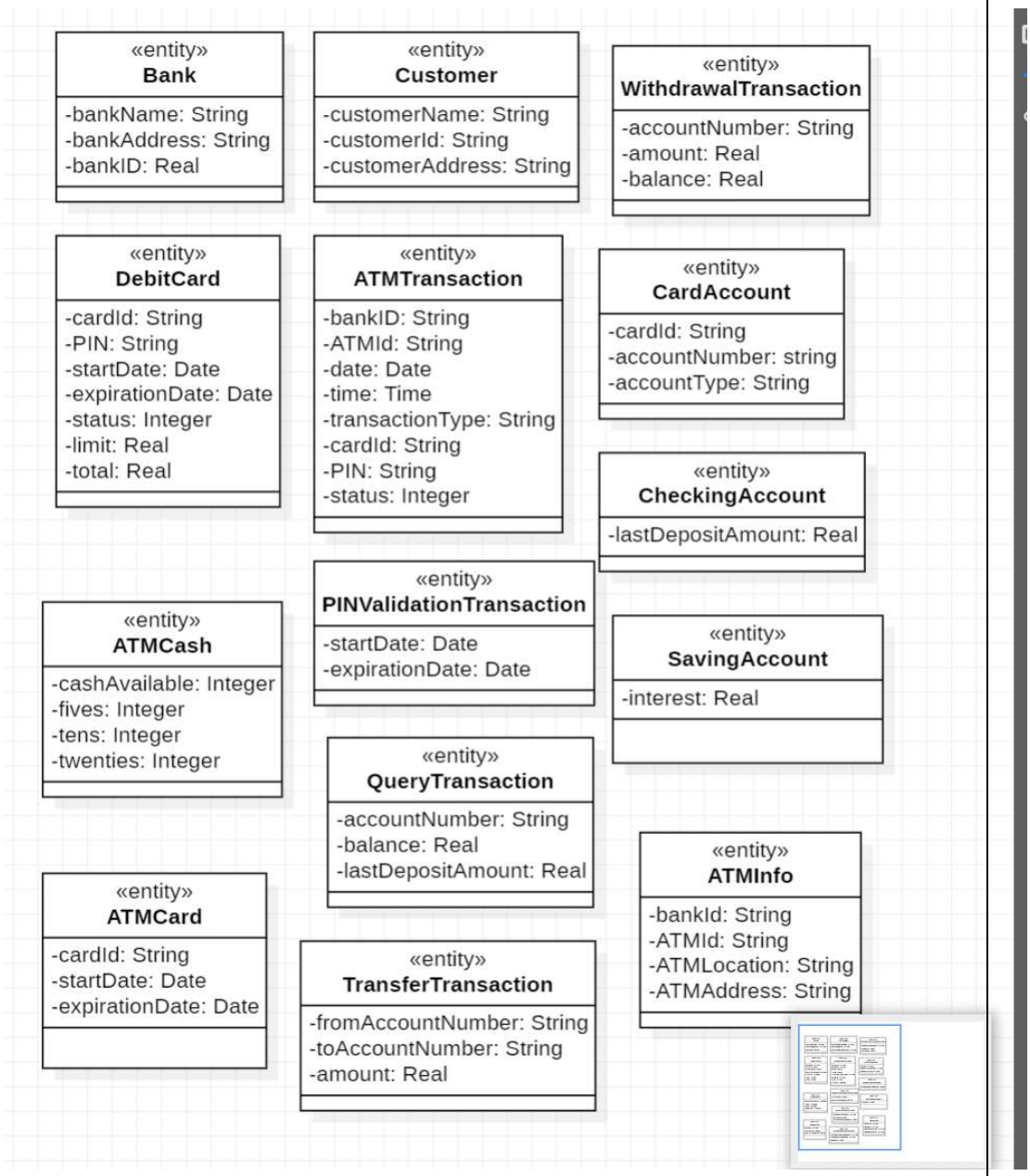
如下图所示：




问题域的静态建模如上图所示。一个银行有若干 ATM 机，每个 ATM 机为一个符合类，包含 Card Reader、Cash Dispenser、Receipt Printer、ATM Customer Keyboard Display。ATM Customer 将卡插入 Card Reader，通过 ATM Customer Keyboard Display 对系统的提示作出响应，Receipt Printer 打印凭条。

