UNIT- V

APPLICATION ANALYSIS

CONTENTS

Application Interaction Model

- Most domain models are static and operations are unimportant.
- After completing the domain model we then shift our attention to the details of an application and consider interaction
- Interaction modeling is used to determine the overall boundary of the system
- Identification of use cases and flesh them out with scenarios and sequence diagrams
- Prepare activity diagrams for use cases

Steps To Construct An Application Interaction Model (DFFF PAF POC)

Q. Elucidate in brief the steps involved in the application interaction modeling. (L-2,CO-3,PO-1,10m)

One can construct an application interaction model with the following steps.

- Determine the system boundary. [13.1.1]
- Find actors. [13.1.2]
- Find use cases. [13.1.3]
- Find initial and final events. [13.1.4]
- Prepare normal scenarios. [13.1.5]
- Add variation and exception scenarios. [13.1.6]
- Find external events. [13.1.7]
- Prepare activity diagrams for complex use cases. [13.1.8]
- Organize actors and use cases. [13.1.9]
- Check against the domain class model. [13.1.10]

1. Determine the system boundary (DFFF PAF POC)

- One should know the boundary of the system—in order to specify functionality.
- This means that you must decide what the system includes and, more importantly, what it doesn't include.
- ATM example: –

    Focus on ATM behavior

    ignore cashier details.

2. Find actors(DFFF PAF POC)

- Once you determine the system boundary, you must identify the external objects that interact directly with the system.
- These are its actors.
- Actors include humans xternal devices, and other software systems.
- Example for ATM application

  actors are

  1. Customer
  2. Bank
  3. Consortium.

Finding Use Cases(DFFF PAF POC)

- For each actor, list the fundamentally different ways in which the actor uses the system. Each of these ways is a use case
- Each use case should represent a kind of service that the system provides—something that provides value to the actor.

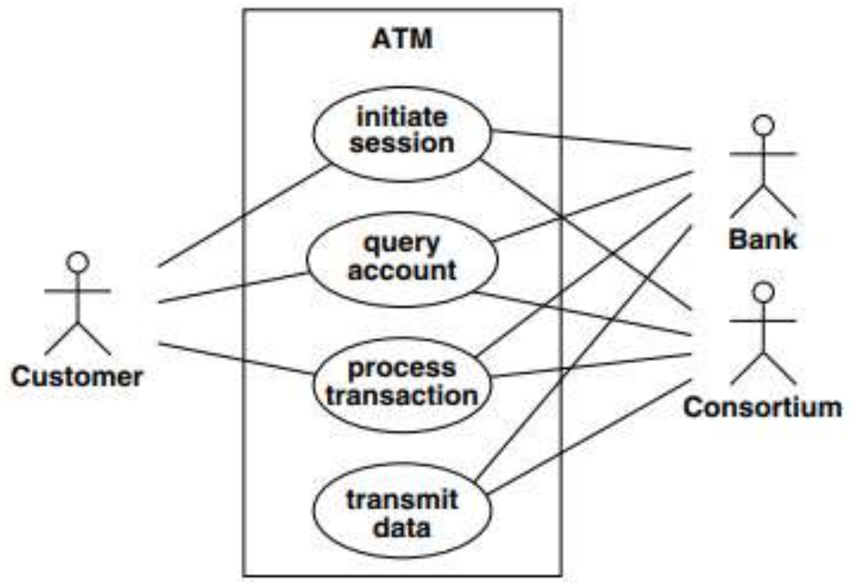


Figure 13.1 Use case diagram for the ATM. Use cases partition the functionality of a system into a small number of discrete units that cover its behavior.

Finding Use Cases

From the previous diagram

1. **Initiate session**. The ATM establishes the identity of the user and makes available a list of accounts and actions.
2. **Query account.** The system provides general data for an account, such as the current balance, date of last transaction, and date of mailing for last statement.
3. **Process transaction.** The ATM system performs an action that affects an account's balance, such as deposit, withdraw, and transfer. The ATM ensures that all completed transactions are ultimately written to the bank's database.
4. **Transmit data**. The ATM uses the consortium's facilities to communicate with the appropriate bank computers.

Find initial and final events(DFFF PAF POC)

- Finding the initial and final events for each use case
- To understand the behavior clearly of system
- Initial events may be

  a. A request for the service that the use case provides

  b. An occurrence that triggers a chain of activity
- You can start by finding the events that initiate each use case. Determine which actor initiates the use case and define the event that it sends to the system.
- You should also determine the final event ,final events indicate completion

For ATM example. Here are initial and final events for each use case.

- **Initiate session**
  - **Initial event**
    - The customer's insertion of a cash card.
  - --**final event**
    - The system keeps the cash card, or
    - The system returns the cash card.

Query account

- **Initial event**
  - A customer's request for account data.
- **final event**
  - The system's delivery of account data to the customer.

