HMR

Institute of Technology & Management

Laboratory Manual

SOFTWARE ENGINEERING

3rd CSE
Dept: Computer Science & Engineering

SUBMITED TO	**SUBMITED BY**
Ms. Isha Gupta	Harsh Rohilla
Associate Professor, CSE	00596502719
HMRITM	CSE 5th C

INDEX

EXCERCISE NO. 1

Aim: To prepare PROBLEM STATEMENT for any project.

REQUIREMENTS:

Hardware Interfaces

- Pentium(R) 4 CPU 2.26 GHz, 128 MB RAM
- Screen resolution of at least 800 x 600 required for proper and complete viewing of screens. Higher resolution would not be a problem.
- CD ROM Driver

Software Interfaces

- Any window-based operating system
- WordPad or Microsoft Word

THEORY:

The problem statement is the initial starting point for a project. It is basically a one to three page statement that everyone on the project agrees with that describes what will be done at a high level. The problem statement is intended for a broad audience and should be written in non-technical terms. It helps the non-technical and technical personnel communicate by providing a description of a problem. It doesn't describe the solution to the problem.

The input to requirement engineering is the problem statement prepared by customer. It may give an overview of the existing system along with broad expectations from the new system.

The first phase of requirements engineering begins with requirements elicitation i.e. gathering of information about requirements. Here, requirements are identified with the help of customer and existing system processes. So from here begins the preparation of problem statement. So, basically a problem statement describes **what** needs to be done without describing **how**.

Conclusion: The problem statement was written successfully by following the steps described above.

ATM MANAGEMENT SYSTEM

The software proposes to manage and enable smooth functioning of an ATM which would include the standard ATM transaction type such as withdrawls, balance, enquiry, pin change,

Mini statement, cash deposit and funds transfer. It may also includes some value-added services like bill payment or mobile top-ups etc.

The ATM system will provide the following services to the user:

- Better ATM transaction operations.
- Minimum balance maintenance for account.
- Routine check-up of ATM.
- If under theft circumstances, we can block the ATM card from further transaction by reversing the pin.
- Frequent fraudulent transactions and fund transfers to a particular account should be reported to bank manager.
- Rewards points also given for value added services as per the bank criteria.
- Easier interactive system for usage.
- Ability to send request for cheque book.

Conclusion: Problem Statement of ATM MANAGEMENT SYSTEM has been written successfully.

EXCERCISE NO. 2

Aim: Understanding an SRS.

Requirements:

Hardware Requirements:

- PC with 300 megahertz or higher processor clock speed recommended; 233 MHz minimum required.
- 128 megabytes (MB) of RAM or higher recommended (64 MB minimum supported)
- 1.5 gigabytes (GB) of available hard disk space
- CD ROM or DVD Drive
- Keyboard and Mouse (compatible pointing device).

Software Requirements:

- Rational Rose, Windows XP,

Theory:

An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies *at a particular point in time* (usually) prior to any actual design or development work. It's a two-way insurance policy that assures that both the client and the organization understand the other's requirements from that perspective at a given point in time.

The SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an eCommerce Web site, and so on) must provide, as well as states any required constraints by which the system must abide. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

It's important to note that an SRS contains functional and nonfunctional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

A well-designed, well-written SRS accomplishes four major goals:

- It provides feedback to the customer. An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.

- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

SRS should address the following

The basic issues that the SRS shall address are the following:

a) ***Functionality.*** What is the software supposed to do?

b) ***External interfaces.*** How does the software interact with people, the system's hardware, other hardware, and other software?

c) ***Performance.*** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) ***Attributes.*** What are the portability, correctness, maintainability, security, etc. considerations?

e) ***Design constraints imposed on an implementation.*** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.?

Characteristics of a good SRS

An SRS should be

a) Correct

b) Unambiguous

c) Complete

d) Consistent

e) Ranked for importance and/or stability
f) Verifiable
g) Modifiable
h) Traceable

Correct - This is like motherhood and apple pie. Of course you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things that are not correct.

Unambiguous - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasing the SRS can be a waste of time. But as you find ambiguities - fix them.

Complete - A simple judge of this is that is should be all that is needed by the software designers to create the software.

Consistent - The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

Ranked for Importance - Very often a new system has requirements that are really marketing wish lists. Some may not be achievable. It is useful provide this information in the SRS.

Verifiable - Don't put in requirements like - "It should provide the user a fast response." Another of my favorites is - "The system should never crash." Instead, provide a quantitative requirement like: "Every key stroke should provide a user response within 100 milliseconds."

Modifiable - Having the same requirement in more than one place may not be wrong - but tends to make the document not maintainable.

Traceable - Often, this is not important in a non-politicized environment. However, in most organizations, it is sometimes useful to connect the requirements in the SRS to a higher level document. Why do we need this requirement?

A sample of basic SRS Outline

1. Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms

and Abbreviations

1.4 Additional information

1.5 References

1.6 Overview

2. Overall Description

2.1 Product perspective

2.1.1 System Interface

2.1.2 User Interfaces

2.1.3 Hardware Interfaces

2.1.4 Software Interfaces

2.1.5 Communication Interfaces

2.1.6 Memory Constraints

2.1.7 Operations

2.1.8 Site Adaptation Requirements

2.2 Product functions

2.3 User characteristics

2.4 Constraints

2.5 Assumptions and dependencies

3. External Interface Requirements

3.1 User interfaces

3.2 Hardware interfaces

3.3 Software interfaces

3.4 Communication protocols and interfaces

4. System Features

4.1 System feature A

4.1.1 Description and priority

4.1.2 Action/result

4.1.3 Functional requirements

4.2 System feature B

5. Other Nonfunctional Requirements

5.1 Performance requirements

5.2 Safety requirements

5.3 Security requirements

5.4 Software quality attributes

5.5 Project documentation

5.6 User documentation

6. Other Requirements

Appendix A: Terminology/Glossary/Definitions list Appendix B:
To be determined

Conclusion: The SRS was made successfully by following the steps described above.

SAMPLE SRS OF ATM Management System

1. Introduction

The ATM management software is to be developed for Automated Teller Machines (ATM). An automated teller machine (ATM) is computerized telecommunications device that provides a financial institution's customers a secure method of performing financial transactions, in a public space without the need for a human bank teller. Through ATM, customers interact with a user-friendly interface that enables them to access their bank accounts and perform various transactions.

1.1 Purpose

The ATM management system (AMS) maintains the records of account holder transactions by interacting with the bank servers to keep them up to date. The ATM management system provides the user to perform bank related operations without going in the bank.

1.2 Scope

The proposed AMS must be able to perform the following functions:

1. Provide login facility to account holder in bank using ATM card and pin.
2. Provide cash dispensing facility to account holder on successful transaction.
3. Block card in case of continuous three wrong pin attempts.
4. Block card in case of reverse pin input for enhanced safety.
5. Provide facility to customers to check their account balance.
6. Provide facility to customers to change their own ATM card pin.
7. Provide facility to customers to generate a mini account statement of last 10 transactions.
8. Provide facility to customers to transfer funds from their account to the beneficiary account.
9. Provide facility to customers to request cheque book from bank.
10. Provide following additional Value-Added Services to the customers:
    - Check Reward Points
    - Recharge prepaid mobile number
    - Pay bill such as electricity, water etc.
11. Provide login facility to the maintainer.

12. Provide following services to the maintainer:

- Start the ATM services
- Stop the ATM services
- Update the ATM software

1.3 Definitions, Acronyms and Abbreviations

AMS: ATM Management System

ATM: Automated Teller Machine - An unattended electronic machine in a public place, connected to a data system and related equipment and activated by a bank customer to obtain cash withdrawals and other banking services.

RAM: Random Access Memory

ATM Card: Card encoded with details of customer's account details. Customer/ Account **Holder:** Any person holding an account in a bank.

Bank: Financial Institution that provides various types of currency related facilities such as storage of currency, dispensing currency etc.

Maintainer: Any person that is designated to resolve the problems/issues related to smooth functioning of ATM.

Bank Server: A machine that holds details about the accounts associated with the bank.

Card Pin: A 4-digit combination that is related to ATM card needed to be entered in ATM to access ATM services.

1.4 Additional information

None

1.5 References

- Object-Oriented Software Engineering by Yogesh Singh & Ruchika Malhotra, PHI Learning Pvt. Ltd., 2012
- IEEE Recommended Practice for Software Requirements Specifications – IEEE Std. 830-1998.

- IEEE Standard for Software Test Documentation – IEEE Std. 829-1998.

1.6 Overview

The rest of the SRS document describes various system requirements, interfaces, features and functionalities.

2. Overall Description

AMS facilitates the customer to perform various transactions in his/her account without going to bank. This software offers benefits such cash withdrawals, balance transfers, deposits, inquiries and other banking related operations for customers. It also allows the maintainer to fix the issues of ATM and update its software. The software takes card number and pin as input and the bank account number of the customer is retrieved for further purposes. The outputs then comprise of an interactive display that lets the customer select the desirable function that he/she wants to perform.

2.1 Product Perspective

- The AMS shall be developed using client-server architecture and be compatible with Linux and UNIX based Operation System. The front-end of the system will be developed using PHP, HTML, CSS and the back-end will be developed using PostgreSQL 12.
- The ATM is a single functional unit consisting of various sub- components.
- AMS allows the user to access their bank accounts remotely through an ATM without any aid of human bank teller.
- AMS also allows to perform various other functions apart from just accessing his bank account such as mobile bill clearings etc.
- Some of its hardware components are cassettes, memory, drives, dispensers i.e., for receipts and cash, a card reader, printer, switches, a console, a telephone dialer port, a networking port and disks.
- The ATM communicates with the bank’s central server through a dial-up communication link.

2.1.1 System Interfaces

None

2.1.2 User Interfaces

The AMS will have the following user-friendly and menu-driven interfaces:

a. **Login:** to allow the entry of only authorized users through valid card and pin.

b. **Cash Withdrawal:** to allow account holder to get cash from ATM.

c. **Fund Transfer:** to allow account holder to transfer fund from his/her account to another account holder’s account.

d. **Check Balance:** to allow account holder to see the details of his/her current account balance.

e. **Change Pin:** to allow account holder to change his/her card current pin.

f. **Request Cheque Book:** to allow account holder to send request for a cheque book to the bank.

g. **View Mini Statement:** to see summary of past transactions.

h. **VAD Services:** to provide additional facilities to account holder that are not provided by traditional financial institutes in their premises.

i. **Start ATM Services:** to start services at any ATM where the services are currently stopped.

j. **Stop ATM Services:** to stop ongoing services at any ATM.

k. **Update ATM Software:** to update the current software system of ATM with new software version.

2.1.3 Hardware Interfaces

a. Screen Resolution of at least 800 x 600 or above.

b. Support for touch screen.

c. Support for card reader.

d. ATM systems will be in the networked environment as it is a multi-user system.

2.1.4 Software Interfaces

a. Linux/ Unix Operating System

b. PHP for designing front-end

c. PostgreSQL 12 for back-end

2.1.5 Communication Interfaces

Communication is via local area network (LAN).

2.1.6 Memory Constraints

At least 256 MB RAM and 100 MB space hard disk will be required to run software.

2.1.7 Operations

None

2.1.8 Site Adaptation Requirement

The terminal at ATM will have support for the hardware and software interfaces specified in sections 2.1.3 and 2.1.4 respectively.

2.2 Product Functions

The major functions that AMS performs are described as follows:

- **Account Maintenance:** The various functions that a user can perform with his account are as follows:

a) Withdrawal/Deposit: The software allows the user to select the kind of operation to be performed i.e., whether he/she wants to withdraw or deposit the money.

b) Fund Transfer: Fund transfer shall be facilitated between two accounts linked to the bank.

c) Balance Enquiry: Balance enquiry for any account linked to the card shall be facilitated.

- **Billing:** Any transaction shall be recorded in the form of a receipt and the same would be dispensed to the customer. The billing procedures are handled by the billing module that enable user to choose whether he/she wants the printed statement of the transaction or just the updating in his/her account.
- **Cancelling:** The customer shall abort a transaction with the press of a Cancel key. For example, on entering a wrong depositing amount. In addition, the user can also cancel the entire session by pressing the abort key and can start a fresh session all over again.
- **Mobile Recharge:** The machine also allows the user to recharge his/her mobile number there only, if the name of his/her operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.
- **Bill Payments:** The machine also allows the user to clear off his pending bills there only, if the name of his/her operator is mentioned there in the list. The machine displays the list of the companies supported by that bank to the user.

2.3 User Characteristics

- Qualification: At least matriculation and comfortable with English.
- Experience: Should be well versed/informed about inserting the card in the card reader.
- Technical Knowledge: Elementary knowledge of keypad and touch screen.

2.4 Constraints

- The software does not create bank account.
- User will not be allowed to update their account number.
- ATM must service at most one person at a time.
- The number of invalid pin entries attempted must not exceed three. After three

unsuccessful login attempts, the card is seized/blocked and need to be unlocked by the bank ● The simultaneous access to an account through both, the ATM and the bank is not

supported.

