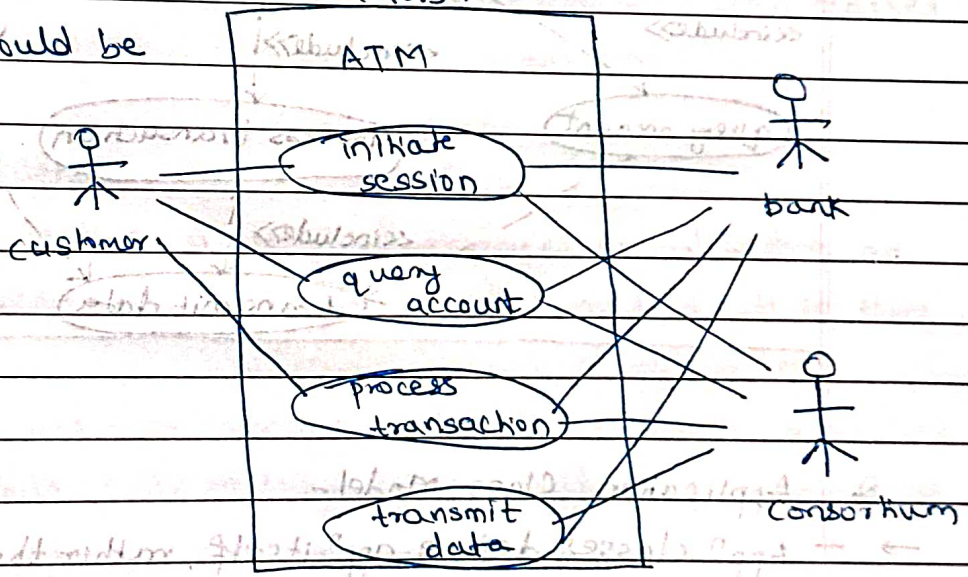
OO MD Unit 4

Q. App^n Interaction Model -

→ - we apply it after completing domain model to shift our attention to details of app^n and begin interaction.
- we begin interaction by determining the overall boundary of system. Then identify use cases and flesh them out with scenarios and sequence diagrams.
- we prepare activity diagrams for use cases that are complex once we fully understand use cases, we organize them with relationships. And finally check with domain class model to ensure that there are no inconsistencies.
- Example of ATM -
- focus on ATM behavior & ignore cashier details.
- Customer, Bank, Consortium are actors.
- Some use cases would be



- Now, find initial & final events. Some may be

initial - customer's insertion of card.
final - system returns the card.

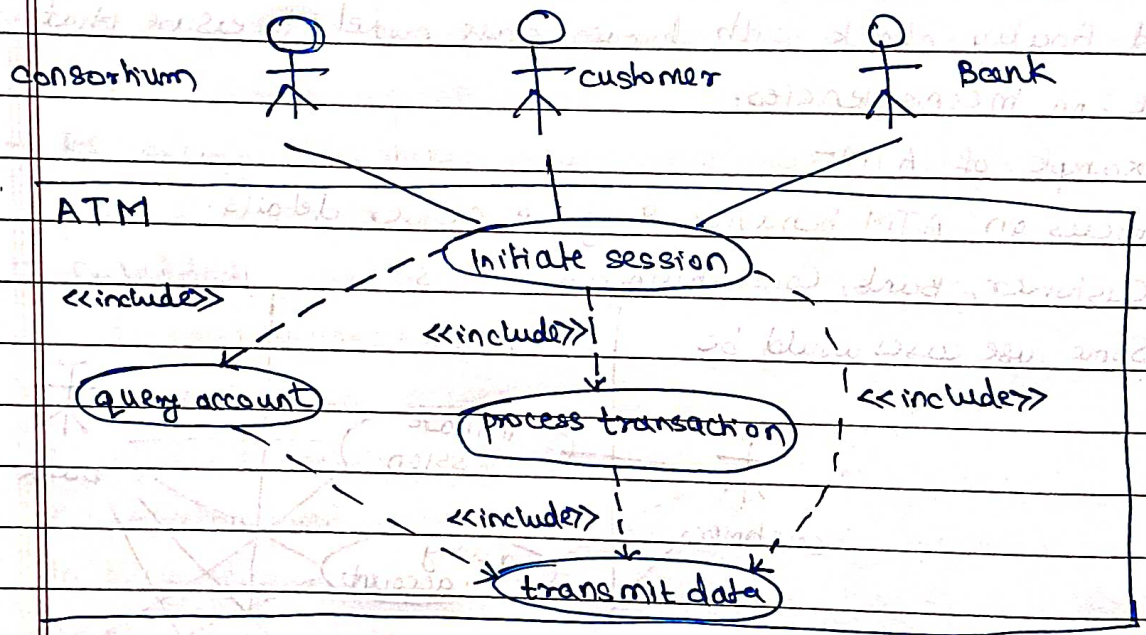
Query -
initial - customer's request for account data
final - system's delivery of account data to customer.