Process transaction

- **Initial event**
  - The customer's initiation of a transaction.
- **final event**
  - Committing or
  - Aborting the transaction

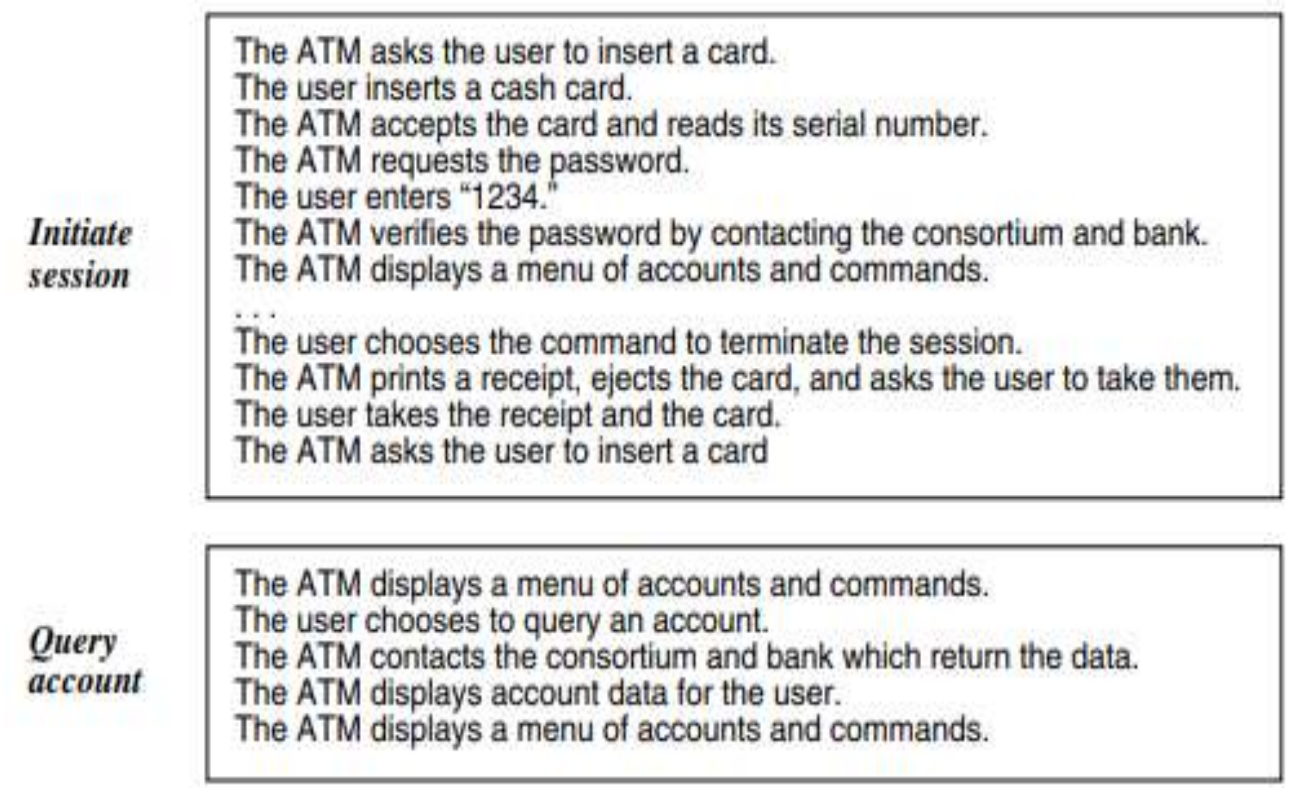
Transmit data

- **Initial event**
  - Triggered by a customer's request for account data, or
  - Recovery from a network, power, or another kind of failure.
- **final event**
  - Successful transmission of data.

Preparing Normal Scenarios(DFFF PAF POC)

- For each use case, prepare one or more typical dialogs.
- A scenario is a sequence of events among a set of interacting objects.
- Let us see some examples.

Normal ATM scenarios



Initiate session

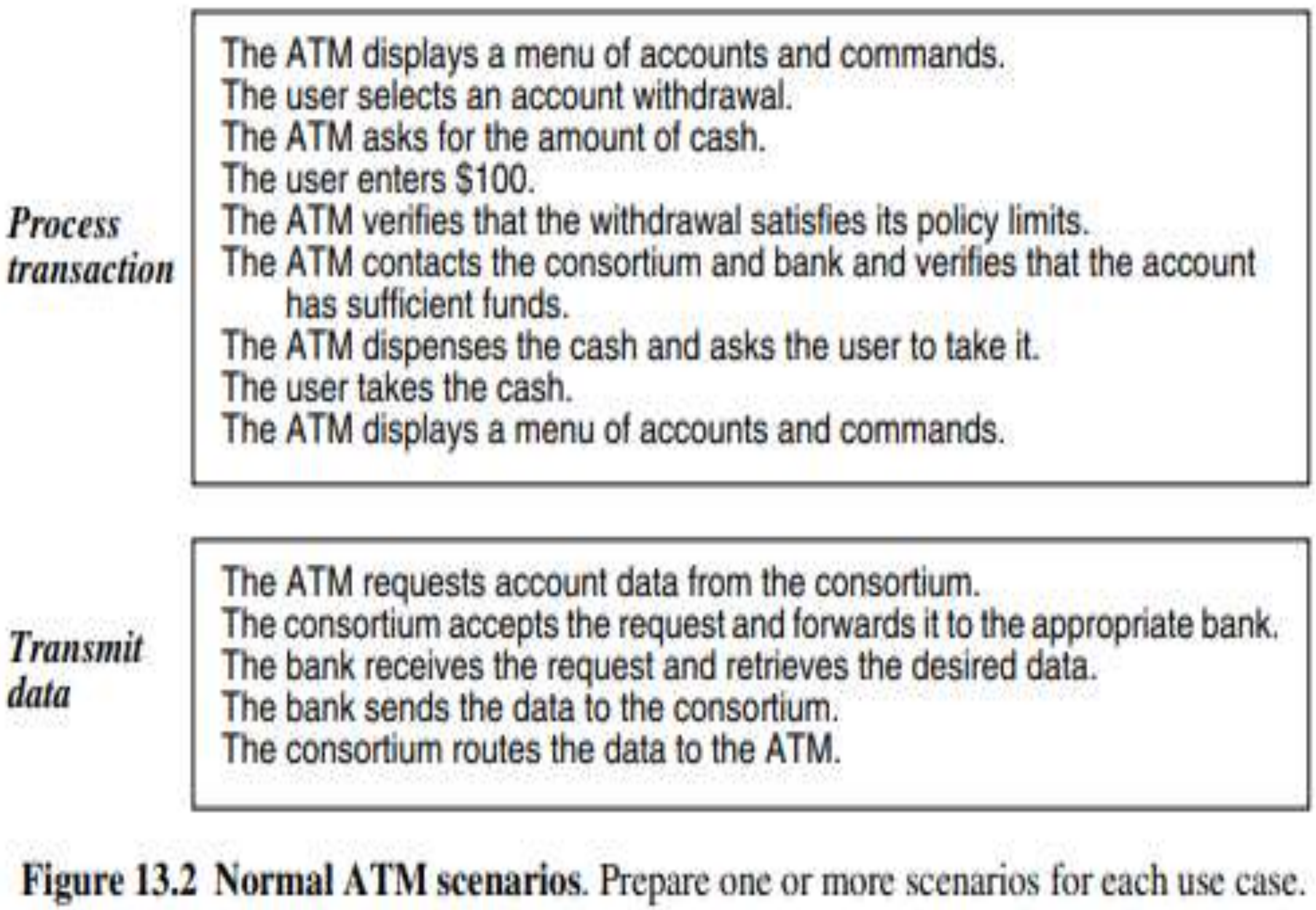
The ATM asks the user to insert a card.
The user inserts a cash card.
The ATM accepts the card and reads its serial number.
The ATM requests the password.
The user enters "1234."
The ATM verifies the password by contacting the consortium and bank.
The ATM displays a menu of accounts and commands.
. . .
The user chooses the command to terminate the session.
The ATM prints a receipt, ejects the card, and asks the user to take them.
The user takes the receipt and the card.
The ATM asks the user to insert a card