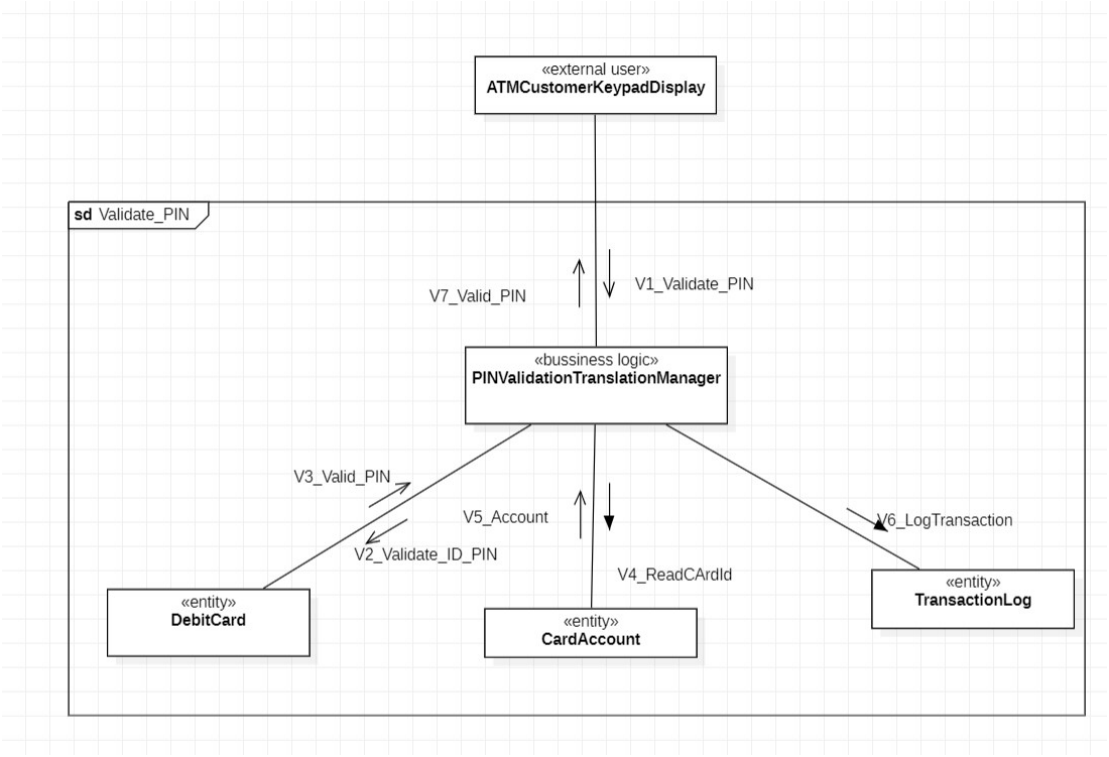
3.识别系统中的各类对象；

静态模型，即实体类模型如下图所示：

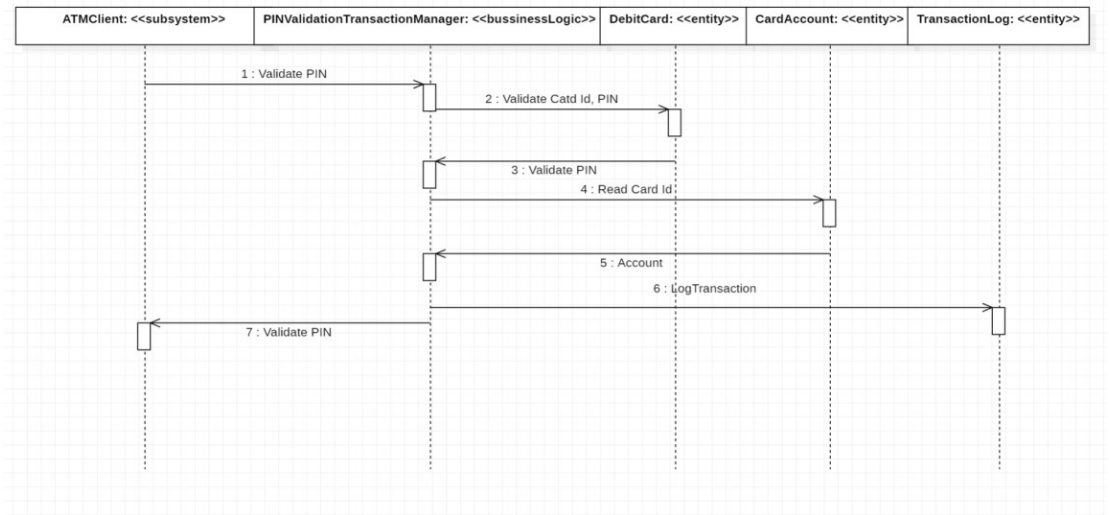



4.动态建模；

以服务器端验证 PIN 码为例，通信图如下图所示：


«external user»
ATMCustomerKeypadDisplay
sd Validate_PIN
V7_Valid_PIN
V1_Validate_PIN
«bussiness logic»
PINValidationTranslationManager
V3_Valid_PIN
V5_Account
V2_Validate_ID_PIN
V6_LogTransaction
V4_ReadCArdId
«entity»
DebitCard
«entity»
CardAccount
«entity»
TransactionLog

可画出其对应时序图：


ATMClient: <<subsystem>>
PINValidationTransactionManager: <<bussinessLogic>>
DebitCard: <<entity>>
CardAccount: <<entity>>
TransactionLog: <<entity>>
1 : Validate PIN
2 : Validate Catd Id, PIN
3 : Validate PIN
4 : Read Card Id
5 : Account
6 : LogTransaction
7 : Validate PIN

消息序列如下：

V1：“ATM 客户端”向“PIN 码验证交易管理器”发送“验证 PIN 码”请求。“PIN 码验证交易管理器”中包含判断 PIN 码是否有效的业务逻辑，即判断客户输入的 PIN 码是否与银行服务数据库中的 PIN 码相符。

V2：“PIN 码验证交易管理器”向“借记卡”实体对象发送一个“验证”（Validate）消息（包含“卡号”、“PIN”），要求其根据卡号和客户输入的 PIN 码来验证客户记卡是否有效。

V3:“借记卡”检查验证客户输入的 PIN 码是否与“借记卡”中记录的 PIN 码符合、卡的状态是否正常（无挂失）以及该卡是否过期。如果所有验证通过，“借记卡”向“PIN 码验证交易管理器”发送一个有效 PIN 码的响应。

V4:如果验证通过，“PIN 码验证交易管理器”向“卡账户”实体对象发送消息，要求其返回当前卡号可访问的账户号。

V5:“卡账户”返回有效的账户号。

V6:“PIN 码验证交易管理器”用“交易日志”（Transaction Log ）来记录交易日志:

V7 :“PIN 码验证交易管理器”向“ATM 客户端”发送一个“有效 PIN 码”的响

应.如果 PIN 码验证成功，还会发送账户号。

5.子系统划分；

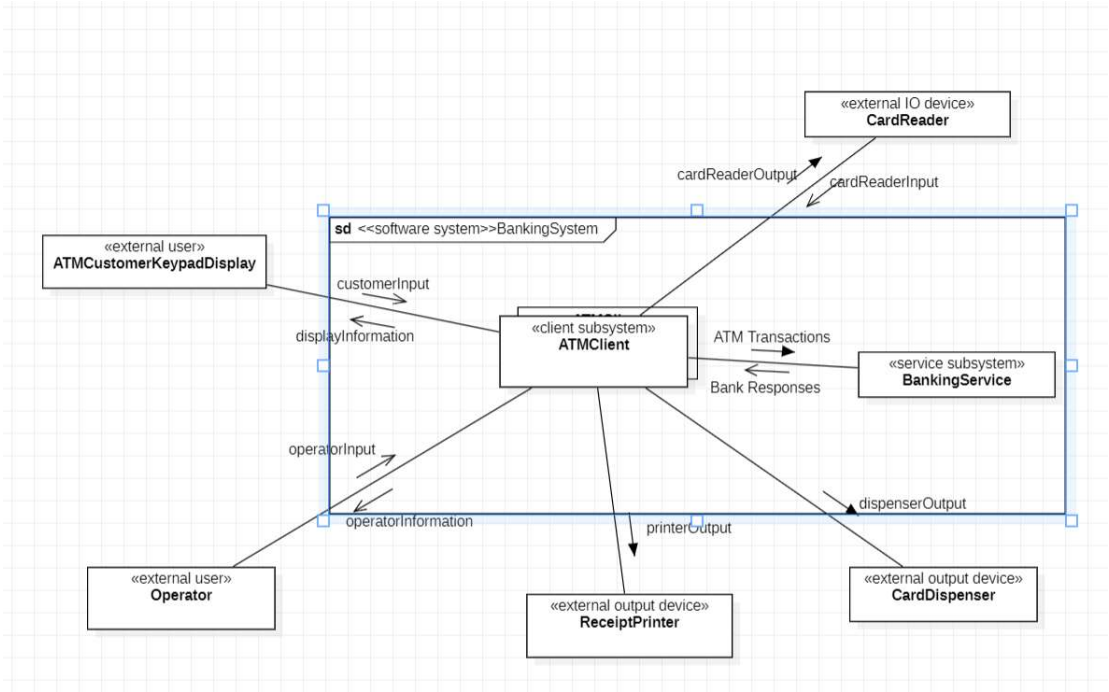
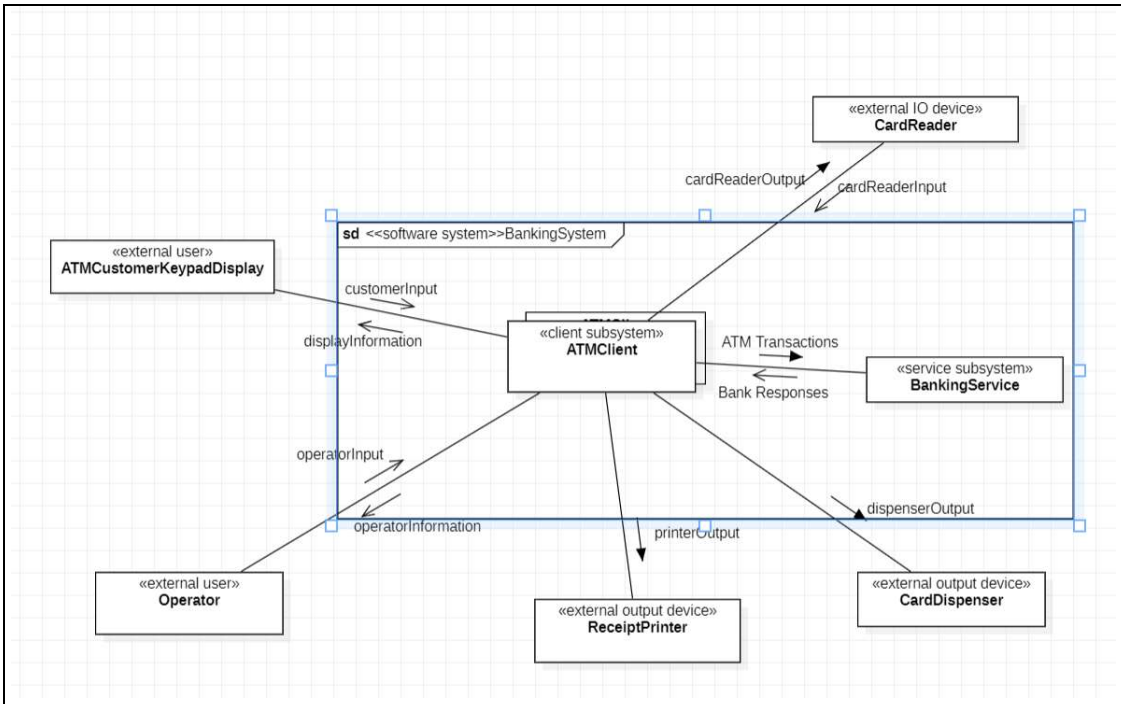