- Then prepare normal scenarios. A scenario is a sequence of events among set of interacting objects.

Normal scenario of Query session →

1. ATM displays a menu of commands & accounts.
2. User chooses to query an account
3. ATM contacts consortium & bank which return data.
4. ATM displays account data
5. Again, ATM displays menu of commands & accounts.

- Finally organize use cases with relationships and actors with generalizations.



Q. Application Class Model-

→ - App^n classes define app^n itself, rather than real-world objects that app^n acts on.

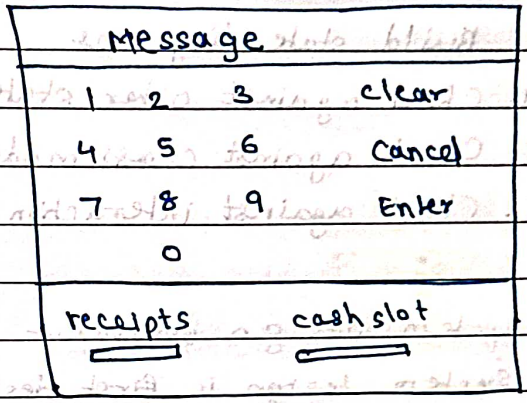
- we construct app^n class model with following steps-

1. Specify user interfaces
2. Define boundary classes
3. Determine controllers.
4. Check against interaction model.

Page :
Date : / /

1. Specify user interface -
- A user interface is group of objects that provides user with a way to access it's domain objects, commands, etc. During analysis, we emphasize on information flow & control.

eg. Possible ATM Layout



2. Define boundary classes -
- Boundry class is an area for communications betⁿ system and external source.
- In ATM example, it would be helpful to define boundary classes to encapsulate communication betⁿ ATM & consortium.

3. Determining controllers -
- A controller is an active object that manages control within an application. It receives signals from outside world, reacts to them, invokes operations
- In ATM example, one loop verifies customers & accounts, other loop services transactions. Each of these loops could most naturally be handled with controller.

4. Check against interaction model -
- As we build appⁿ class model, we go over use cases and think about how they would work

Q. Application state model -
→ Appⁿ state model focuses on appⁿ classes and augments the domain state model. Appⁿ classes are more likely to have imp.

temporal behavior than domain classes.
- We construct appn state model with following steps-
1. Determine appn classes with states.
2. Find events.
3. Build state diagrams.
4. Check against other state diagrams.
5. Check against class model.
6. Check against interaction model.

Q. System design overview-
→ - System design is first design stage for devising basic approach to solving problem. During system design, developers decide overall structure and style.
- Following are steps involved in it-
1. Estimate system performance.
2. Make reuse plan.
3. Organize system into subsystems.
4. Identify concurrency inherent in problem.
5. Manage data stores.
6. Choose software control strategy.
7. Handle boundary conditions
8. Select architectural style.