Query account

The ATM displays a menu of accounts and commands.
The user chooses to query an account.
The ATM contacts the consortium and bank which return the data.
The ATM displays account data for the user.
The ATM displays a menu of accounts and commands.

Figure 13.2 Normal ATM scenarios. Prepare one or more scenarios for each use case.

Preparing Normal Scenarios



Process transaction

The ATM displays a menu of accounts and commands.
The user selects an account withdrawal.
The ATM asks for the amount of cash.
The user enters $100.
The ATM verifies that the withdrawal satisfies its policy limits.
The ATM contacts the consortium and bank and verifies that the account has sufficient funds.
The ATM dispenses the cash and asks the user to take it.
The user takes the cash.
The ATM displays a menu of accounts and commands.

Transmit data

The ATM requests account data from the consortium.
The consortium accepts the request and forwards it to the appropriate bank.
The bank receives the request and retrieves the desired data.
The bank sends the data to the consortium.
The consortium routes the data to the ATM.

Figure 13.2 Normal ATM scenarios. Prepare one or more scenarios for each use case.

Adding Variation and Exception Scenarios (DFFF PAF POC)

- After you have prepared typical scenarios, consider "special cases and error cases". (SPECIAL ERROR OTHERS)

1. **Special cases** such as Omitted input E.g., maximum values, minimum value
2. **Error cases** E.g. Invalid values, failures to respond
3. **Other cases** E.g. Help requests, status queries

For ATM example. Some variations and exceptions follow.

- The ATM can't read the card.
- The card has expired.
- The ATM times out waiting for a response.
- The amount is invalid.
- The machine is out of cash or paper.
- The communication lines are down.
- The transaction is rejected because of suspicious patterns of card usage.

Finding External Events(DFFF PAF POC)

- The external events include
  - All inputs,
  - decisions,
  - interrupts, and
  - Interactions to or from users or external devices.
- An event can trigger effects for a target object. (Defination of the events)
- Use scenarios for normal events.
- Internal computation (not events) steps are not events, except for computations that interact with the external world.
- A transmittal of information to an object is an event.
- **For example**, enter password is a message sent from external agent(from keyboard) User to application object ATM. Some information flows are implicit. Many events have parameters

This comes in the external events

Sequence diagram

- Prepare a sequence diagram for each scenario.
- The sequence diagram captures the dialog and interplay between actors.