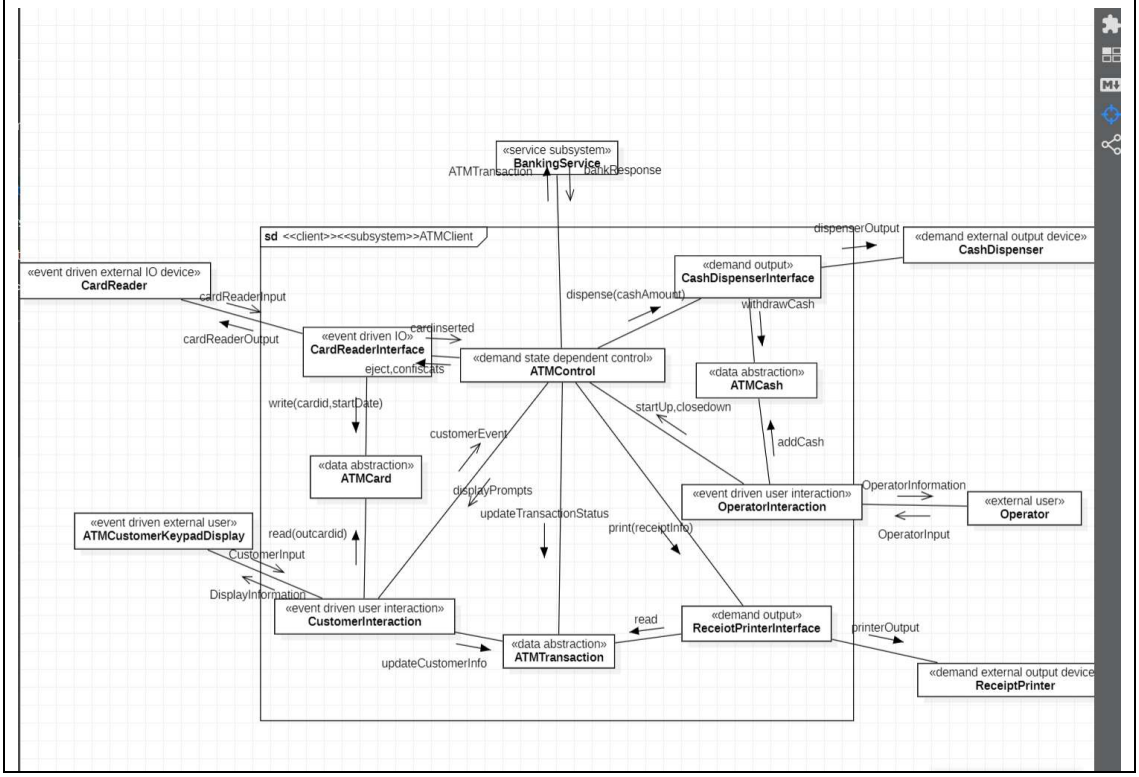

图 21-27 是一个描述两个子系统间简单消息传递的分析层面的通信图。“ATM 客户产子系统”向“银行服务子系统”发送“ATM 交易”消息，并得到一个“银行响应”。“ATM 交易”消息是一个包含了“PIN 码验证”(PIN Validation)、“取款”( Withdraw )、“查询( Query)、“转账”(Transfer )、“确认”( Confirm）以及“终止”(Abort）消息的聚合消 t“银行响应”是对这些消息的各种响应。