Q. Estimating performance-
→ - Early in planning for new system, we should prepare a rough performance estimate.
- In this step, we determine if system is feasible.
- It is done in order to make simplifying assumptions.
- ATM example- Suppose we are planning an ATM network for bank. So, bank has 40 branches, suppose there are equal no. of terminals in supermarkets. On a busy day, half the terminals are busy at once. Suppose each customer takes a minute to perform

session & that most transactions involve a single deposit or withdrawal. So, we estimate a peak requirement of about 40 transactions a minute, or about one per second.

Q. Making reuse plan-

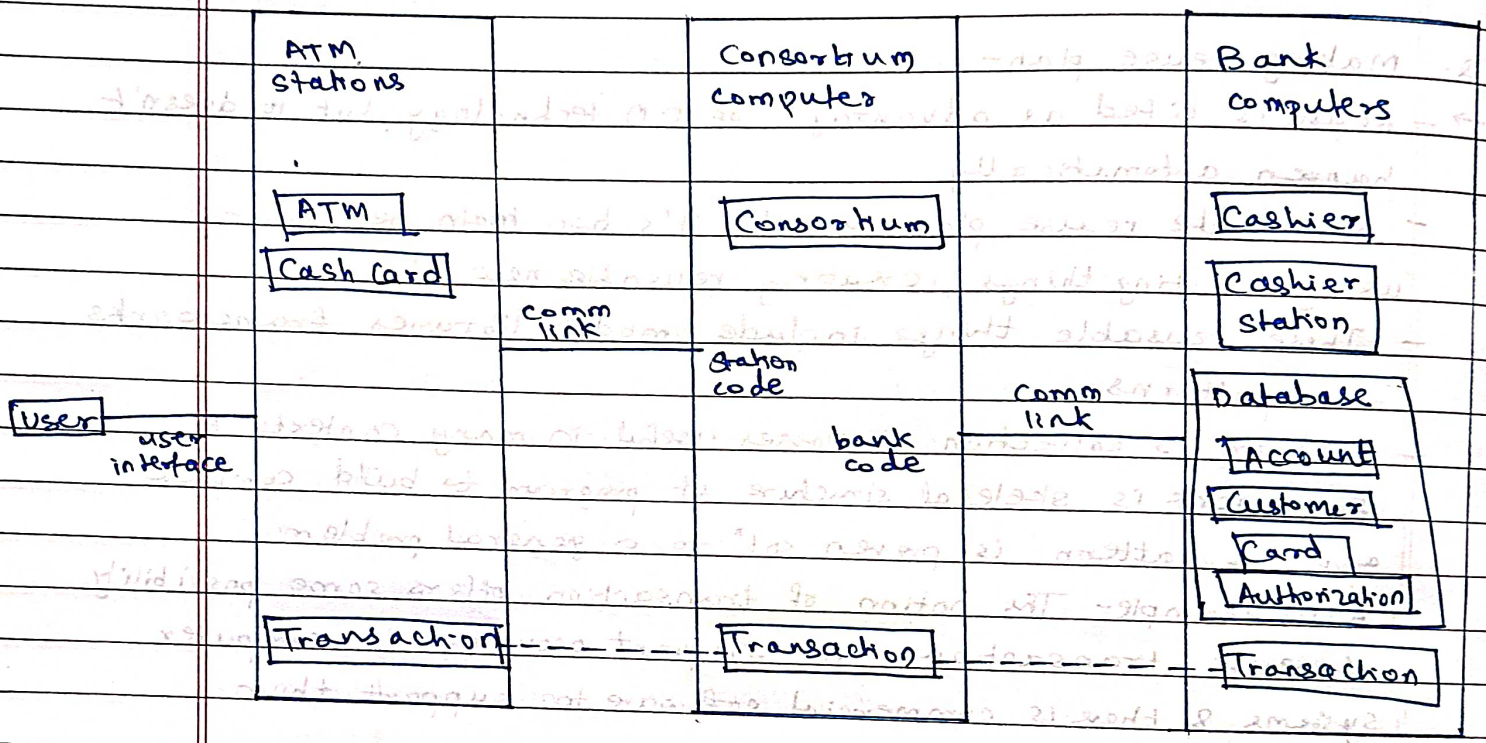
→ - Reuse is cited as advantage of OO technology, but it doesn't happen automatically.
- We make reuse plan due to it's two main benefits- using existing things, creating reusable new things.
- These reusable things include models, libraries, frameworks and patterns.
- A library is collection of classes useful in many contexts. A framework is skeletal structure of program to build complete appⁿ. A pattern is proven solⁿ to a general problem.
- ATM example- The notion of transaction offers some possibility of reuse - transactions are frequent occurence in computer systems & there is commercial software to support them.