The sequence diagram clearly shows the sender and receiver of each event

ATM Example: Sequence diagram of the process transaction

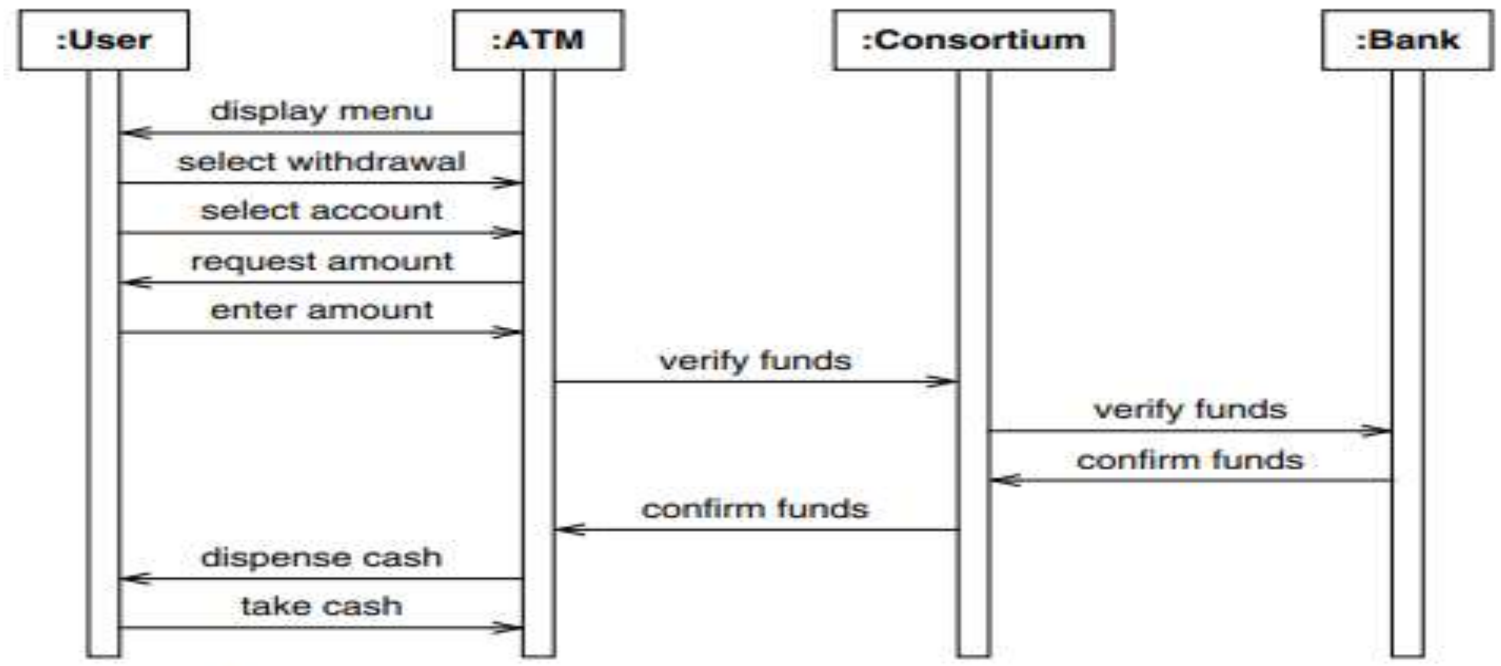


Figure 13.3 Sequence diagram for the *process transaction* scenario. A sequence diagram clearly shows the sender and receiver of each event.

- Events for the ATM case study (İmportant Diagrams)