6.子系统设计；

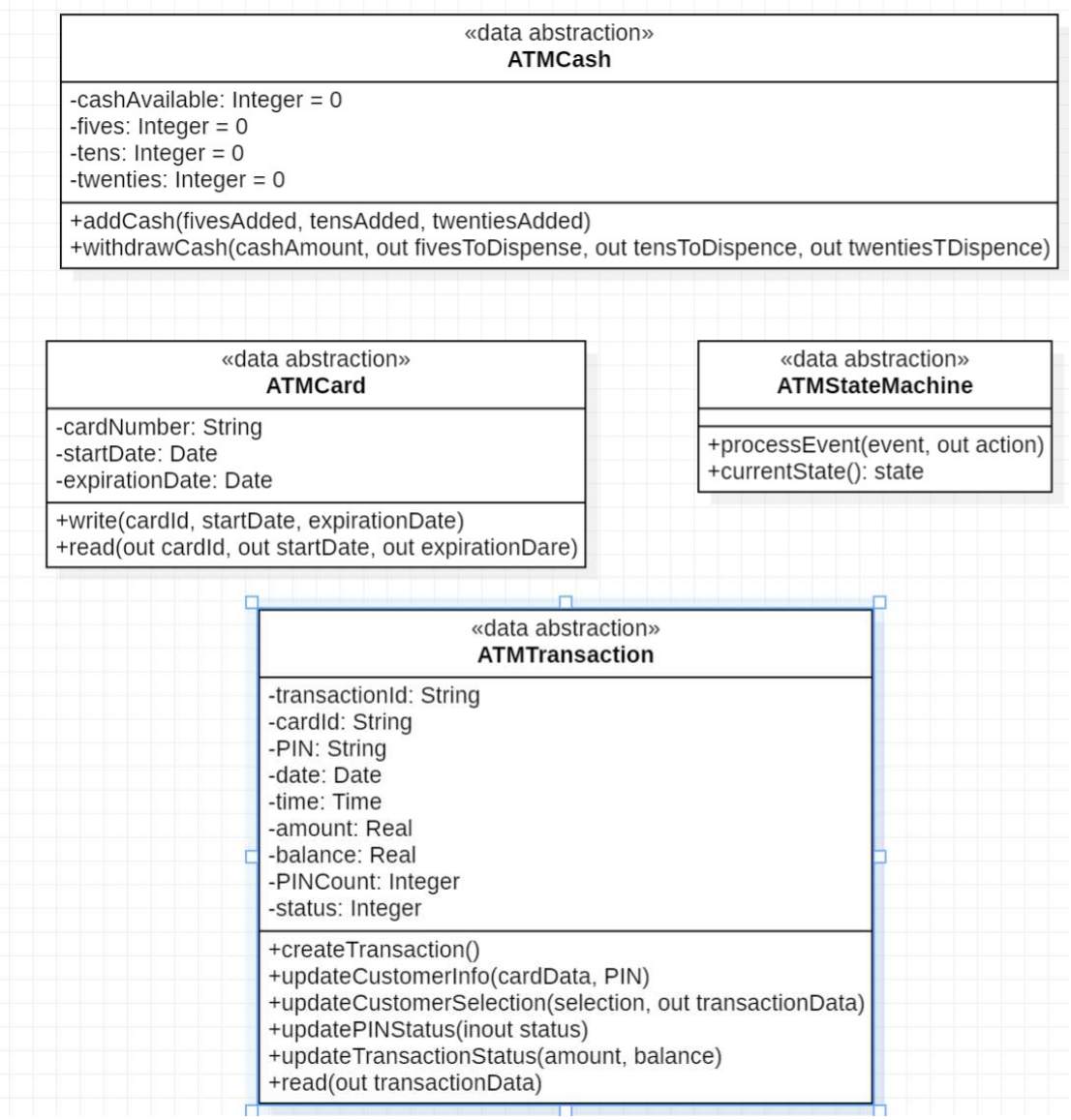



下图为银行系统的高层并发通信图。

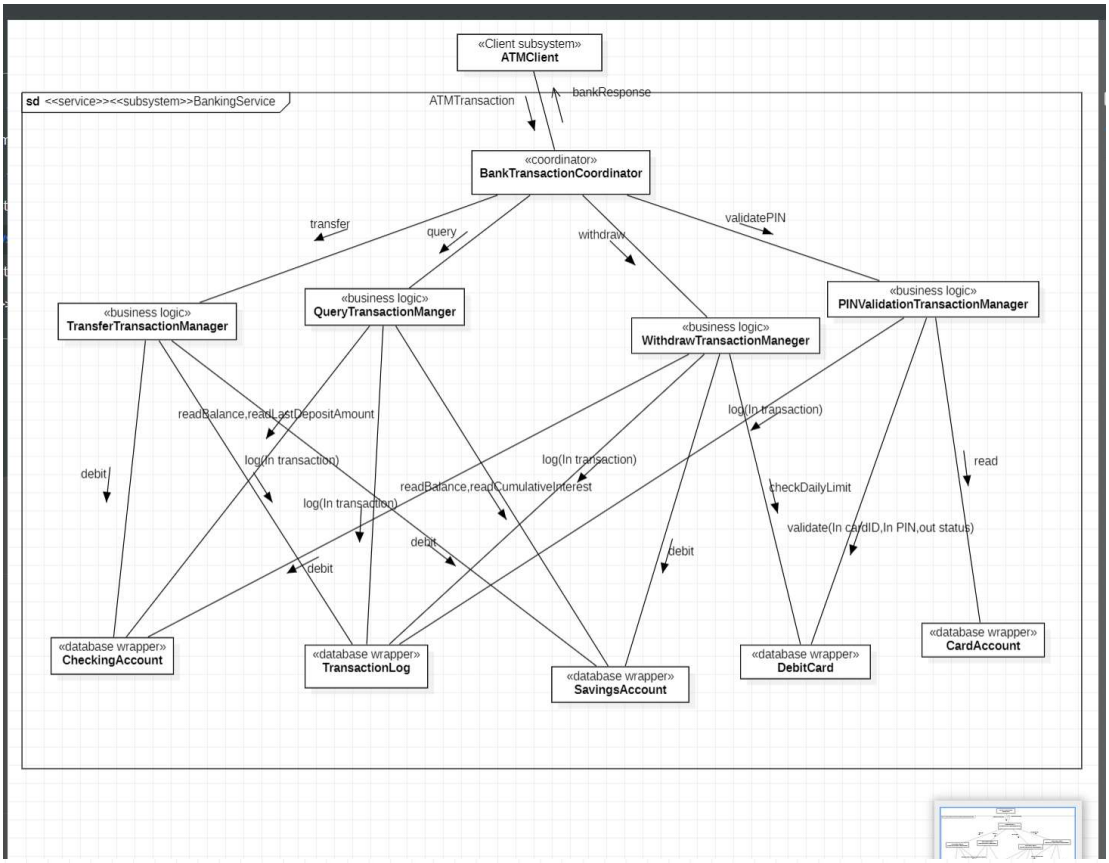
下图为 ATM 客户端子系统的并发通信图。

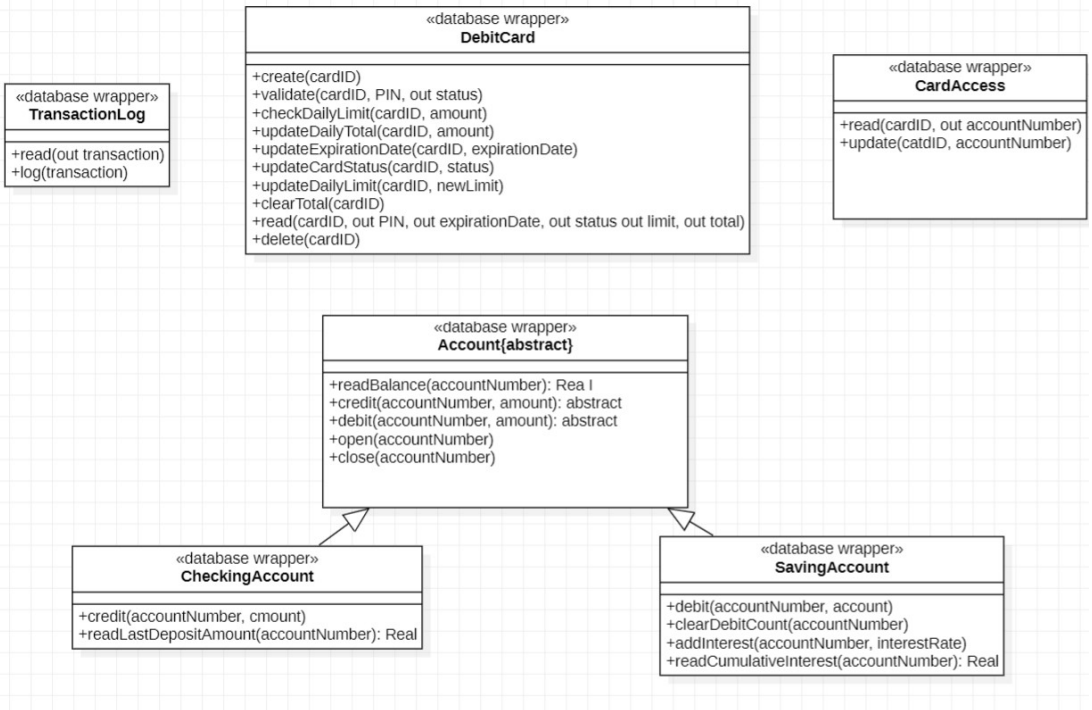


ATM 客户端的信息隐藏类如下图所示：




银行服务子系统、容器包装类如下图所示：


«Client subsystem»
ATMClient
sd <<service>><<subsystem>>BankingService
ATMTransaction
bankResponse
«coordinator»
BankTransactionCoordinator
transfer
query
withdraw
validatePIN
«business logic»
TransferTransactionManager
«business logic»
QueryTransactionManger
«business logic»
WithdrawTransactionManeger
«business logic»
PINValidationTransactionManager
readBalance,readLastDepositAmount
log(In transaction)
debit
log(In transaction)
readBalance,readCumulativeInterest
log(In transaction)
debit
debit
log(In transaction)
checkDailyLimit
read
validate(In cardID,In PIN,out status)
debit
«database wrapper»
CheckingAccount
«database wrapper»
TransactionLog
«database wrapper»
SavingsAccount
«database wrapper»
DebitCard
«database wrapper»
CardAccount


«database wrapper»
TransactionLog
+read(out transaction)
+log(transaction)
«database wrapper»
DebitCard
+create(cardID)
+validate(cardID, PIN, out status)
+checkDailyLimit(cardID, amount)
+updateDailyTotal(cardID, amount)
+updateExpirationDate(cardID, expirationDate)
+updateCardStatus(cardID, status)
+updateDailyLimit(cardID, newLimit)
+clearTotal(cardID)
+read(cardID, out PIN, out expirationDate, out status out limit, out total)
+delete(cardID)
«database wrapper»
CardAccess
+read(cardID, out accountNumber)
+update(catdID, accountNumber)
«database wrapper»
Account{abstract}
+readBalance(accountNumber): Rea l
+credit(accountNumber, amount): abstract
+debit(accountNumber, amount): abstract
+open(accountNumber)