- The minimum amount of money a user can withdraw is ₹ 100/- and the maximum amount of money a user can withdraw in a session is ₹ 10,000/- and the maximum amount he/she can withdraw in a day is ₹ 25,000/-

- Before the transaction is carried out, a check is performed by the machine to ensure that a minimum amount of ₹ 2000/- is left in the user's account after the withdrawal failing which the withdrawal is denied.
- A user can select only that cellular operator for mobile bill clearings that is supported by the bank.
- There shall be a printer installed with the machine to provide the user with the printed statement of the transaction.

2.5 Assumptions and Dependencies

- One major dependency that the project might face is the changes that need to be incorporated with the changes in the bank policies regarding different services. As the policies changes the system needs to be updated with the same immediately. A delay in doing the same will result to tremendous loss to the bank. So, this should be changed as and when required by the developer.
- Another constraint relating to the operating environment is that we are specific to PostgreSQL database.
- The project could be largely affected if some amount is withdrawn from the user's account from the bank at the same time when someone is accessing that account through the ATM machine. Such a condition shall be taken care of.
- At this stage no quantitative measures are imposed on the software in terms of speed and memory although it is implied that all functions will be optimized with respect to speed and memory.
- The account and card are added by the bank to the bank server.

3. External Interface Requirements

3.1 User Interfaces

The following user interfaces (or forms) will be provided by the system.

i. Login Screen -This will be the first form, which will be displayed. It will allow the user to access the different forms based on his/her account.

Various fields available on this form will be:

● Card No.: 12-digit numeric values that is displayed by card reader after reading the card. Special characters and spaces are not allowed.

● Pin: A 4-digit numeric value associated with the card inserted in the card reader. Special characters and spaces are not allowed.

ii. Customer Home -The system provides the customer with variety of options to choose for further operations on ATM.

iii. Cash Withdrawal -This form facilitates the customer to enter amount to money that he/she needs to get from ATM system. The field available on this form will be:

● Cash Amount: A numeric value (multiple of 100) ranging from minimum of ₹ 100 to maximum of ₹ 10000. Special characters and spaces are not allowed.

iv. Fund Transfer- This form facilitates the customer to transfer fund from his/her account to another account holder's account.

Various fields available on this form will be:

● Account Number: A 12-digit number representing the account number of beneficiary. Special characters and spaces are not allowed.

● Amount: A numeric value ranging with minimum of ₹ 100. Special characters and spaces are not allowed.

v. Check Balance- The system facilitates the customer to check his/her account balance.

vi. Change Pin- This form facilitates the customer to change his/her ATM card pin.

Various fields available on this form will be:

● Old Pin: A 4-digit numeric value associated with the card inserted in the card reader. Special characters and spaces are not allowed.

● New Pin: A 4-digit numeric value that customer wants to use as pin with the card inserted in the card reader. Special characters and spaces are not allowed.

vii. Request Cheque Book- This form facilitates the customer to send request for a new cheque book to the bank.

The field available on this form will be:

● Account Number: A 12-digit number representing the account number of customer.

Special characters and spaces are not allowed.

viii. View Mini Statement- The system facilitates the customer to see summary of past transactions.

ix. Value Added Services- The system provides the customer with various additional services apart from basic banking services.

a. Check Reward Points- The system facilitates the customer to check his/her reward points earned using ATM card services.

b. Mobile Recharge- This form facilitates the customer to recharge his/her mobile number using his/her account balance.

Various fields available on this form are:

● Choose Operator: A limited set of mobile service operators supported by ATM provided to customer to choose his/her mobile number operator.

● Mobile Number: A 10-digit numeric value representing the mobile number that is needed to be recharged. Special characters and spaces are not allowed.

● Recharge Amount: A numeric value ranging from ₹ 10 to ₹ 2000 and representing the amount to add to mobile number account. Special characters and spaces are not allowed.

c. Bill Payment- This form facilitates the customer to pay outstanding amount of his/her bills such as electricity, water and PNG using his/her account balance.

Various fields available on this form are:

● Choose Bill Type: A limited set of type of bill services supported by ATM provided to customer to choose his/her bill service type.

● CA Number: A 10-digit alphanumeric value representing the customer bill service account number for which bill payment is needed. Special characters and spaces are not allowed.

● Bill Amount: A numeric value starting from ₹ 10 and representing the amount of bill. Special characters and spaces are not allowed.

x. Maintainer Home- The system provides the maintainer with variety of options to choose for further operations on ATM network.

xi. Start ATM Services- The system facilitates the customer to start services at any ATM where the services are currently stopped.

The field available on this form is:

● ATM ID: A 10-digit alphanumeric values to uniquely identify an ATM. Special characters are allowed but spaces are not allowed.

xii. Update ATM Software- The system facilitates the customer to update the current software system of ATM with new software version.

The field available on this form is:

- ATM ID: A 10-digit alphanumeric values to uniquely identify an ATM. Special characters are allowed but spaces are not allowed.

xiii. Stop ATM Services- The system facilitates the customer to stop ongoing services at any ATM.

The field available on this form is:

- ATM ID: A 10-digit alphanumeric values to uniquely identify an ATM. Special character are allowed but spaces are not allowed.

3.2 Hardware Interfaces

There are various hardware components with which the machine is required to interact. Various hardware interface requirements that need to be fulfilled for successful functioning of the software are as follows:

- The ATM power supply shall have a 10/220 V AC manual switch.
- The ATM card should have the following physical dimensions:

a) Width – 85.47mm-85.72mm

b) Height – 53.92mm-54.03mm

c) Thickness – 0.76mm+0.08mm

- The card reader shall be a magnetic stripe reader.
- The card reader shall have Smart card option.
- The slot for a card in the card reader may include an extra indentation for the embossed area of the card. In effect it acts as a polarization key and may be used to aid the correct insertion orientation of the card. This is an additional characteristic to the magnetic field sensor which operates off the magnetic stripe and is used to open a mechanical gate on devices such as ATMs.
- There shall be a 40-column dot matrix receipt printer.
- There shall be a 40-column dot matrix statement printer.
- The receipt dispenser shall be a maximum of 4" width and 0.5" thickness.
- The statement dispenser shall be a maximum of 5" width and 0.5" thickness.

● The envelope depository shall be a maximum of 4.5" width, 10" length and 0.5" thickness. ● Screen resolution of at least 800 x 600-required for proper and complete viewing of

screens. Higher resolution would not be a problem.

3.3 Software Interface

In order to perform various different functions, this software needs to interact with various other software. So, there are certain software interface requirements that need to be fulfilled which are listed as follows:

● The transaction management software used to manage the transaction and keep track of

resources shall be BMS version 2.0.

● The card management software used to verify pin no and login shall be CMS version 3.0. ● The database used to keep record of user accounts shall be PostgreSQL 12.0.

3.4 Communication Protocols and Interfaces

The machine needs to communicate with the main branch for each session for various functions such as login verification, account access etc. so the following are the various communication interface requirements that are needed to be fulfilled in order to run the software successfully: -

●The system will employ dial-up POS with the central server for low-cost communication. ●The communication protocol used shall be TCP/IP. Protocol used for data transfer shall be

File Transfer Protocol. (FTP)

4. System Features

4.1 System Feature A

Remote Banking and Account Management

4.1.1 Description and Priority

The system is designed to provide the user with the facility of remote banking and perform various other functions at an interface without any aid of human bank teller.

The functioning of the system shall be as follows: -

●At the start, the user is provided with a log in screen and he is required to enter his PIN NO. and Account details which are then verified by the machine. In case of an unsuccessful attempt a user is asked again for his credentials but the maximum number of attempts given to the user is limited to 3 only, failing which his card is blocked and need to be unblocked by the bank for any future use.

●After a successful log in, the user is presented with a list of language. The user can select anyone in the list for interaction with the machine for the entire session. After the language selection the user is also asked whether he wants to fix that language for future use also so that he is never asked for language in future. In addition, there is also a facility for the user to switch to any other language during that session.

●After the language selection, the user is directed towards a main page that displays a set of options/services along with their brief description, enabling the user to understand their functioning. The user can select any of the listed option and can continue with the transaction.

The machine also provides the user with a number of miscellaneous services such as:

The machine lists a set of operators that are supported by the bank. A user can clear off his pending mobile phone bills be selecting his operator.

The machine also has the facility to display a map that marks the location of other ATMs of the same bank in the city. This may help the user to look for the ATM nearest to his destination.

At any moment if the user wants to abort the transaction, he is provided with an option to cancel it. Just by pressing the abort button he can cancel all the changes made so far and can begin with a new transaction.

After the user is finished with his work, for security purpose, he is required to log out and then take his card out of the slot.

Validity Checks: - In order to gain access to the system, the user is required to enter his/her correct user id/pin no and account no failing which his card may be blocked. The user can access only one account at a time and can enter only one account no. Also, if the user is an administrator, he is required to enter his login id in order to access and change the facilities provided by the system.

Sequencing Information: - The information about the users and their account should be entered into the database prior to any of the transactions and the backup be maintained for all account information.

4.1.2 Action/result

If any of the above validation/sequencing flow does not hold true, appropriate error messages will be prompted to the user for doing the needful. Otherwise, the process done or successfully completed. After each transaction user has performed, a receipt is generated that contains all the information about the transaction.

4.1.3 Functional requirements

None

4.2 System feature B

None

5.Other Non-Functional Requirements

5.1 Performance requirements

- The ATM shall provide customers a 24-hour service.
- The card verification time must not exceed 0.8 sec. under normal server workload and 1 sec. under peak server workload.
- The pin number verification time must not exceed 0.3 sec. under normal server workload and 0.5 sec. under peak server workload.
- Account balance display time must not exceed 2 sec. under normal server workload and 3 sec. under peak server workload.
- Account balance transfer time must not exceed 3 sec. under normal server workload and 4 sec. under peak server workload.
- Cash withdrawal transaction time must not exceed 4 sec. under normal server workload and 5 sec. under peak server workload.

- Receipt printing time after must not exceed 3 sec. under normal server and peak server workload.
- Touch screen and button response time must not exceed 5000ms.

5.2 Safety requirements

- User should be provided with only three attempts to enter pin after which card will be blocked for further use.
- There shall be a security camera installed near the ATM.

5.3 Security requirements

- The system shall be compatible with AIMS security standards.
- The system shall have two levels of security i.e. ATM card and pin verification both authenticated by the CMS software.
- The Encryption standard used during pin transmission shall be triple DES.
- The password shall be 6-14 characters long.
- Passwords shall not contain name of customers as they are easy to be hacked.
- Passwords can contain digit, hyphen and underscore.
- User should be provided with only three attempts for login failing which his card needs to be blocked. There shall be a security camera installed near the ATM.
- The product cabinet cover shall be manufactured using Fiber glass for security purposes.

5.4 Software quality attributes

Usability

- The application will be user-friendly and easy to operate and the functions will be easily understandable.

Reliability

- The application will have non-volatile memory system and have a high degree of fault tolerance.

Availability

- The system will have a backup power supply in case of power failures.
- Any abnormal operations shall result in shutting down of the system. After abnormal shutdown of the ATM, the system shall have to be manually restarted by a maintenance personnel.
- There should be no inconsistency introduced in the account in case of abnormal system shutdown.

Maintainability

- The application will be designed in a maintainable manner. It will be easy to incorporate new requirements in the individual modules.
- The system should have the mechanism of self-monitoring periodically in order to detect any fault.
- The system should inform the main branch automatically as soon as it detects any error. The kind of fault and the problem being encountered should also be mentioned by the system automatically.

5.5 Project documentation

None

5.6 User documentation

None

6.Other Requirements

None

CONCLUSION: The SRS was made successfully by following the steps described above.

EXCERCISE NO. 3

Aim: To draw a sample ENTITY RELATIONSHIP DIAGRAM for real project or system.

Hardware Requirements:

Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, coloured monitor.

Software Requirements:

Umbrello, Windows

XP,

Theory:

Entity Relationship Diagrams are a major data modelling tool and will help organize the data in your project into entities and define the relationships between the entities. This process has proved to enable the analyst to produce a good database structure so that the data can be stored and retrieved in a most efficient manner.

Entity:

A data entity is anything real or abstract about which we want to store data. Entity types fall into five classes: roles, events, locations, tangible things or concepts. E.g., employee, payment, campus, book. Specific examples of an entity are called **instances.** E.g., the employee John Jones, Mary Smith's payment, etc.

Relationship:

A data relationship is a natural association that exists between one or more entities. E.g., Employees process payments. **Cardinality** defines the number of occurrences of one entity for a single occurrence of the related entity. E.g., an employee may process many payments but might not process any payments depending on the nature of her job.