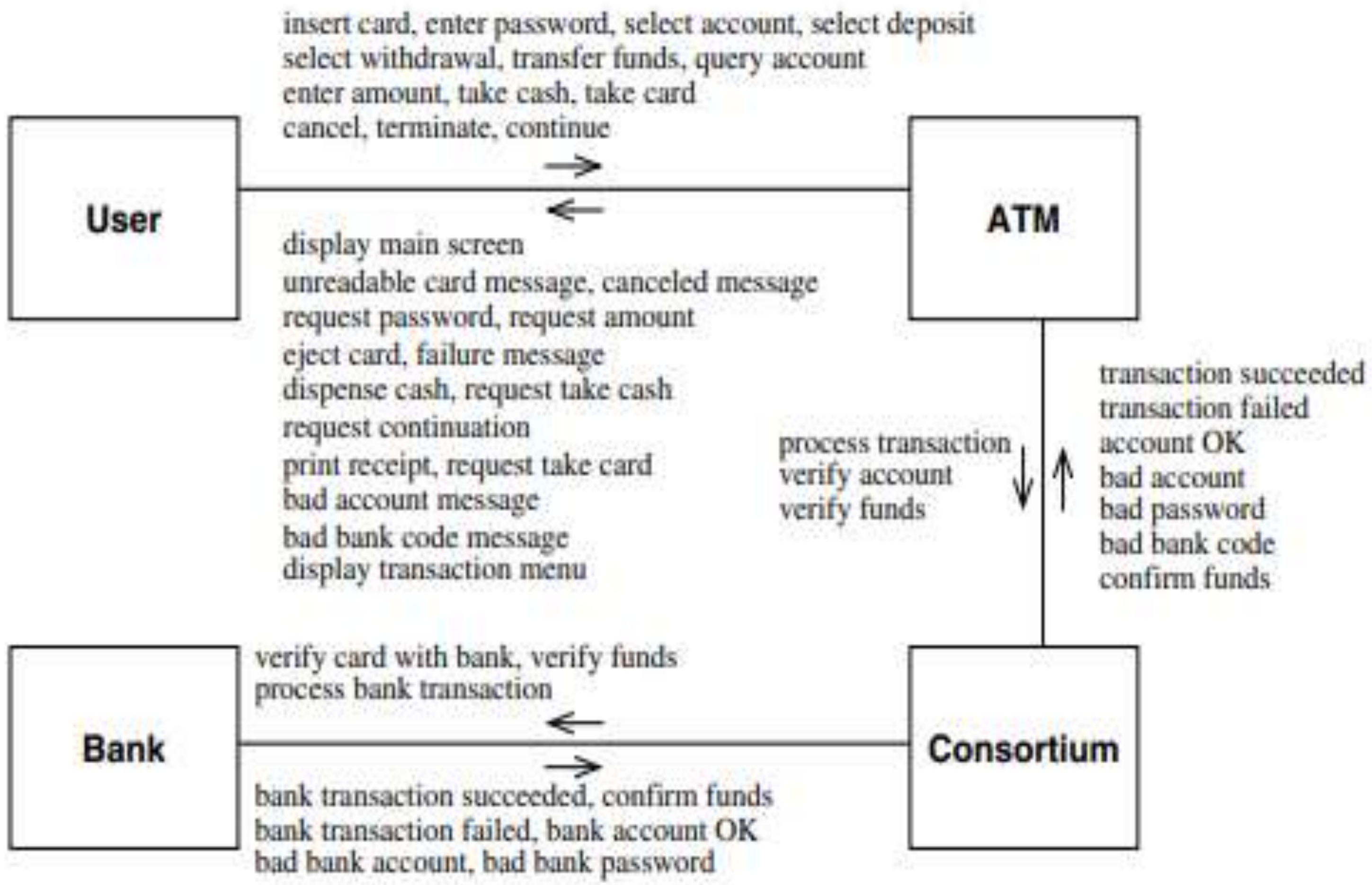

insert card, enter password, select account, select deposit
select withdrawal, transfer funds, query account
enter amount, take cash, take card
cancel, terminate, continue
User
ATM
display main screen
unreadable card message, canceled message
request password, request amount
eject card, failure message
dispense cash, request take cash
request continuation
print receipt, request take card
bad account message
bad bank code message
display transaction menu
process transaction
verify account
verify funds
transaction succeeded
transaction failed
account OK
bad account
bad password
bad bank code
confirm funds
verify card with bank, verify funds
process bank transaction
Bank
Consortium
bank transaction succeeded, confirm funds
bank transaction failed, bank account OK
bad bank account, bad bank password

Figure 13.4 Events for the ATM case study. Tally the events in the scenarios and note the classes that send and receive each event.

Preparing Activity Diagrams for Complex Use Cases (dfff paf Poc)

- **Sequence diagrams capture the dialog and interplay between actors, but they do not clearly show alternatives and decisions.**
- Example : When the user inserts a card, there are many possible responses. Some responses indicate a possible problem with the card or account; hence the ATM retains the card. Only the successful completion of the tests allows ATM processing to proceed.

Comes under preparing the activity diagrams for complex usecases.

Activity diagram for card verification.

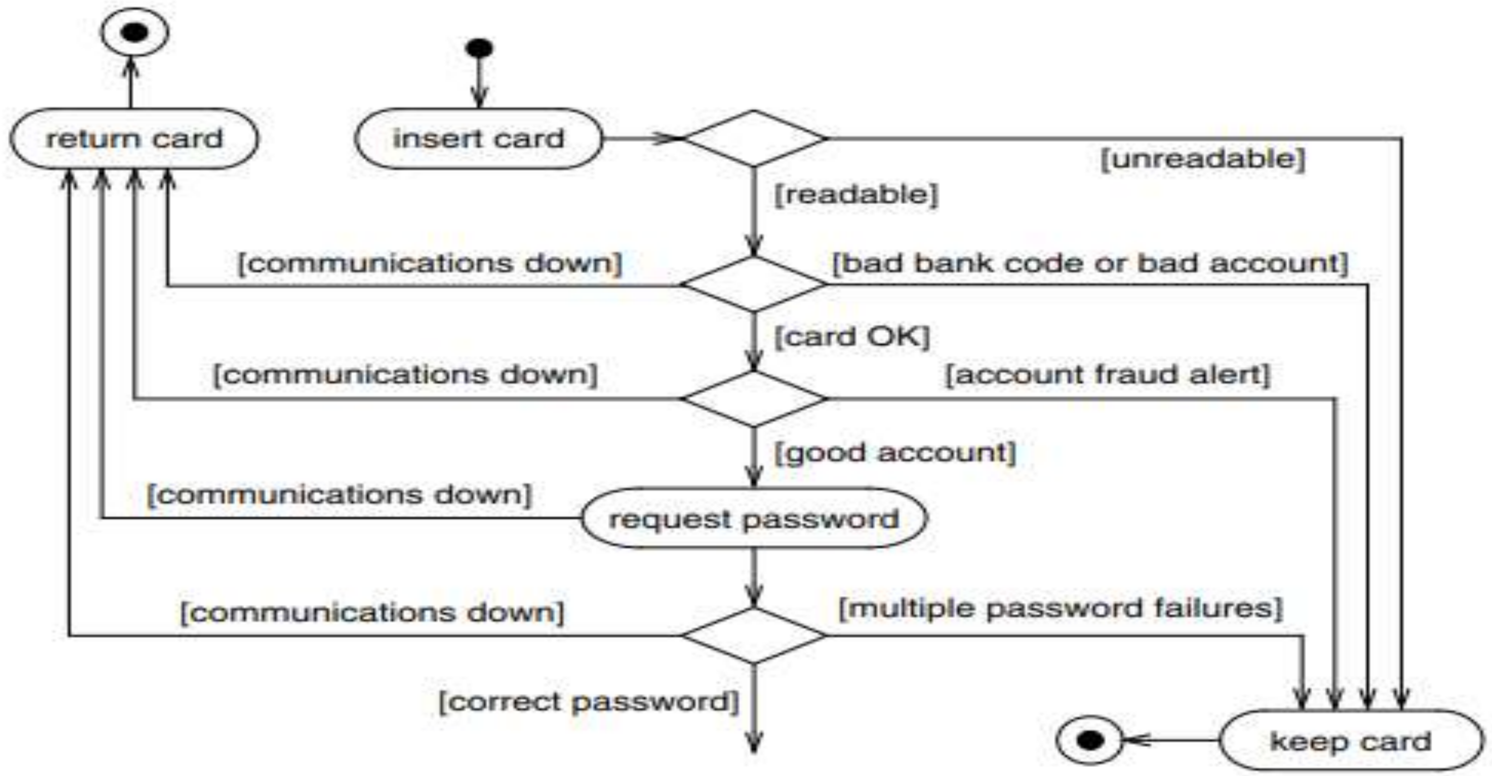


Figure 13.5 Activity diagram for card verification. You can use activity diagrams to document business logic, but do not use them as an excuse to begin premature implementation.