+close(accountNumber)
«database wrapper»
CheckingAccount
+credit(accountNumber, cmount)
+readLastDepositAmount(accountNumber): Real
«database wrapper»
SavingAccount
+debit(accountNumber, account)
+clearDebitCount(accountNumber)
+addInterest(accountNumber, interestRate)
+readCumulativeInterest(accountNumber): Real

7.数据库设计；

对于“账户”泛化/特化层次结构，我们决定通过将“账户”（Account）父类的属性玄在“支票账户”（Checking Account）与“储蓄账户”（Savings Account）表中来实现层次结构的扁平化。虽然账户类型（储蓄账户或者支票账户）是“账户”类的一个属性，但是我{假定知道“账户号”后可以确定账户类型，因此“支票账户”和“储蓄账户”表的主键是accountNumber(账户号)。与之相关联的“卡账户”类（在图21-4中定义）被设计为一-张关联表，用来表示“卡”和“账户”之间的多对多关系。“客户账户”也被设计为一张关联表.用来表示“客户”和“账户”之间的多对多关系。即使在静态模型中没有“客户账户”这臼一个关联类（因为ATM交易中并不需要它)，在关系数据库中也需要这样一张表。

对于“ATM交易”泛化/特化层次结构，我们对其进行扁平化处理，但只提供交易子类的天系表。一个ATM交易的主键是交易号，它由多个字段的复合主键组成，包括: bankId（银行号ATMId (ATM 号 )、date(日期)、time(时间)。bankld 和ATMd 也是外键，因为由它们可以导三到“银行”表和“ATM 机信息”表。“ATM机信息”有一个复合主键，由bankId和ATMId组式.其中 bankld也是一个外键。日期和时间属性可以提供一个时间戳来唯一确定一笔交易。

Bank (bankName, bankAddress, bankId)

Customer (customerName, customerId, customerAddress)

Debit Card (cardId,PIN, startDate, expirationDate,status,limit,total,*customerId*)

Checking Account(accountNumber,accountType,balance,lastDepositAmount)

Savings Account (accountNumber, accountType, balance,

interest)

Card Account (cardld, accountNumber)

Customer Account (customerId, accountNumber)

ATM Info (*bankId*,ATMId,ATMLocation,ATMAddress)

Withdrawal Transaction(*bankId*,*ATMId*,date,time,transactionType,cardld,PIN, accountNumber, amount, balance)

Query Transaction (*bankId*,*ATMId*,date,time,transactionType,cardId,PIN, accountNumber, balance)

Transfer Transaction(*bankId*,*ATMId*,date,time,transactionType,cardld,PIN,

fromAccountNumber, toAccountNumber, amount)