Q. Breaking system into subsystem-

→ - A subsystem is group of classes, associations, operations, events & constraints. It is usually identified by service it provides.
- Each subsystem has a well-defined interface to rest of system. The relation between two subsystems can be Client-server relationship, peer-to-peer relationship.
- In a client-server relationship, the client calls on server, which performs some service and replies with a result. The client must know the server's interface.
- In a peer-to-peer relationship, each sub-system may call on others. A communication from one subsystem to another is not necessary followed by immediate response. Peer-to-peer interactions are more complicated.
- Decomposition of systems into subsystems can organized as a

sequence of Horizontal layers, vertical partitions or combination of layers & partitions.

- ATM example -



In above ATM architecture, there are three major subsystems - ATM stations, consortium computer and bank computer. The architecture uses station code and bank code to distinguish phone lines to consortium computer.

Q. Identifying concurrency -

→ - In this step we identify objects that must be active concurrently, objects that have mutually exclusive activity.
- The state model is guide to identify concurrency. Two objects are inherently concurrent if they can receive events at same time without interacting.
- In ATM machine example, if ATM contained requirement that each machine should continue to operate locally in event of central system failure, then we would need to include inherent CPU in each machine.

Page :
Date : / /

- By examining the state diagrams, we can fold many objects onto a single thread of control. A thread of control is path through a set of state diagrams on which only a single object at time is active.
- eg. combine ATM object with bank transaction object as a single task.

Q. Management of data storage -

→ - There are several alternatives for data storage that we can use separately or in combination like data structures, files and databases. Different kinds of data stores provide trade-offs among cost, access time, capacity & reliability.
- Files are cheap, simple and permanent. The data is with high volume and low information density. In it data is accessed sequentially & it can be fully read into memory.
- Database makes app^n easier to port, but interface is complex. It is suitable for data that require updates at fine levels, data requires co-ordinated updates via transactions, data must be secured against unauthorized access, etc.
- Typical bank computer would use a relational DBMS as they are fast, readily available and cost-effective.

Q. Handling global resources -

→ - The system designer must identify global resources and determine mechanisms for controlling access to them.
- There are several kinds of global resources like -

1. Physical units - eg. processors, tape drivers, communication satellites.
2. Space - eg. disk space, workstation screen, etc.
3. Logical names - eg. object IDs, filenames, class names.
4. Access to shared data - eg. databases.

- If resource is physical object, then it can control itself by establishing a protocol for obtaining access.

- If resource is logical entity such as object ID, then there is danger of conflicting access in shared environment.
- We can avoid conflict by having 'guardian object' own each global resource and control access to it.
- In ATM example, Bank codes and account numbers are global resources. Bank codes must be unique within context of consortium, account codes must be unique within context of bank.

Q. Choosing software control strategy.

→ - Although all subsystems need not use same implementation, it is best to choose a single control style for whole system. There are two kinds of control flows in software system - external, internal.

A. External Control-

- External control concerns the flow of externally visible events and objects in system. There are three kinds of control for external events-

1. Procedure driven control- In it, control resides within program code. Procedures request external input and wait; it resumes within procedure that made the call.
2. Event driven control- In it, ~~event~~ control resides within dispatcher or monitor that language, subsystem or operating system provides.
3. Concurrent control- In it, control resides concurrently in several independent objects, each a separate task.

B. Internal Control-

Internal control refers to flow of control within process. It exists only in implementation and therefore is neither inherently concurrent nor sequential.

- In ATM example, event-driven control is appropriate paradigm. ATM services a single user, so there is little need for concurrent control.