Organizing Actors and Use Cases(DFFF PAF POC)

- The next step is to organize use cases with relationships (include, extend, and generalization) It is helpful for large and complex systems
- ATM example. Figure 13.6 organizes the use cases with the include relationship.(Write any example for the complex use cases)

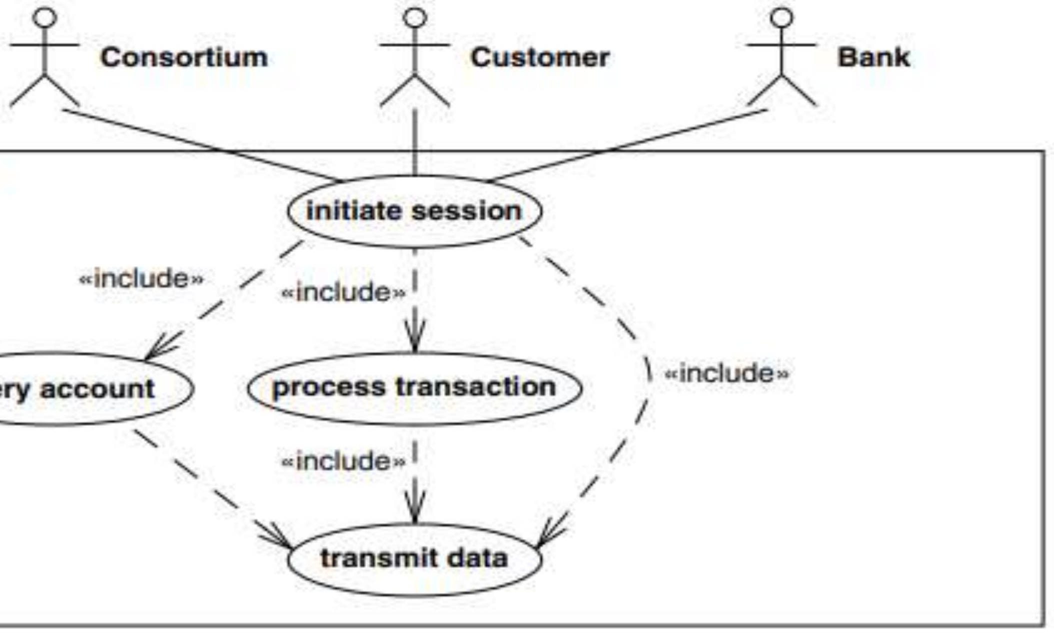
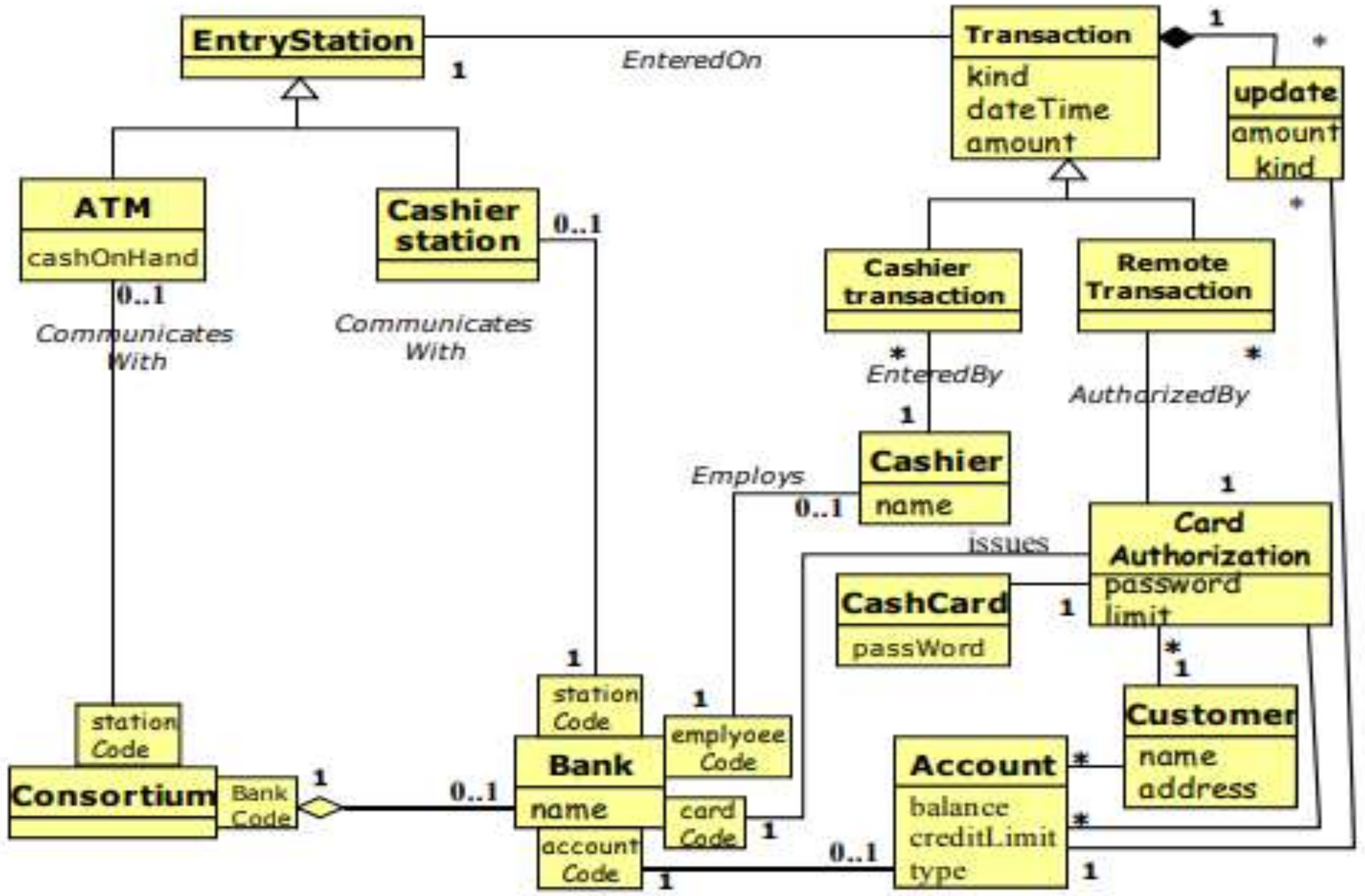