Attribute:

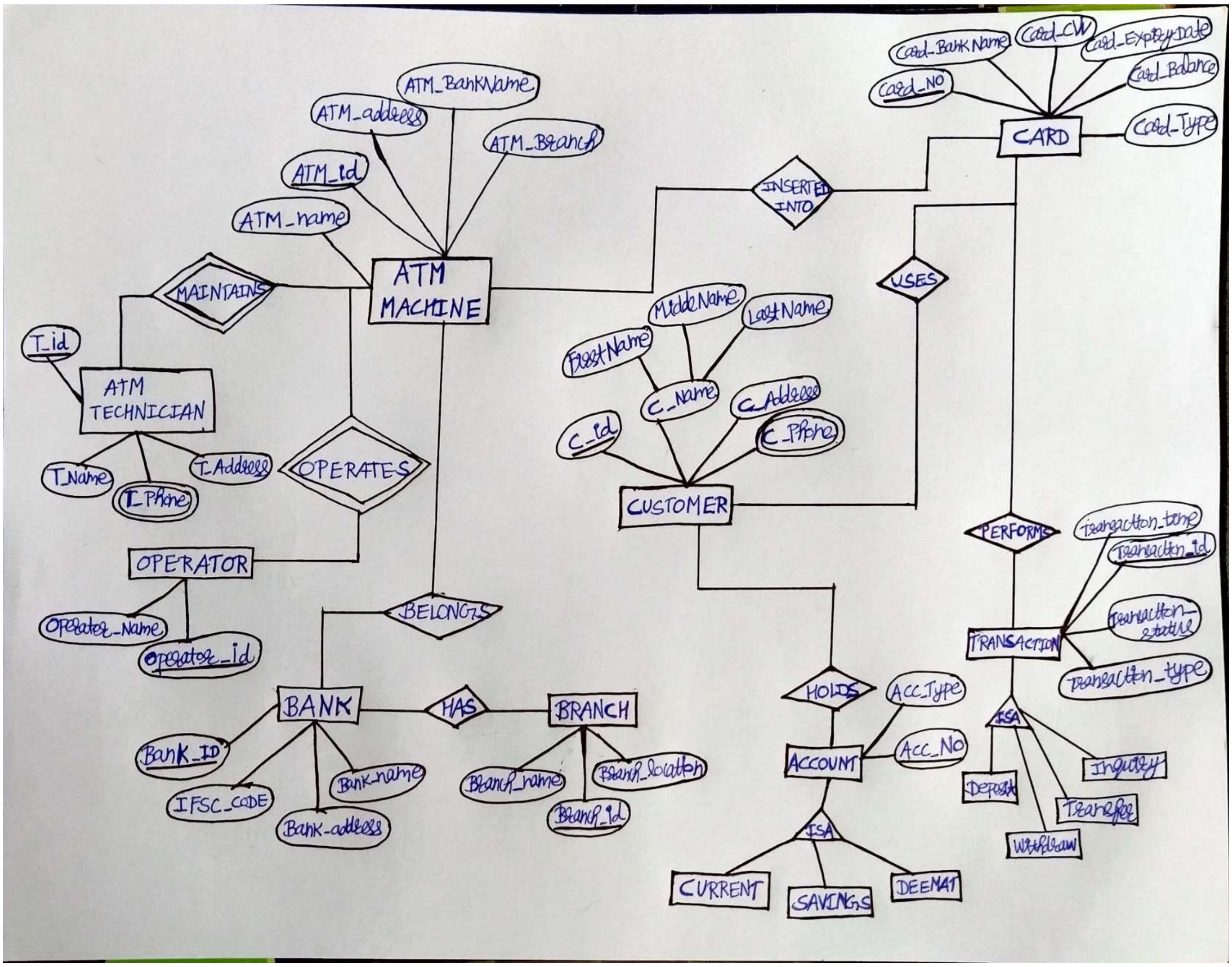
A data attribute is a characteristic common to all or most instances of a particular entity. Synonyms include property, data element, field. E.g., Name, address, Employee Number, pay rate are all attributes of the entity employee. An attribute or combination of attributes that uniquely identifies one and only one instance of an entity is called a **primary key** or **identifier**. E.g., Employee Number is a primary key for Employee.

AN ENTITY RELATIONSHIP DIAGRAM METHODOLOGY:

(One way of doing it)

1. Identify Entities	Identify the roles, events, locations, tangible things or concepts about which the end-users want to store data.
2. Find Relationships	Find the natural associations between pairs of entities using a relationship matrix.
3. Draw Rough ERD	Put entities in rectangles and relationships on line segments connecting the entities.
4. Fill in Cardinality	Determine the number of occurrences of one entity for a single occurrence of the related entity.
5. Define Primary Keys	Identify the data attribute(s) that uniquely identify one and only one occurrence of each entity.
6. Draw Key-Based ERD	Eliminate Many-to-Many relationships and include primary and foreign keys in each entity.
7. Identify Attributes	Name the information details (fields) which are essential to the system under development.
8. Map Attributes	For each attribute, match it with exactly one entity that it describes.
9. Draw fully attributed ERD	Adjust the ERD from step 6 to account for entities or relationships discovered in step 8.
10. Check Results	Does the final Entity Relationship Diagram accurately depict the system? data?

Conclusion: The entity relationship diagram was made successfully by following the steps described above.



Conclusion: The entity relationship diagram for ATM Management System has been made successfully by following the steps described above.

EXCERCISE NO. 4

Aim: To prepare DATA FLOW DIAGRAM for ATM Management System.

REQUIREMENTS:

Hardware Interfaces

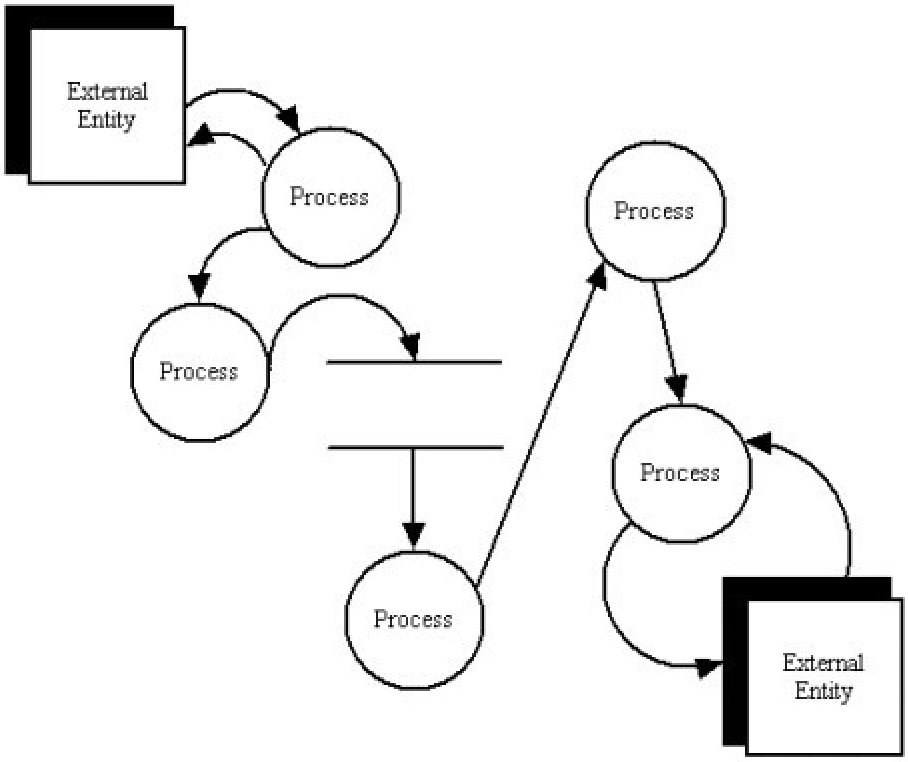
- Pentium(R) 4 CPU 2.26 GHz, 128 MB RAM
- Screen resolution of at least 800 x 600 required for proper and complete viewing of screens. Higher resolution would not be a problem.
- CD ROM Driver

Software Interfaces

- Any window-based operating system (Windows 95/98/2000/XP/NT)
- WordPad or Microsoft Word

THEORY:

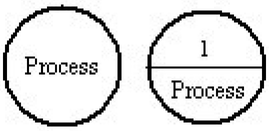
Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs.



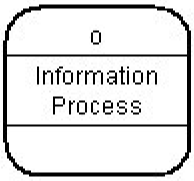
Data Flow Diagram Notations

You can use two different types of notations on your data flow diagrams: *Yourdon & Coad* or *Gane & Sarson*.

Process Notations



Yourdon and Coad
Process Notations

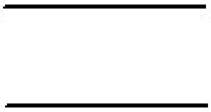


Gane and Sarson
Process Notation

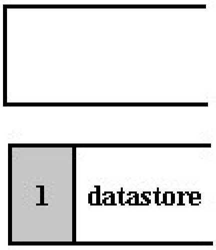
Process

A process transforms incoming data flow into outgoing data flow.

Datastore Notations



Yourdon and Coad
Datastore Notations

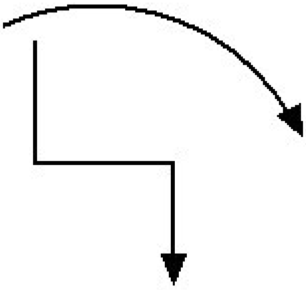


Gane and Sarson
Datastore Notations

DataStore

Datastores are repositories of data in the system. They are sometimes also referred to as files.

Dataflow Notations

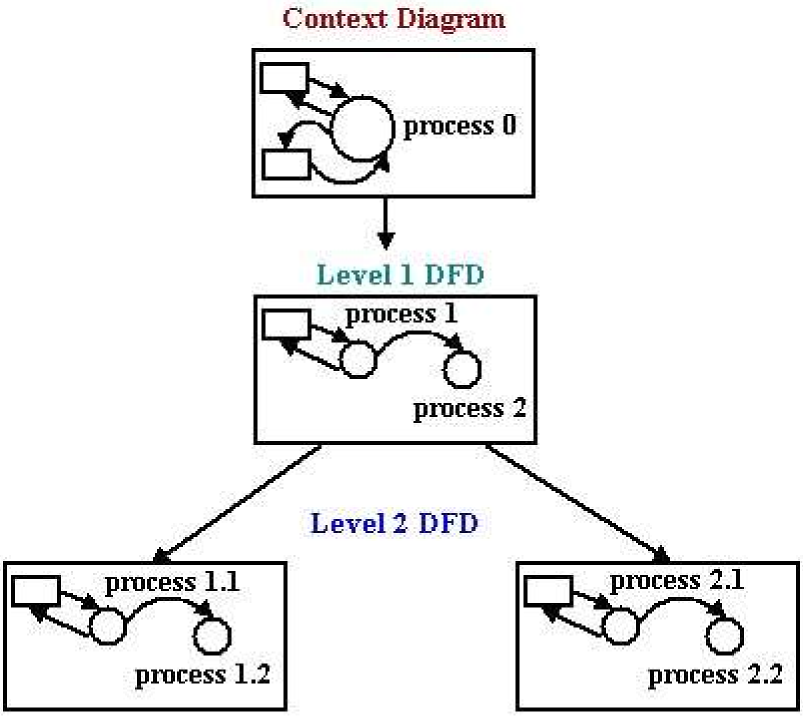


Dataflow
Dataflows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.

HOW TO DRAW DATA FLOW DIAGRAMS (cont'd)

Data Flow Diagram Layers

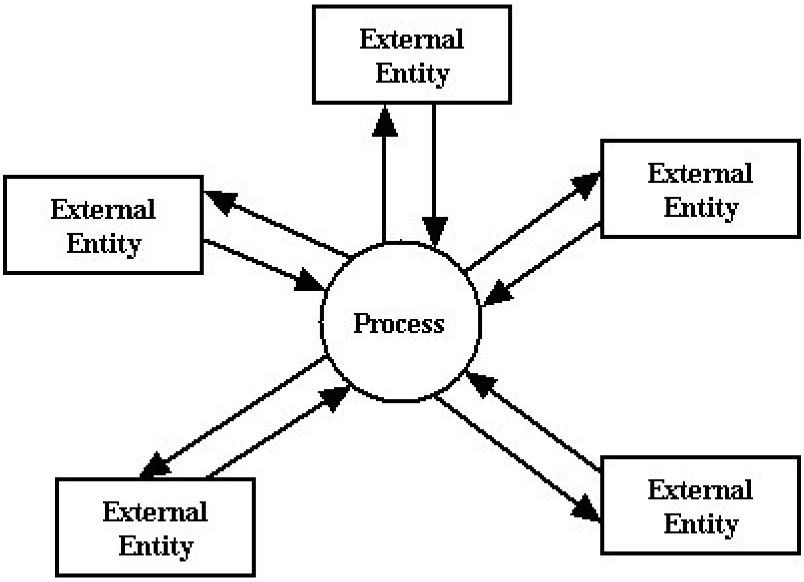
Draw data flow diagrams in several nested layers. A single process node on a high-level diagram can be expanded to show a more detailed data flow diagram. Draw the context diagram first, followed by various layers of data flow diagrams.



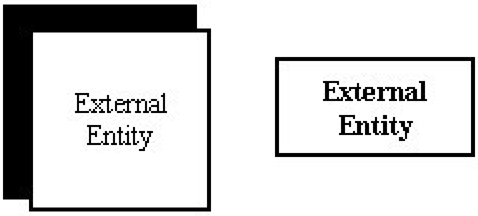
The nesting of data flow layers

Context Diagrams

A context diagram is a top level (also known as Level 0) data flow diagram. It only contains one process node (process 0) that generalizes the function of the entire system in relationship to external entities.