Page :
Date : / /

Handling Boundary Conditions-

→ - Most of system design is concerned with steady-state behavior, but boundary conditions are also important
- Boundary conditions are as follows -

1. Initialization-
- The system must initialize constant data, parameters, global variables, tasks, etc.
- During initialisation, only subset of functionality of system is usually available.

2. Termination-
- It is simpler than initialization, becoz many internal objects can simply be abadoned.
- The task must release any external resources that it had reserved.

3. Failure-
- It is unplanned termination of system. Good system designer plans for orderly failure.
- Failure can also arise from bugs in the system.

Q. Common architectural styles-

→ - Several prototypical architectural styles are common in existing systems. Each of it is well suited to certain kind of system. Some of them are-

1. Batch transformation-
- It is a data transformation executed once on an entire input set.
- It performs sequential computations, there is no ongoing interaction with outside world.

2. Continuous transformation-
- It is data transformation performed continuously as inputs change.
- A continuous transformation updates outputs frequently.

Page :
Date : / /

3. Interactive Interface-
- It is a system that is dominated by interactions between system and external agents like humans or devices.
- It usually includes only part of an entire application.

4. Dynamic Simulation-
- It models or tracks real-world objects. Examples include molecular motion modelling, economic models, etc.
- The objects & operations come directly from application.

5. Real time system-
- It is an interactive system with tight time constraints on actions.
- It's design is complex and involves issues such as interrupt handling, prioritization of tasks, etc.

6. Transaction Manager-
- It is system whose main function is to store and retrieve data.
- It deals with multiple users who read and write data at the same time.
- They are often built on top of database management system (DBMS).

Q. Architecture of ATM system-
→ See diagram in subsystems question.

14.13 Architecture of the ATM System

The ATM system is a hybrid of an interactive interface and a transaction management system. The entry stations are interactive interfaces—their purpose is to interact with a human to gather information needed to formulate a transaction. Specifying the entry stations consists of constructing a class model and a state model. The consortium and banks are primarily a distributed transaction management system. Their purpose is to maintain data and allow it to be updated over a distributed network under controlled conditions. Specifying the transaction management part of the system consists primarily of constructing a class model. Figure 14.2 shows the architecture of the ATM system.

The only permanent data stores are in the bank computers. A database ensures that data is consistent and available for concurrent access. The ATM system processes each transaction as a single batch operation, locking an account until the transaction is complete.

Concurrency arises because there are many ATM stations, each of which can be active at any time. There can be only one transaction per ATM station, but each transaction requires the assistance of the consortium computer and a bank computer. As Figure 14.2 shows, a transaction cuts across physical units; the diagram shows each transaction as three connected pieces. During design, each piece will become a separate implementation class. Although there is only one transaction per ATM station, there may be many concurrent transactions per consortium computer or bank computer. This does not pose any special problem, because the database synchronizes access to any one account.

The consortium computer and bank computers will be event driven. Each of them queues input events but processes them one at a time in the order received. The consortium computer has minimal functionality. It simply forwards a message from an ATM station to a bank computer and from a bank computer to an ATM station. The consortium computer must be large enough to handle the transaction load. It may be acceptable to block an occasional transaction, provided the user receives an appropriate message.

The bank computer is the only unit with any nontrivial procedures, but even those are mostly just database updates. The only complexity might come from failure handling. The bank computers must have capacity to handle the expected worst-case load, and they must have enough disk storage to record all transactions.

The system must contain operations for adding and deleting ATM stations and bank computers. Each physical unit must protect itself against the failure or disconnection from the rest of the network. A database protects against loss of data. However, special attention must be paid to failure during a transaction so that neither the user nor the bank loses money—this may require a complicated acknowledgment protocol before committing the transaction. The ATM station should display an appropriate message if the connection is down. The ATM must handle other kinds of failure as well, such as exhaustion of cash or paper for receipts.

On a financial system such as this, fail-safe transactions are the highest priority. If there is any doubt about the integrity of a transaction, then the ATM must abort the transaction with an appropriate message to the user.