Figure 13.6 Organizing use cases. Once the basic use cases are identified, you can organize them with relationships.

Checking Against the Domain Class Model(dfff paf poC)

- The application(dynamic) and domain models(static in nature) should be mostly consistent.
- The actors, use cases, and scenarios are all based on classes and concepts from the domain model.
- Cross check the application and domain models to ensure that there are no inconsistencies.
- Examine the scenarios and make sure that the domain model has all the necessary data.
- Also make sure that the domain model covers all event parameters.

Domain class model for ATM Example.



Application Class Model

- Application classes define the application itself, rather than the real-world objects that the application acts on
- 4 steps → mneumonics (SDDC) Sach DaD Chudi
- Most application classes are computer-oriented and define the way that users perceive the applications
- Application Class Model – steps
  1. Specify user interfaces
  2. Define boundary classes
  3. Determine controllers
  4. Check against the interaction model

1. Specify user interfaces

User interface

a. Is an object or group of objects

b. Provide user a way to access system's

i. domain objects,

ii. commands, and

iii. Application options.

Try to determine the commands that the user can perform.

A command is a large-scale request for a service,

c. E.g.

i. Make a flight reservation

ii. Find matches for a phrase in a database Decoupling application logic from the user interface.

ATM example - The details are not important at this point.

- The important thing is the information exchanged.

Comes under the application class models (Sach DaD Chudi)

ATM example. Figure 13.7 shows a possible ATM layout. Its exact details are not important at this point. The important thing is the information exchanged.

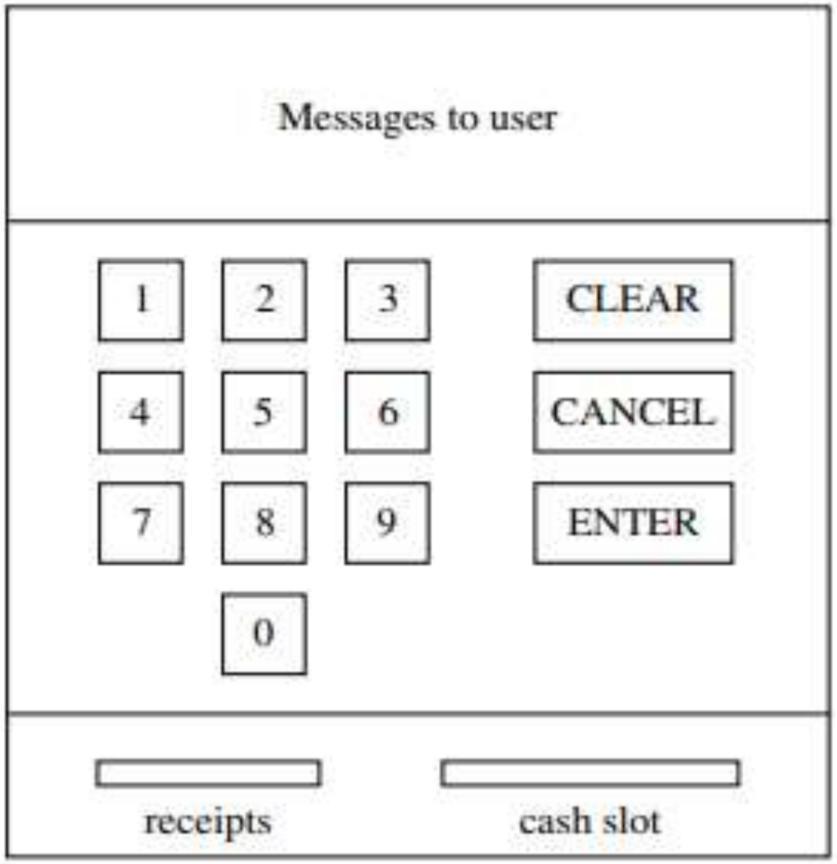


Figure 13.7 Format of ATM interface. Sometimes a sample interface can help you visualize the operation of an application.