PIN Validation Transaction (*bankId*,*ATMId*,date,time,transactionType,cardId,PIN, startDate, expirationDate)

8.详细设计；

“读卡器接口”任务会被一个外部读卡器事件唤醒，接着读取“ATM 卡”（ ATM Card）输入、将卡内容写入“ATM 卡”对象、向“ATM 控制（ ATM Control）发送一

个“卡片插入”（ cardInserted）消息，然后等待消息。如果“ATM 控制”发送回来的消息是“弹出卡”，那么就弹出 ATM 卡;如果发送回来的消息是“没收”，那么就没收卡。被动的数据抽象对象“ATM 卡”处于任务之外，用于保存卡的内容。

“银行系统”中的所有消息通信都是通过调用操作系统来完成的。因此，“读卡器接口”任务（生产者）和“ATM 控制”（消费者）之间的消息队列 ATMControlMessageQ 由操作系统提供，而“ATM 控制”和“读卡器接口”任务之间的同步通信也是如此。一个名为 cardReaderMessageBuffer 的消息缓存接收来自 ATM 控制”的同步消息。

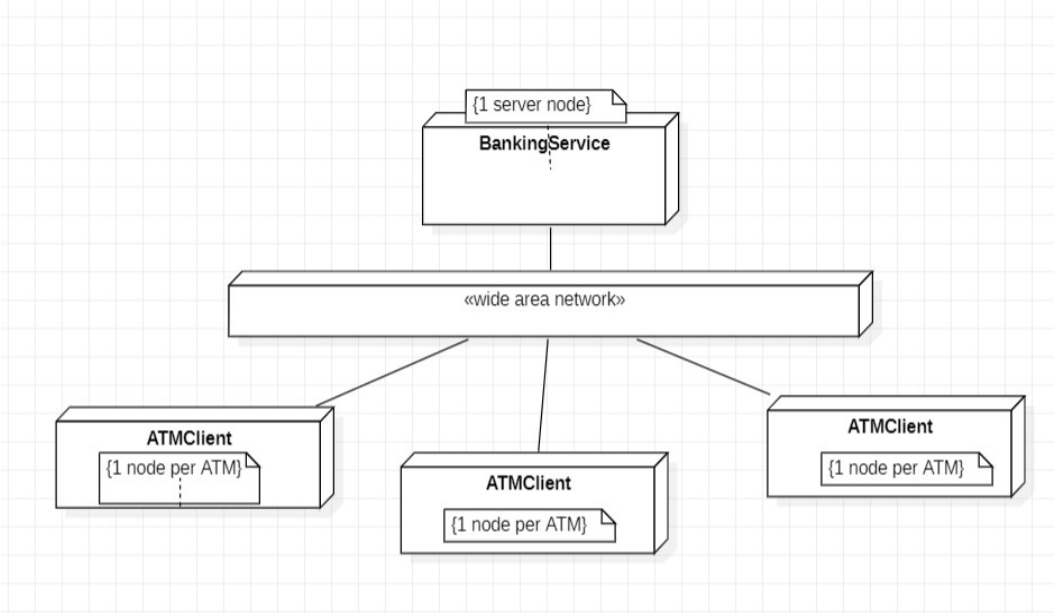
伪代码如下图所示：

```
Initialize card reader;
loop
--等待来自读卡器的外部中
wait (cardReaderEvent);
Read card data held on card's magnetic strip;
if card is recognized
then--将卡数据写到 ATM 卡对象中

    ATMCard.write (cardID , startDate, expirationDate);
--向 ATM 控制对象发送卡已插入的消息
send(ATMControlMessageQ, cardInserted);
```

```
--等待来自 ATM 控制的消息
receive (cardReaderMessageBuffer,message);
if message = eject
then
    Eject card;
    --向 ATM 控制对象发送卡已弹出的消息
    send(ATMControlMessageQ, cardEjected);
elseif message = confiscate
    then
    Confiscate card;
    --向 ATM 控制对象发送卡已没收的消息
    send (ATMControlMessageQ, cardConfiscated);
    else error condition;
    end if;
else --未能识别卡，因此弹出
    Eject card;
end if;
end loop;
```

9.部署方式设计。




由于这是一个客户端/服务器系统，所以客户端子系统有多个实例而服务子系统只有一个实例。如图 21-36 的部署图所示，每个子系统实例在自己的结点上运行。因此，每个.A.'.客户端实例在 ATM 结点上运行，“银行服务”实例在服务器结点上运行。