External Entity Notations

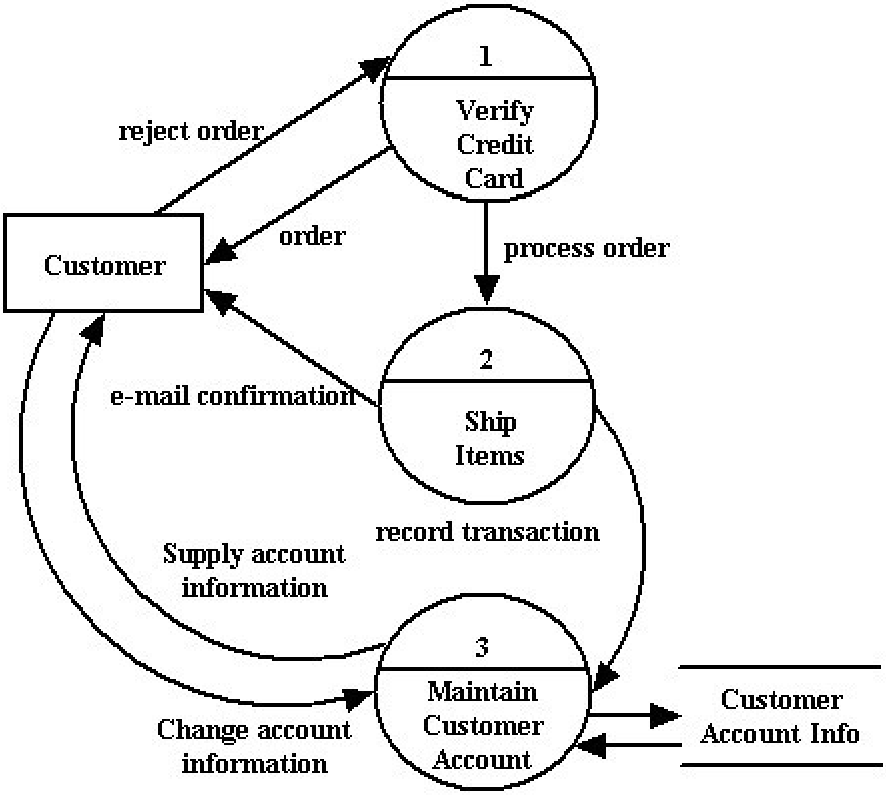


External Entity

External entities are objects outside the system, with which the system communicates. External entities are sources and destinations of the system's inputs and outputs.

DFD levels

The first level DFD shows the main processes within the system. Each of these processes can be broken into further processes until you reach pseudocode.

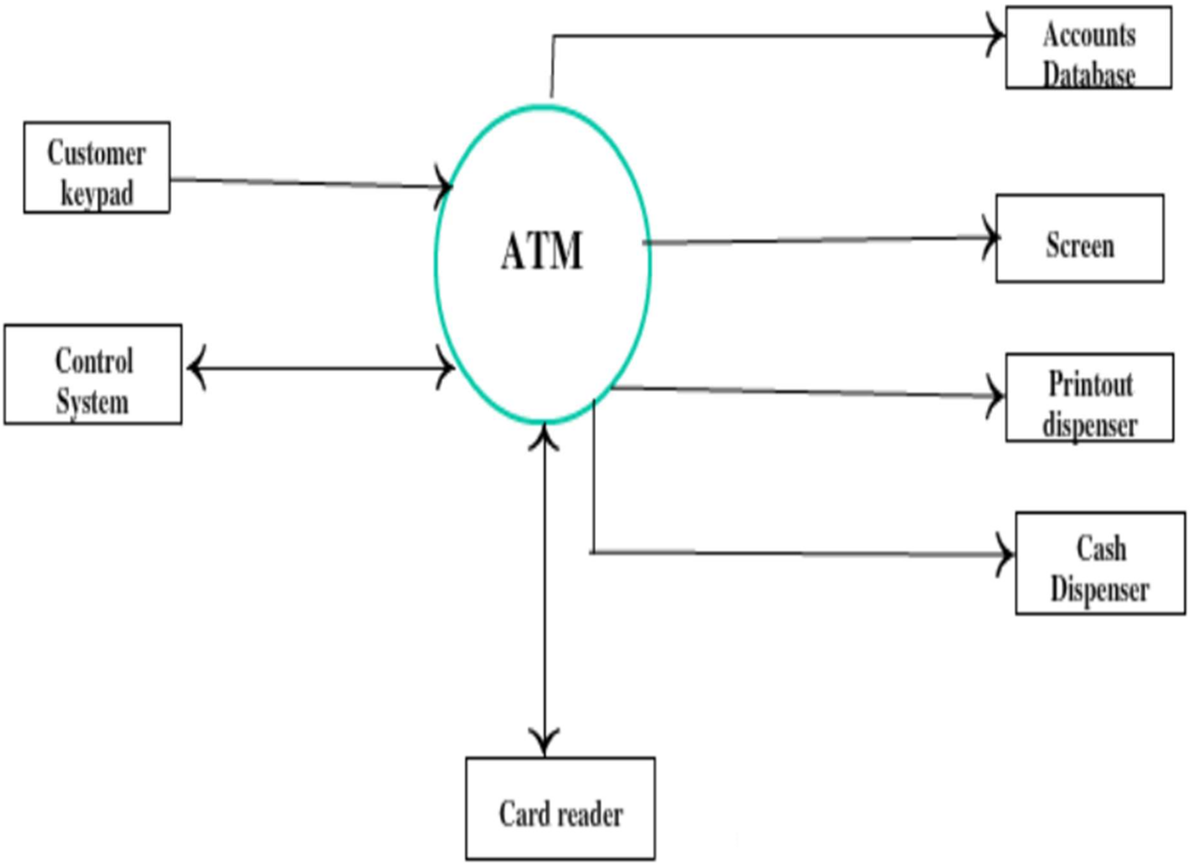


An example first-level data flow diagram

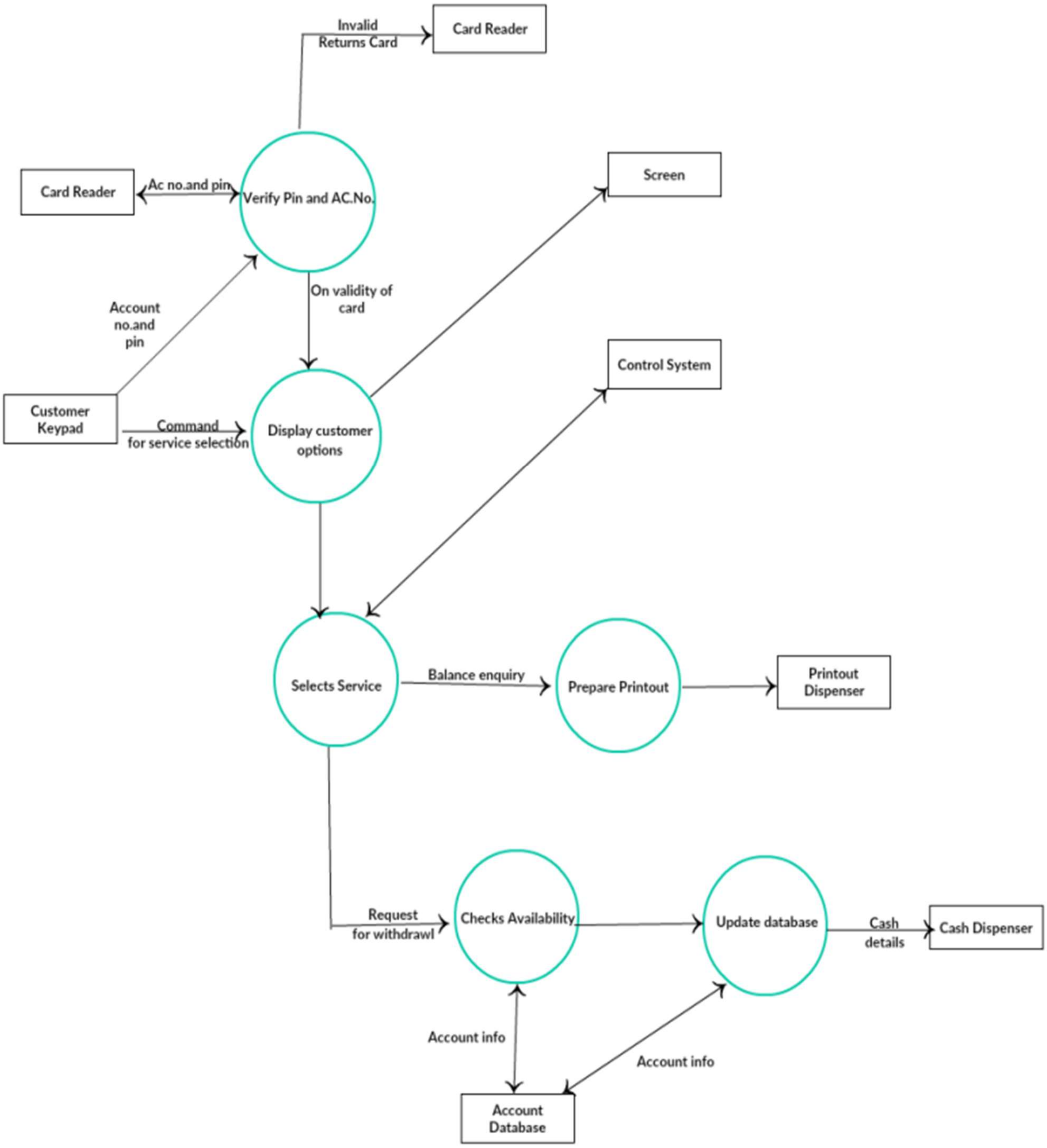
Conclusion: The dataflow diagram was made successfully by following the steps described above.

DFD OF ATM Management System

Level-0 DFD of ATM management system



Level-1 DFD of ATM management system



Conclusion: The dataflow diagram for ATM Management System was made successfully by following the steps described above.

EXERCISE NO. 5

Aim: Steps to draw the Use Case Diagram using Rational Rose.

Hardware Requirements:

Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, colored monitor.

Software Requirements:

Rational Rose, Windows XP,

Theory:

According to the UML specification a use case diagram is "a diagram that shows the relationships among actors and use cases within a system." Use case diagrams are often used to:

- Provide an overview of all or part of the usage requirements for a system or organization in the form of an essential model or a business model
- Communicate the scope of a development project
- Model your analysis of your usage requirements in the form of a system use case model

Use case models should be developed from the point of view of your project stakeholders and not from the (often technical) point of view of developers. There are guidelines for:

Use Cases

Actors

Relationships

System Boundary Boxes

1. Use Cases

A use case describes a sequence of actions that provide a measurable value to an actor. A use case is drawn as a horizontal ellipse on a UML use case diagram.

1. Use Case Names Begin With a Strong Verb
2. Name Use Cases Using Domain Terminology
3. Place Your Primary Use Cases In The Top-Left Corner Of The Diagram 4. Imply Timing Considerations By Stacking Use Cases.

2. Actors

An actor is a person, organization, or external system that plays a role in one or more interactions with your system (actors are typically drawn as stick figures on UML Use Case diagrams).

1. Place Your Primary Actor(S) In The Top-Left Corner Of The Diagram
2. Draw Actors To The Outside Of A Use Case Diagram
3. Name Actors With Singular, Business-Relevant Nouns
4. Associate Each Actor With One Or More Use Cases
5. Actors Model Roles, Not Positions
6. Use <<system>> to Indicate System Actors
7. Actors Don't Interact With One Another
8. Introduce an Actor Called "Time" to Initiate Scheduled Events

3. Relationships

There are several types of relationships that may appear on a use case diagram:

- An association between an actor and a use case
- An association between two use cases
- A generalization between two actors
- A generalization between two use cases

Associations are depicted as lines connecting two modeling elements with an optional openheaded arrowhead on one end of the line indicating the direction of the initial invocation of the relationship. Generalizations are depicted as a close-headed arrow with the arrow pointing towards the more general modeling element.