2. Defining Boundary Classes

- A boundary class
  - Is an area for communications between a system and external source.
  - Converts information for transmission to and from the internal system.
- ATM example
  - CashCard Boundary
  - Account Boundary
    - Between the ATM and the consortium

3. **Determining Controllers**

- Controller is an active object that manages control within an application.
- **Controller**

1. Receives signals from the outside world or
2. Receives signals from objects within the system,
3. Reacts to them,
4. Invokes operation on the objects in the system, and
5. Sends signals to the outside world.

ATM Example

- There are two controllers for ATM ex.

– The outer loop verifies customers and accounts.

– The inner loop services transactions.

4.Checking Against the Interaction Model

- Go over the use cases and think about how they would work.
- When the domain and application class models are in place, you should be able to simulate a use case with the classes. (very important point in the pocket bro !!!)
- ATM Example

ATM example. Figure 13.8 shows a preliminary application class model and the domain classes with which it interacts. There are two interfaces—one for users and the other for communicating with the consortium. The application model just has stubs for these classes, because it is not clear how to elaborate them at this time.

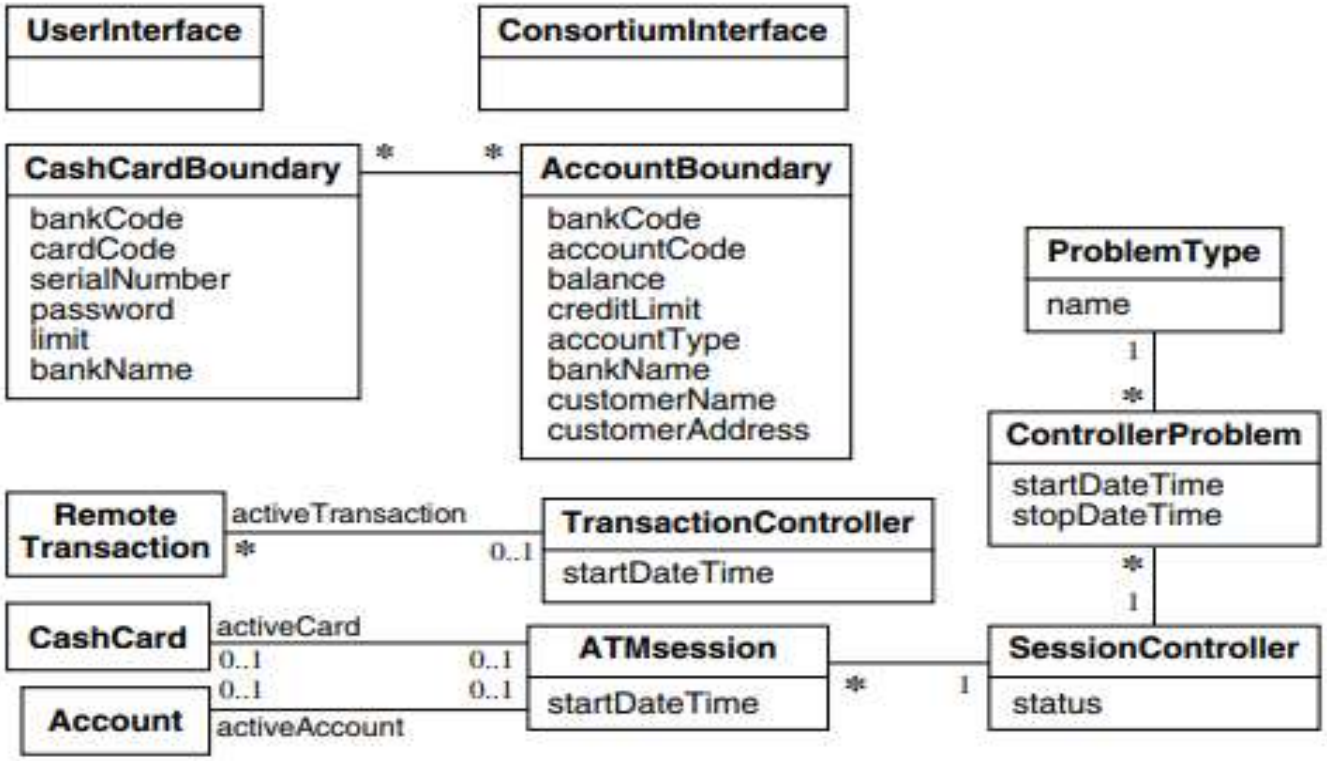


Figure 13.8 ATM application class model. Application classes augment the domain classes and are necessary for development.

Overview of Class Design

- Review of Analysis: Domain analysis and Application analysis
- The analysis model describes the information that the system must contain at high level operations that it must perform.
- The purpose of class design is to complete the definition of classes and association and choose algorithms and operation.
- The simplest and best approach is to carry out analysis classes directly to design

Overview of Class Design

- During design, you choose among different ways to realize the analysis classes with an eye toward minimizing execution time, memory, and other cost measures.
- You can introduce new classes to store intermediate results during program execution and avoid recomputation.
- OO design is an iterative process.

Overview of Class Design

- Class design involves the following steps.

■ Bridge the gap from high-level requirements to low-level services.

■ Realize use cases with operations.

■ Formulate an algorithm for each operation.

■ Recurse downward to design operations that support higher-level operations.

■ Refactor the model for a cleaner design.

■ Optimize access paths to data.

■ Reify behavior that must be manipulated.

■ Adjust class structure to increase inheritance.

■ Organize classes and associations.

Thank You