五、分析与讨论

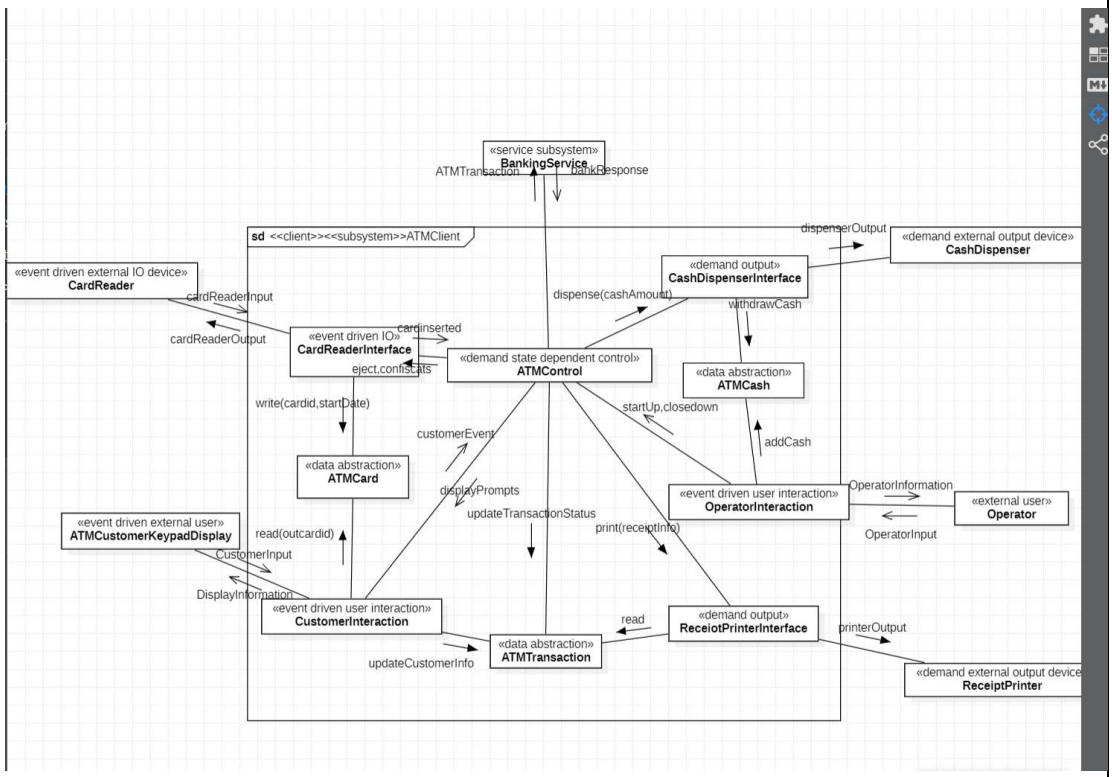
1．问题域静态建模的目的是什么？

问题域是包含现实世界与概念的领域，与系统解决的问题有关。作为 UML 模型中静态部分的基础，在构建领域模型时，首先要确定现实世界中的抽象，即系统会涉及主要概念对象。 这种想法是因为现实世界的变化不那么频繁。为什么从域模型而不是用例开始，是因为模型的动态部分（操作、方法）和静态（属性）部分结合在一起，这是用例驱动设计的必要条件。 同时，领域模型充当词汇表，用例编写者可以使用用它。同时，这一阶段的出发点是从系统的核心对象入手，

由内而外地确定这些对象将如何参与到要构建的系统中。此外，这个阶段的主要任务是让开发团队对正在开发的软件有一个全局的了解。

2．怎样在通信图上描述通信序列分支（从本次实验中举一例说明）？

如图所示，初始的“ATM 客户端子系统”并发通信图如下图所示：



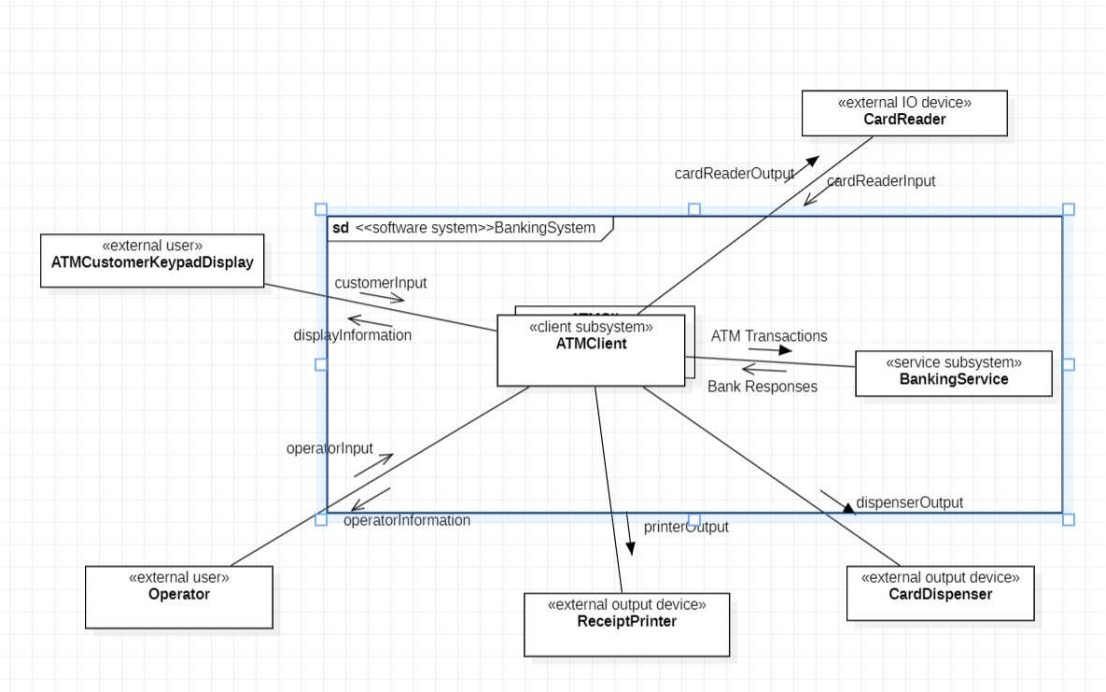

3．状态相关的控制类的状态图与通信图有何关系？

当通信图中的消息到达控制对象时，消息的事件部分可能会导致状态图中的状态转换。状态图中的操作可以表示状态转换的结果，该状态转换同时对应于通信图中的输出消息。为了使通信图和状态图保持一致，通信图中的相应消息和状态图

中的时间应具有相同的名称，并且消息编号规则应相同。

4．ATM 客户端与服务器通信模式是什么？给出相关的设计图。

如下图所示：




六、教师评语

签名：

日期：

成绩