1. Indicate An Association Between An Actor And A Use Case If The Actor Appears Within The Use Case Logic
2. Avoid Arrowheads On Actor-Use Case Relationships
3. Apply <<include>> When You Know Exactly When To Invoke The Use Case
4. Apply <<extend>> When A Use Case May Be Invoked Across Several Use Case Steps 5. Introduce <<extend>> associations sparingly
6. Generalize Use Cases When a Single Condition Results In Significantly New Business Logic
7. Do Not Apply <<uses>>, <<includes>>, or <<extends>>

8. Avoid More Than Two Levels Of Use Case Associations
9. Place An Included Use Case To The Right Of The Invoking Use Case
10. Place An Extending Use Case Below The Parent Use Case
11. Apply the “Is Like” Rule to Use Case Generalization
12. Place an Inheriting Use Case Below The Base Use Case
13. Apply the “Is Like” Rule to Actor Inheritance
14. Place an Inheriting Actor Below the Parent Actor

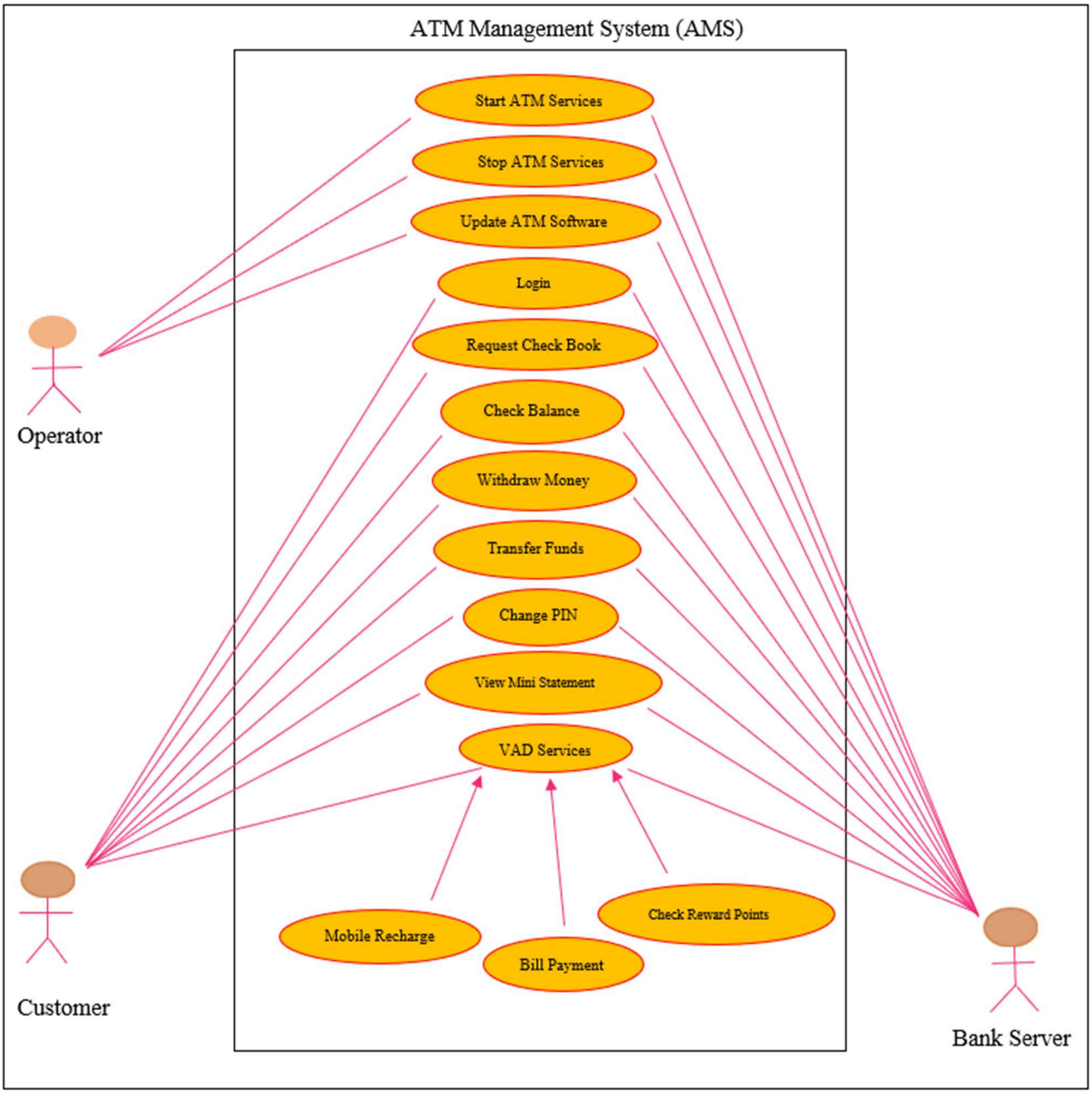
4. System Boundary Boxes

The rectangle around the use cases is called the system boundary box and as the name suggests it indicates the scope of your system – the use cases inside the rectangle represent the functionality that you intend to implement.

1. Indicate Release Scope with a System Boundary Box.
2. Avoid Meaningless System Boundary Boxes.

Conclusion: Use Case Diagram has been drawn Successfully.

ATM MANAGEMENT SYSTEM USE CASE DIAGRAM



Conclusion: Use Case Diagram for ATM Management System has been drawn Successfully.

EXERCISE NO. 6

Aim: To draw a sample activity diagram for real project or system.

Hardware Requirements:

Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, colored monitor.

Software Requirements:

Rational Rose, Windows XP,

THEORY

UML 2 activity diagrams are typically used for business process modeling, for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Although UML activity diagrams could potentially model the internal logic of a complex operation it would be far better to simply rewrite the operation so that it is simple enough that you don't require an activity diagram. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structured development.

Let's start by describing the basic notation :

- **Initial node**. The filled in circle is the starting point of the diagram. An initial node isn't required although it does make it significantly easier to read the diagram.
- **Activity final node**. The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes.
- **Activity**. The rounded rectangles represent activities that occur. An activity may be physical, such as *Inspect Forms*, or electronic, such as *Display Create Student Screen*.
- **Flow/edge**. The arrows on the diagram. Although there is a subtle difference between flows and edges,never a practical purpose for the difference although.
- **Fork**. A black bar with one flow going into it and several leaving it. This denotes the beginning of parallel activity.
- **Join**. A black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue. This denotes the end of parallel processing.
- **Condition**. Text such as *[Incorrect Form]* on a flow, defining a guard which must evaluate to true in order to traverse the node.
- **Decision**. A diamond with one flow entering and several leaving. The flows leaving include conditions although some modelers will not indicate the conditions if it is obvious.
- **Merge**. A diamond with several flows entering and one leaving. The implication is that one or more incoming flows must reach this point until processing continues, based on any guards on the outgoing flow.

- **Partition**. If figure is organized into three partitions, it is also called swimlanes, indicating who/what is performing the activities (either the *Applicant*, *Registrar*, or *System*).
- **Sub-activity indicator**. The rake in the bottom corner of an activity, such as in the *Apply to University* activity, indicates that the activity is described by a more finely detailed activity diagram.
- **Flow final**. The circle with the X through it. This indicates that the process stops at this point.

GUIDELINES ASSOCIATED FOR DRAWING AN ACTIVITY DIAGRAM

1.General Guidelines

2.Activities

3.Decision Points

4.Guards

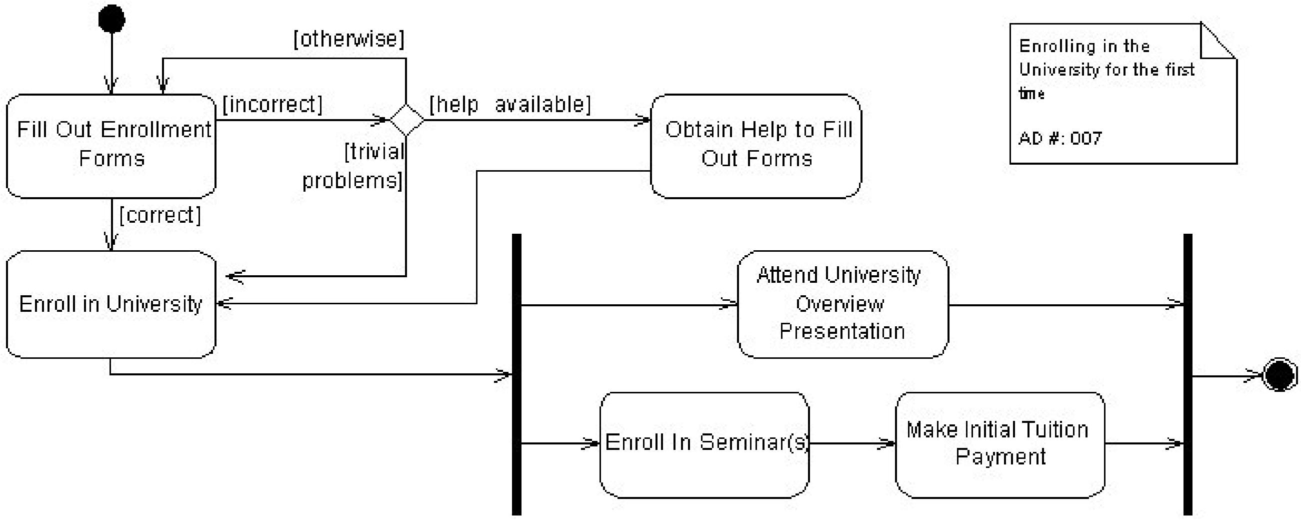
5.Parallel Activities

6.Swimlane Guidelines

7.Action-Object Guidelines

1. General Guidelines

Figure1. Modeling a business process with a UML Activity Diagram.


[otherwise]
[incorrect]
[help available]
Fill Out Enrollment Forms
[trivial problems]
Obtain Help to Fill Out Forms
Enrolling in the University for the first time
AD #: 007
[correct]
Enroll in University
Attend University Overview Presentation
Enroll In Seminar(s)
Make Initial Tuition Payment

1. Place The Start Point In The Top-Left Corner. A start point is modeled with a filled in circle, using the same notation that UML State Chart diagrams use. Every UML Activity Diagram should have a starting point, and placing it in the top-left corner reflects the way that people in Western cultures begin reading. Figure1, which models the business process of enrolling in a university, takes this approach.
2. Always Include an Ending Point. An ending point is modeled with a filled in circle with a border around it, using the same notation that UML State Chart diagrams use. Figure1 is interesting because it does not include an end point because it describes a continuous process – sometimes the guidelines don't apply.
3. Flowcharting Operations Implies the Need to Simplify. A good rule of thumb is that if an operation is so complex you need to develop a UML Activity diagram to understand it that you should consider refactoring it.

2. Activities

An activity, also known as an activity state, on a UML Activity diagram typically represents the invocation of an operation, a step in a business process, or an entire business process.

1. Question "Black Hole" Activities. A black hole activity is one that has transitions into it but none out, typically indicating that you have either missed one or more transitions.
2. Question "Miracle" Activities. A miracle activity is one that has transitions out of it but none into it, something that should be true only of start points.

3. Decision Points

A decision point is modeled as a diamond on a UML Activity diagram.

1. Decision Points Should Reflect the Previous Activity. In figure1 we see that there is no label on the decision point, unlike traditional flowcharts which would include text describing the actual decision being made, we need to imply that the decision concerns whether the person was enrolled in the university based on the activity that the decision point follows. The guards, depicted using the format *[description]*, on the transitions leaving the decision point also help to describe the decision point.
2. Avoid Superfluous Decision Points. The *Fill Out Enrollment Forms* activity in FIGURE1 includes an implied decision point, a check to see that the forms are filled out properly, which simplified the diagram by avoiding an additional diamond.

4. Guards

A guard is a condition that must be true in order to traverse a transition.

1. Each Transition Leaving a Decision Point Must Have a Guard
2. Guards Should Not Overlap. For example guards such as x <0, x = 0, and x > 0 are consistent whereas guard such as x <= 0 and x >= 0 are not consistent because they overlap – it isn't clear what should happen when x is 0.

3. Guards on Decision Points Must Form a Complete Set. For example, guards such as x < 0 and x >0 are not complete because it isn't clear what happens when x is 0.
4. Exit Transition Guards and Activity Invariants Must Form a Complete Set. An activity invariant is a condition that is always true when your system is processing an activity.
5. Apply a [Otherwise] Guard for "Fall Through" Logic.
6. Guards Are Optional. It is very common for a transition to not include a guard, even when an activity includes several exit transitions.

5. Parallel Activities

It is possible to show that activities can occur in parallel, as you see in FIGURE 1 depicted using two parallel bars. The first bar is called a fork, it has one transition entering it and two or more transitions leaving it. The second bar is a join, with two or more transitions entering it and only one leaving it.

1. A Fork Should Have a Corresponding Join. In general, for every start (fork) there is an end (join). In UML 2 it is not required to have a join, but it usually makes sense.
2. Forks Have One Entry Transition.
3. Joins Have One Exit Transition
4. Avoid Superfluous Forks. FIGURE 2 depicts a simplified description of the software process of enterprise architectural modeling, a part of the Enterprise Unified Process (EUP). There is significant opportunity for parallelism in this process, in fact all of these activities could happen in parallel, but forks were not introduced because they would only have cluttered the diagram.

6. Swimlane Guidelines

A swimlane is a way to group activities performed by the same actor on an activity diagram or to group activities in a single thread. FIGURE 2 includes three swimlanes, one for each actor.

Figure2. A UML activity diagram for the enterprise architectural modeling (simplified).

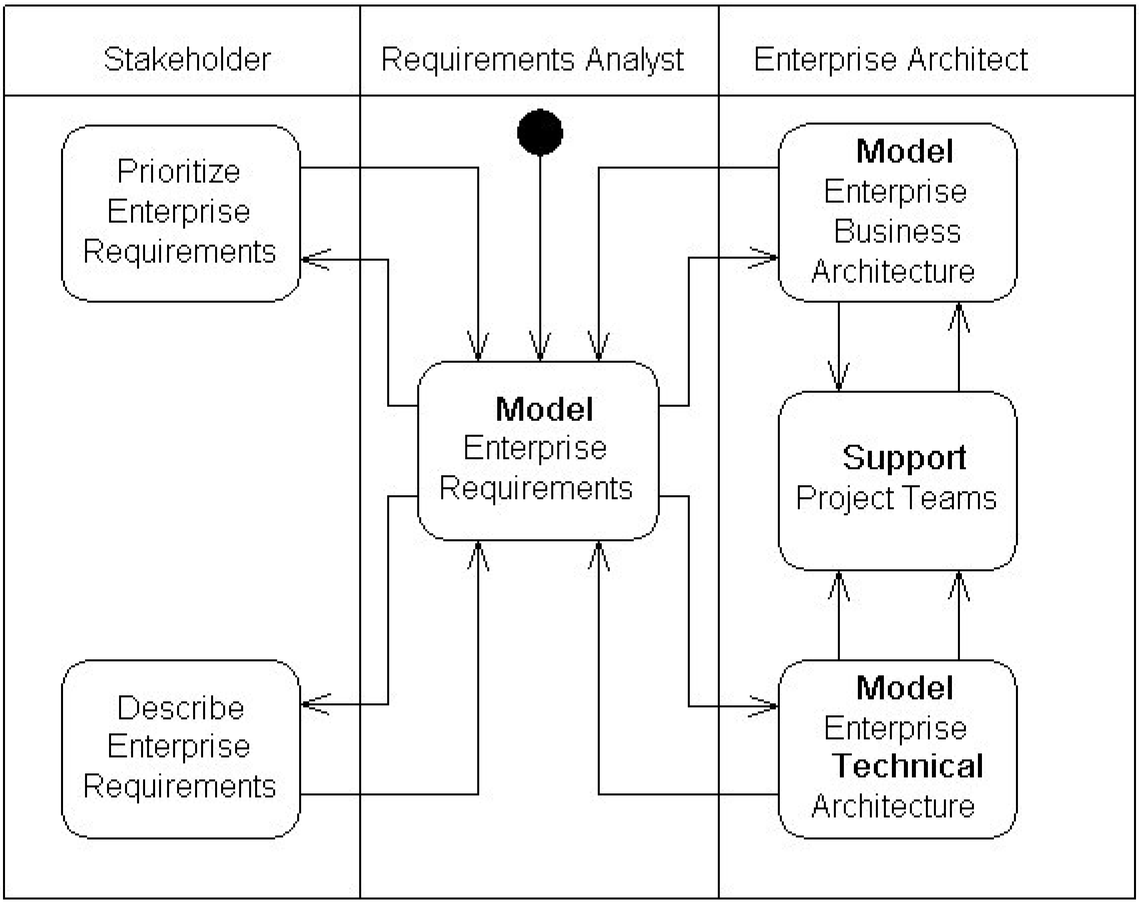
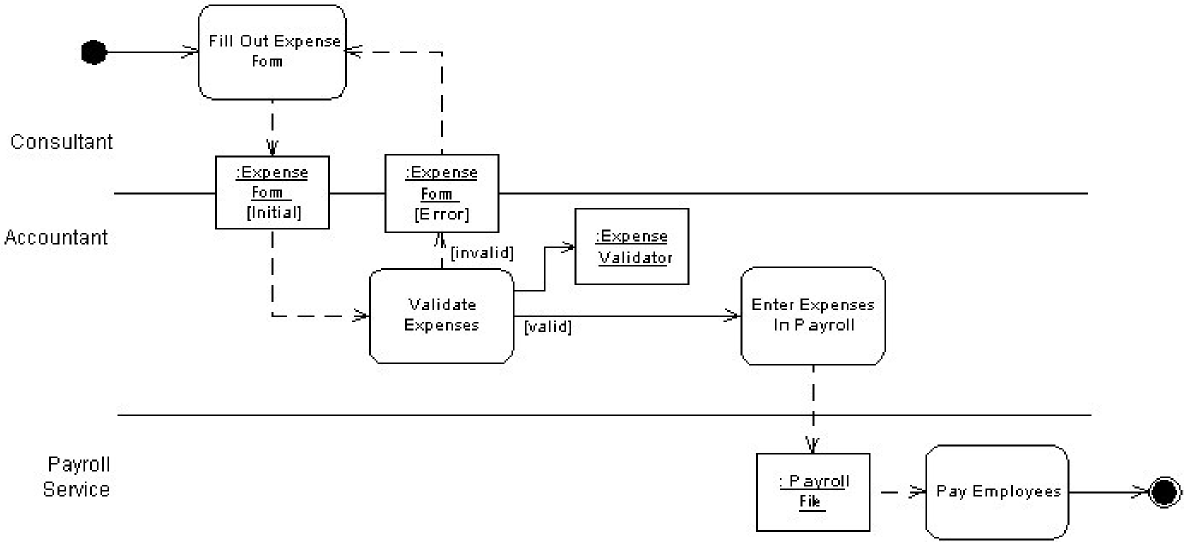

Stakeholder
Requirements Analyst
Enterprise Architect
Prioritize Enterprise Requirements
Model Enterprise Requirements
Model Enterprise Business Architecture
Support Project Teams
Describe Enterprise Requirements
Model Enterprise Technical Architecture

Figure 3. Submitting expenses.



1. Order Swimlanes in a Logical Manner.
2. Apply Swim Lanes To Linear Processes. A good rule of thumb is that swimlanes are best applied to linear processes, unlike the one depicted in FIGURE 3.
3. Have Less Than Five Swimlanes.
4. Consider Swimareas For Complex Diagrams.
5. Swimareas Suggest The Need to Reorganize Into Smaller Activity Diagrams.
6. Consider Horizontal Swimlanes for Business Processes. In FIGURE 3 you see that the swimlanes are drawn horizontally, going against common convention of drawing them vertically.

7 Action-Object Guidelines

Activities act on objects, In the strict object-oriented sense of the term an action object is a system object, a software construct. In the looser, and much more useful for business application modeling, sense of the term an action object is any sort of item. For example in FIGURE 3 the *ExpenseForm* action object is likely a paper form.

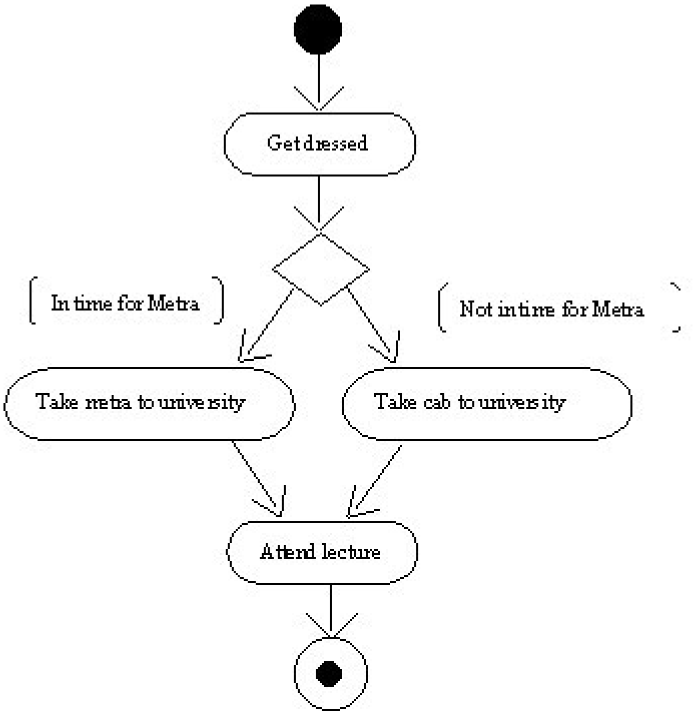
1. Place Shared Action Objects on Swimlane Separators
2. When An Object Appears Several Time Apply State Names
3. State Names Should Reflect the Lifecycle Stage of an Action Object
4. Show Only Critical Inputs and Outputs
5. Depict Action Objects As Smaller Than Activities

Conclusion: The activity diagram was made successfully by following the steps described above.

SAMPLE ACTIVITY DIAGRAMS:

SAMPLE 1:

Let us consider the example of attending a course lecture, at 8 am.



An example Activity diagram

As you can see in Figure , the first activity is to get dressed to leave for the lecture. A decision then has to be made, depending on the time available for the lecture to start, and the timings of the public trains (metra). If there is sufficient time to catch the train, then take the train; else, flag down a cab to the University. The final activity is to actually attend the lecture, after which the Activity diagram terminates.

SAMPLE 2:

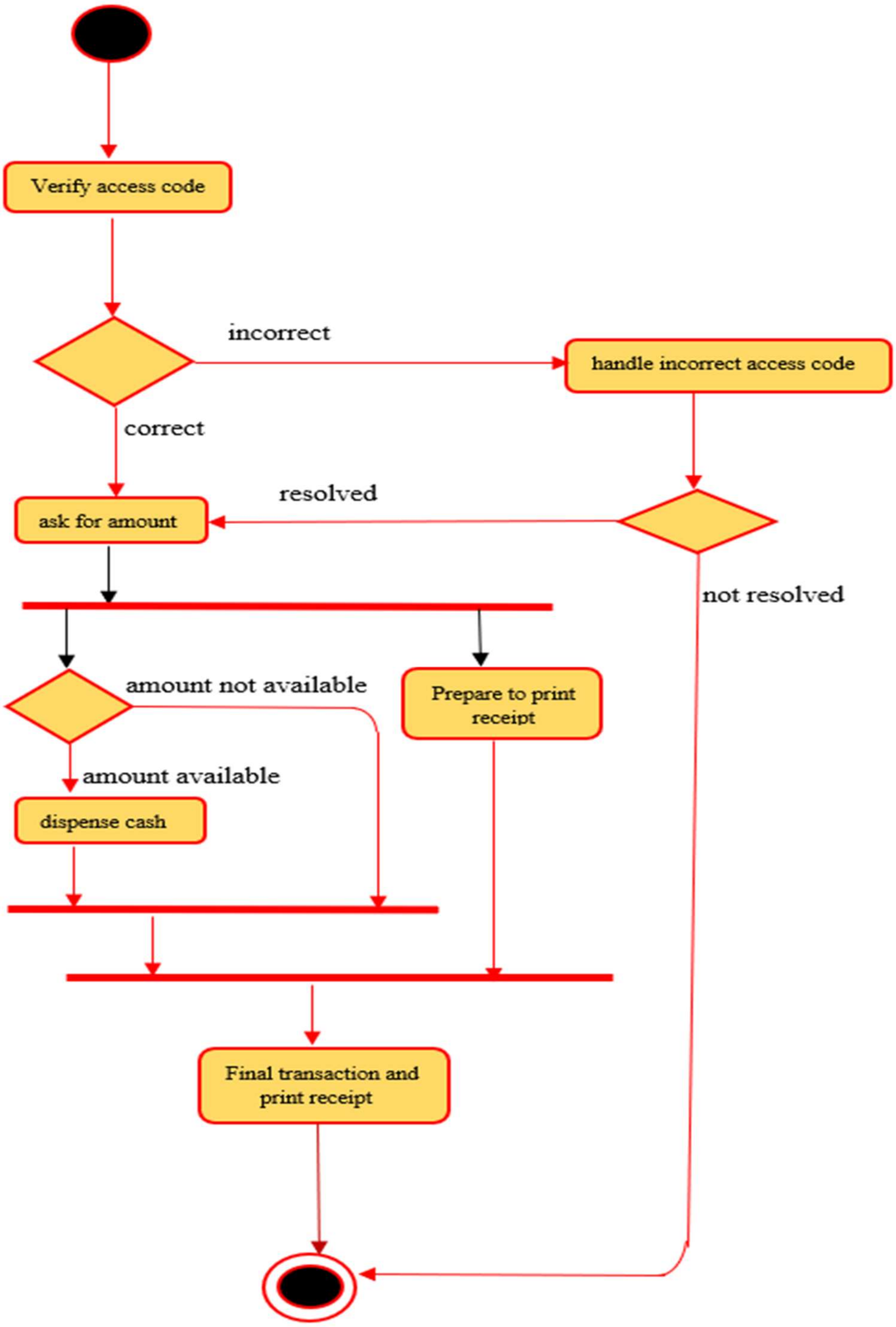
Identifying the activities and transitions for managing course information

The course administrator is responsible for managing course information in the Courseware Management System. As part of managing the course information, the course administrator carries out the following activities:

- Check if course exists
- If course is new, proceed to the "Create Course" step
- If course exists, check what operation is desired—whether to modify the course or remove the course
- If the modify course operation is selected by the course administrator, the "Modify Course" activity is performed
- If the remove course operation is selected by the course administrator, the "Remove Course" activity is performed

In the first step in this Activity diagram, the system determines whether the course that is to be managed is a new course or an existing course. For managing a new course, a separate activity, "Create Course," is performed. On the other hand, if a course exists, the course administrator can perform two different activities—modify an existing course or remove an existing course. Hence, the system checks the type of operation desired based on which two separate activities can be performed—"Modify Course" or "Remove Course".

Activity Diagram for ATM Management System:



Conclusion: The activity diagram for ATM Management system has been made successfully by following the steps described above.

EXERCISE NO. 7

Aim: To prepare STATE CHART DIAGRAM for any project.

REQUIREMENTS:

Hardware Interfaces

- Pentium(R) 4 CPU 2.26 GHz, 128 MB RAM
- Screen resolution of at least 800 x 600 required for proper and complete viewing of screens. Higher resolution would not be a problem.
- CD ROM Driver

Software Interfaces

- Any window-based operating system(Windows98/2000/XP/NT) ▪ IBM Rational Rose Software

THEORY:

- ✓ State Chart Diagrams provide a way to model the various states in which an object can exist.
- ✓ There are two special states: the start state and the stop state.

The **Start state** is represented by a **block dot**.

The **Stop state** is represented by a **bull's eye**.

- ✓ A condition enclosed in square brackets is called a **guard condition,** and controls when a transition can or cannot occur.
- ✓ **Process** that occur while an object is in certain state are called **actions.**

STEPS TO DRAW STATE CHART DIAGRAM IN RATIONAL ROSE SOFTWARE

- To insert new state diagram secondary click on Logical View.
  - ❖ **Select---New----State chart Diagram**.
  - ❖ A new diagram will be created, type in a name for the new diagram.
- Now double click on the new diagram to open it on the stage.
  - ❖ To begin the diagram click on the **"START STATE"** button.
  - ❖ Place a start state icon on the diagram by clicking the mouse once.
- Now add states to the diagram, these make up the content of the diagram. Click on the state button. Place the instances for each state into the diagram and type in names for them.

- Now arrange the states to fill the diagram better. Drag the states to new positions to make the easiest layout to work with.

- Add an end state to the diagram by clicking the **“END STATE”** button. Place an instance into the diagram. Now add relationships to the diagram.

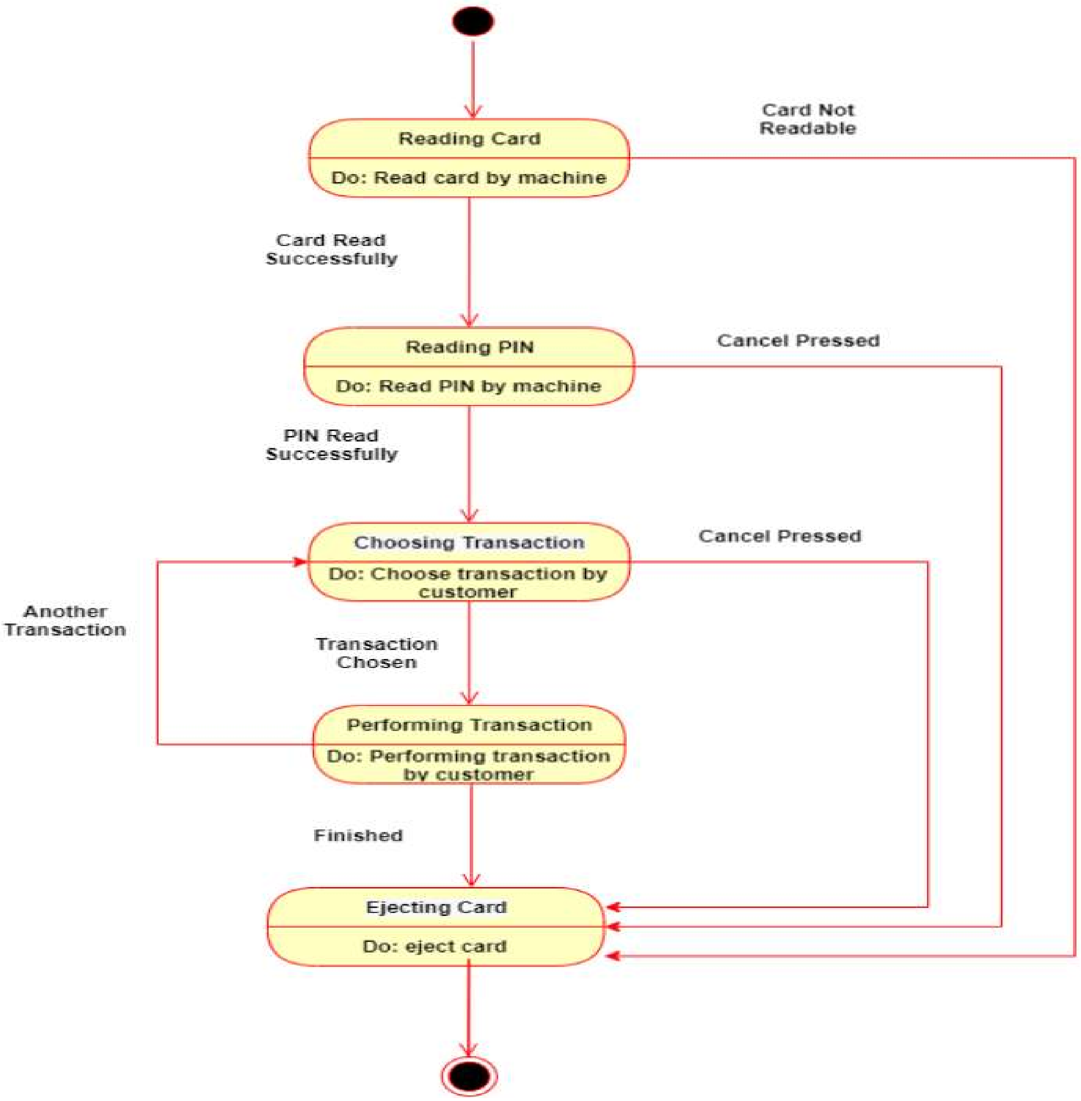
- Click on the **“STATE TRANSITION”** button and drag arrows between the appropriate states.

- To edit the specification secondary click on the relation lines and select **“OPEN SPECIFICATION”** button. Add a name for the event in the specification. Then click on “apply” and then on “OK” button.
- Add details to the specifications of the other relationships in the same way.

- There may be instances on the diagram where a state can join more than one state. In this case add a relationship in the same way. Then enter the specification for the new state.

Conclusion: The state chart diagram of ATM Management System was made successfully by following the steps described above.

State Chart Diagram of ATM Management System



Conclusion: The state chart diagram of ATM Management System was made successfully by following the steps described above.

EXERCISE NO. 8

Aim: Steps to draw the Sequence Diagram using Rational Rose.

Hardware Requirements:

Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard and mouse, colored monitor.

Software Requirements:

Rational Rose, Windows XP

Theory:

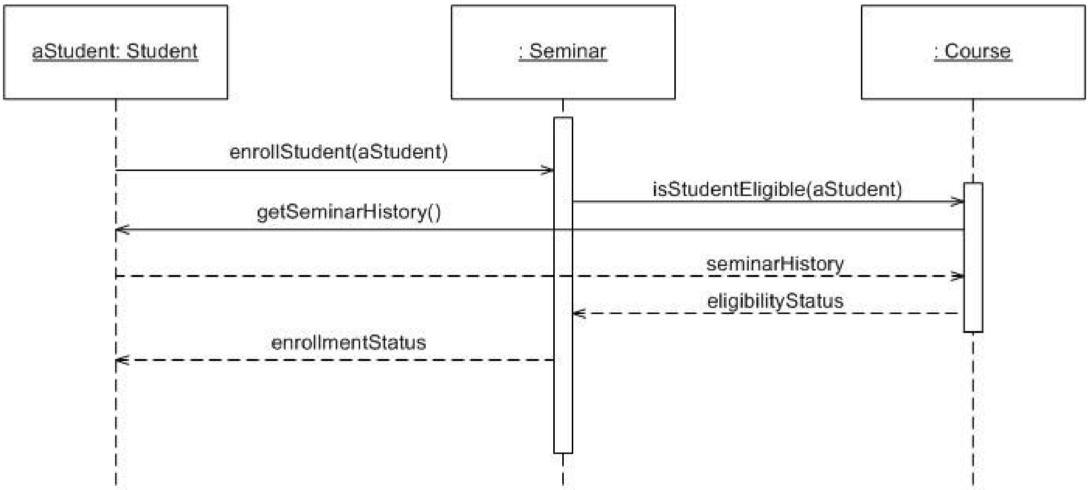
UML sequence diagrams model the flow of logic within the system in a visual manner, enabling the user both to document and validate the logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within your system. Sequence diagrams, along with class diagrams and physical data models are the most important design-level models for modern application development.

Sequence diagrams are typically used to model:

1. **Usage scenarios**. A usage scenario is a description of a potential way the system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several use cases. For example, a student enrolls in the university, and then immediately enrolls in three seminars.
2. **The logic of methods**. Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as visual object code.
3. **The logic of services**. A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies such as CICS/COBOL or CORBA-compliant object request brokers (ORBs).

FIG .shows the logic for how to enroll in a seminar. One should often develop a system-level sequence diagram to help both visualize and validate the logic of a usage scenario. It also helps to identify significant methods/services, such as checking to see if the applicant already exists as a student, which the system must support.

Figure 3. Enrolling in a seminar (method).



The dashed lines hanging from the boxes are called object lifelines, representing the life span of the object during the scenario being modeled. The long, thin boxes on the lifelines are activation boxes, also called method-invocation boxes, which indicate processing is being performed by the target object/class to fulfill a message.

How to Draw Sequence Diagrams

Sequence diagramming really is visual coding, even when you are modeling a usage scenario via a system-level sequence diagram.

While creating a sequence diagram ,start by identifying the scope of what you are trying to model.You should typically tackle small usage scenarios at the system level or a single method/service at the detailed object level.

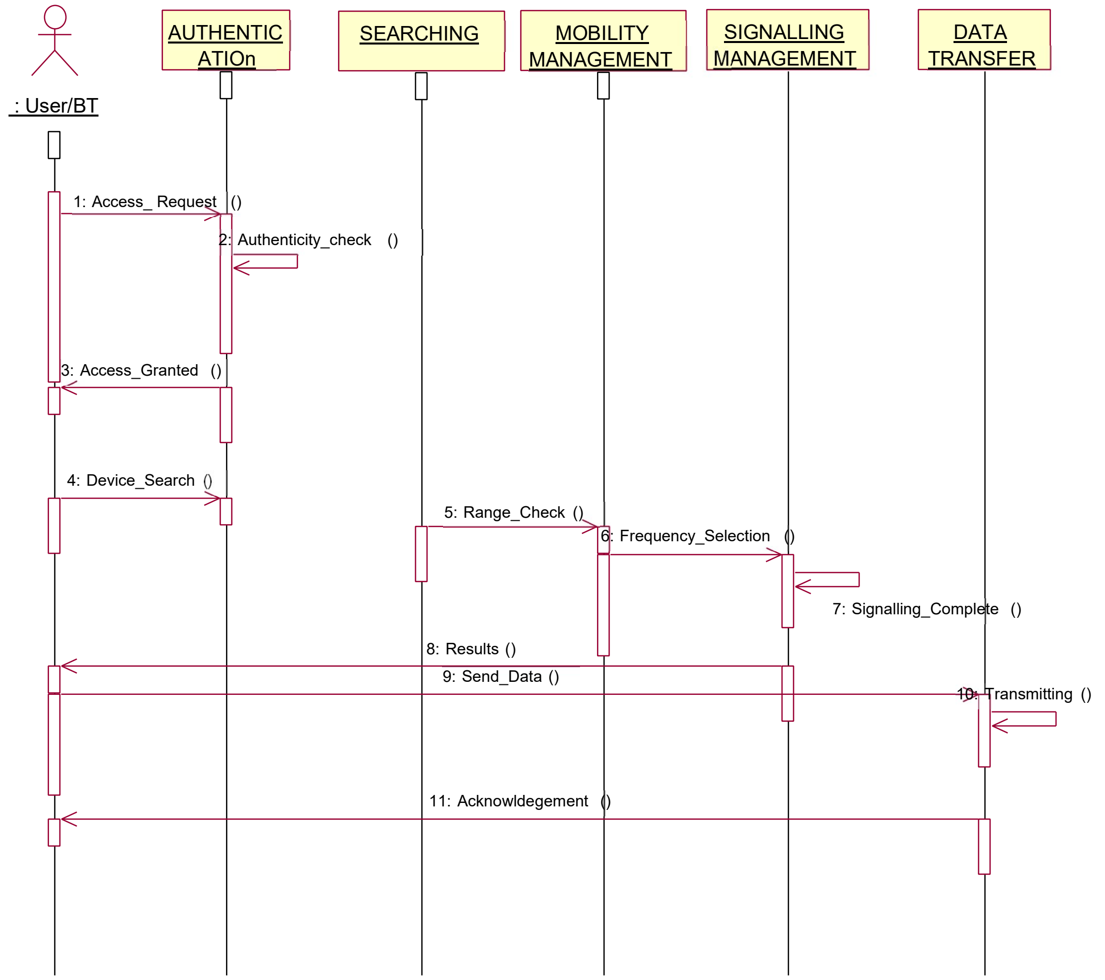
You should then work through the logic with at least one more person, laying out classifiers across the top as you need them. . The heart of the diagram is in the messages, which you add to the diagram one at a time as you work through the logic. You should rarely indicate return values, instead you should give messages intelligent names which often make it clear what is being returned.

It is interesting to note that as you sequence diagram you will identify new responsibilities for classes and objects, and, sometimes, even new classes. The implication is that you may want to update your class model appropriately, agile modelers will follow the practice *Create Several Models in Parallel*, something that CASE tools will do automatically. Remember, each message sent to a class invokes a static method/operation on that class each message sent to an object invokes an operation on that object.

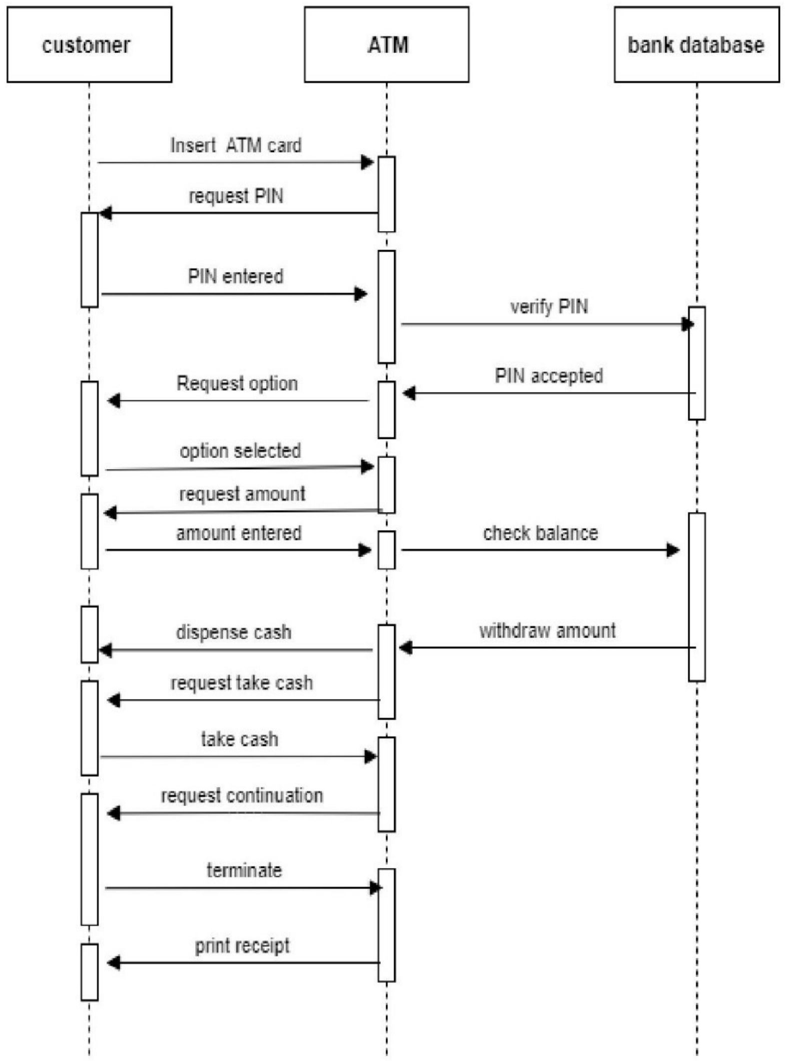
Regarding style issues for sequence diagramming, prefer drawing messages going from left-toright and return values from right-to-left, although that doesn’t always work with complex objects/classes. Justify the label on messages and return values, so they are closest to the arrowhead. Also prefer to layer the sequence diagrams: from left-to-right. indicate the actors, then the controller class(es), and then the user interface class(es), and, finally, the business class(es). During design, you probably need to add system and persistence classes, which you should usually put on the right-most side of sequence diagrams. Laying your sequence diagrams in this manner often makes them easier to read and also makes it easier to find layering logic problems, such as user interface classes directly accessing persistence .

Conclusion: The sequence diagram was made successfully by following the steps described above.

Another example of a sequence diagram



Sequence Diagram of ATM Management System:



Conclusion: The Sequence diagram of ATM Management System has been drawn successfully by following the steps described above.

EXERCISE NO. 9

Aim: Steps to draw the collaboration Diagram using Rational Rose.

Hardware Requirements:

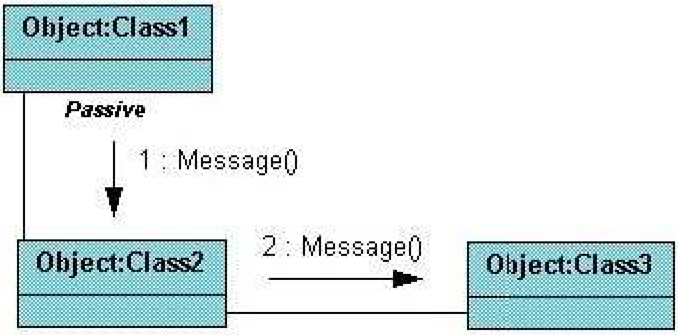
Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard and mouse, colored monitor.

Software Requirements:

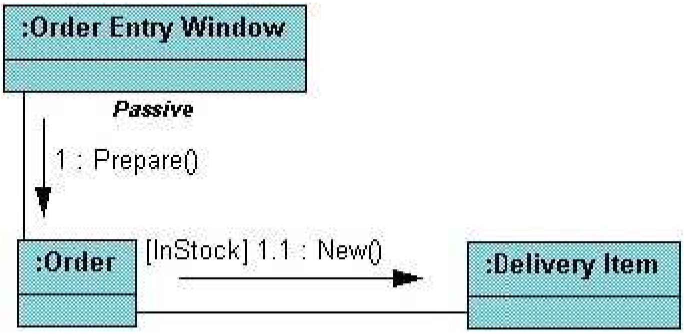
Rational Rose, Windows XP

THEORY:

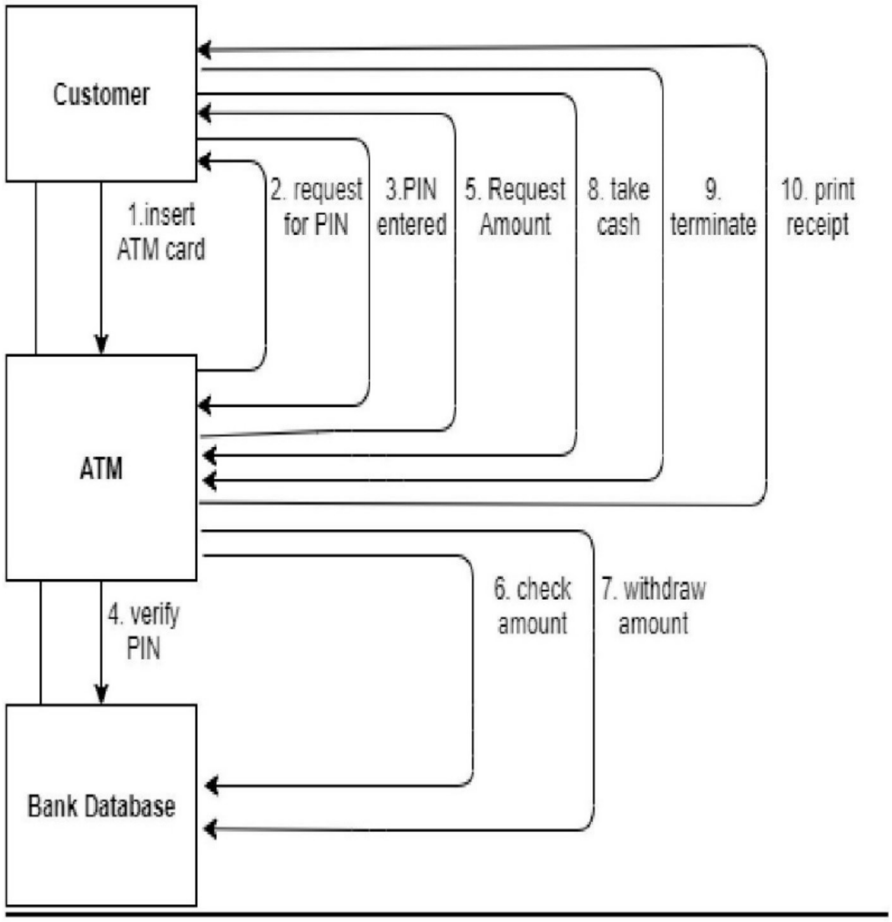
Collaboration diagrams are also relatively easy to draw. They show the relationship between objects and the order of messages passed between them. The objects are listed as icons and arrows indicate the messages being passed between them. The numbers next to the messages are called sequence numbers. As the name suggests, they show the sequence of the messages as they are passed between the objects. There are many acceptable sequence numbering schemes in UML. A simple 1, 2, 3... format can be used, as the example below shows, or for more detailed and complex diagrams a 1, 1.1 ,1.2, 1.2.1... scheme can be used.



The example below shows a simple collaboration diagram for the placing an order use case. This time the names of the objects appear after the colon, such as :Order Entry Window following the objectName:className naming convention. This time the class name is shown to demonstrate that all of objects of that class will behave the same way.


:Order Entry Window
Passive
1 : Prepare()
:Order
[InStock] 1.1 : New()
:Delivery Item

Collaboration Diagram of ATM Management System:



Conclusion: The Collaboration diagram of ATM Management System has been drawn successfully by following the steps described above.

EXERCISE NO. 10

Aim: To draw class diagram for any project.

REQUIREMENTS:

Hardware Interfaces

- Pentium(R) 4 CPU 2.26 GHz, 128 MB RAM
- Screen resolution of at least 800 x 600 required for proper and complete viewing of screens. Higher resolution would not be a problem.
- CD ROM Driver

Software Interfaces

- Any window-based operating system (Windows98/2000/XP/NT) ▪ IBM Rational Rose Software

THEORY:

A **class diagram** is a type of **static structure diagram** that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

Class diagrams show the classes of the system, their inter-relationships, and the operations and attributes of the classes. Class diagrams are typically used, although not all at once, to:

- Explore domain concepts in the form of a domain model
- Analyze requirements in the form of a conceptual/analysis model
- Depict the detailed design of object-oriented or object-based software

A class model is comprised of one or more class diagrams and the supporting specifications that describe model elements including classes, relationships between classes, and interfaces. There are guidelines

1. General issues
2. Classes
3. Interfaces
4. Relationships
5. Inheritance
6. Aggregation and Composition

GENERAL GUIDELINES

Because class diagrams are used for a variety of purposes – from understanding requirements to describing your detailed design – it is needed to apply a different style in each circumstance. This section describes style guidelines pertaining to different types of class diagrams.

CLASSES

A class in the software system is represented by a box with the name of the class written inside it. A compartment below the class name can show the class's *attributes* (i.e. its *properties*). Each *attribute* is shown with at least its name, and optionally with its type, initial value, and other properties.

A class is effectively a template from which objects are created (instantiated). Classes define attributes, information that is pertinent to their instances, and operations, functionality that the objects support. Classes will also realize interfaces (more on this later).

Class diagrams are widely used to describe the types of objects in a system and their relationships. Class diagrams model class structure and contents using design elements such as classes, packages and objects. Class diagrams describe three different perspectives when designing a system, conceptual, specification, and implementation. These perspectives become evident as the diagram is created and help solidify the design.

INTERFACES

An interface is a collection of operation signature and/or attribute definitions that ideally defines a cohesive set of behaviours. ***I***nterface a class or component must implement the operations and attributes defined by the interface. Any given class or component may implement zero or more interfaces and one or more classes or components can implement the same interface.

RELATIONSHIPS

A relationship is a general term covering the specific types of logical connections found on a class and object diagram.

Class diagrams also display relationships such as containment, inheritance, associations and others.

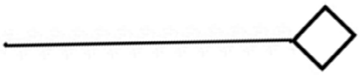
The association relationship is the most common relationship in a class diagram. The association shows the relationship between instances of classes.

Another common relationship in class diagrams is a generalization. A generalization is used when two classes are similar, but have some differences.

AGGREGATION

Aggregation is a variant of the "has a" or association relationship; composition is more specific than aggregation.

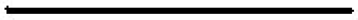
Aggregation occurs when a class is a collection or container of other classes, but where the contained classes do not have a strong ***life cycle dependency*** on the container--essentially, if the



container is destroyed, its contents are not.

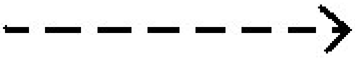
ASSOCIATION

Association are semantic connection between classes. When an association connects two classes, each class can send messages to other in a sequence or a collaboration diagram . Associations can be bidirectional or unidirectional.



DEPENDENCIES

Dependencies connect two classes. Dependencies are always unidirectional and show that one class, depends on the definitions in another class.



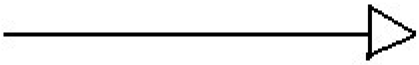
GENERALIZATION

The generalization relationship indicates that one of the two related classes (the *supertype*) is considered to be a more general form of the other (the *subtype*). In practice, this means that any instance of the subtype is also an instance of the supertype.

The generalization relationship is also known as the *inheritance* or *"is a"* relationship.

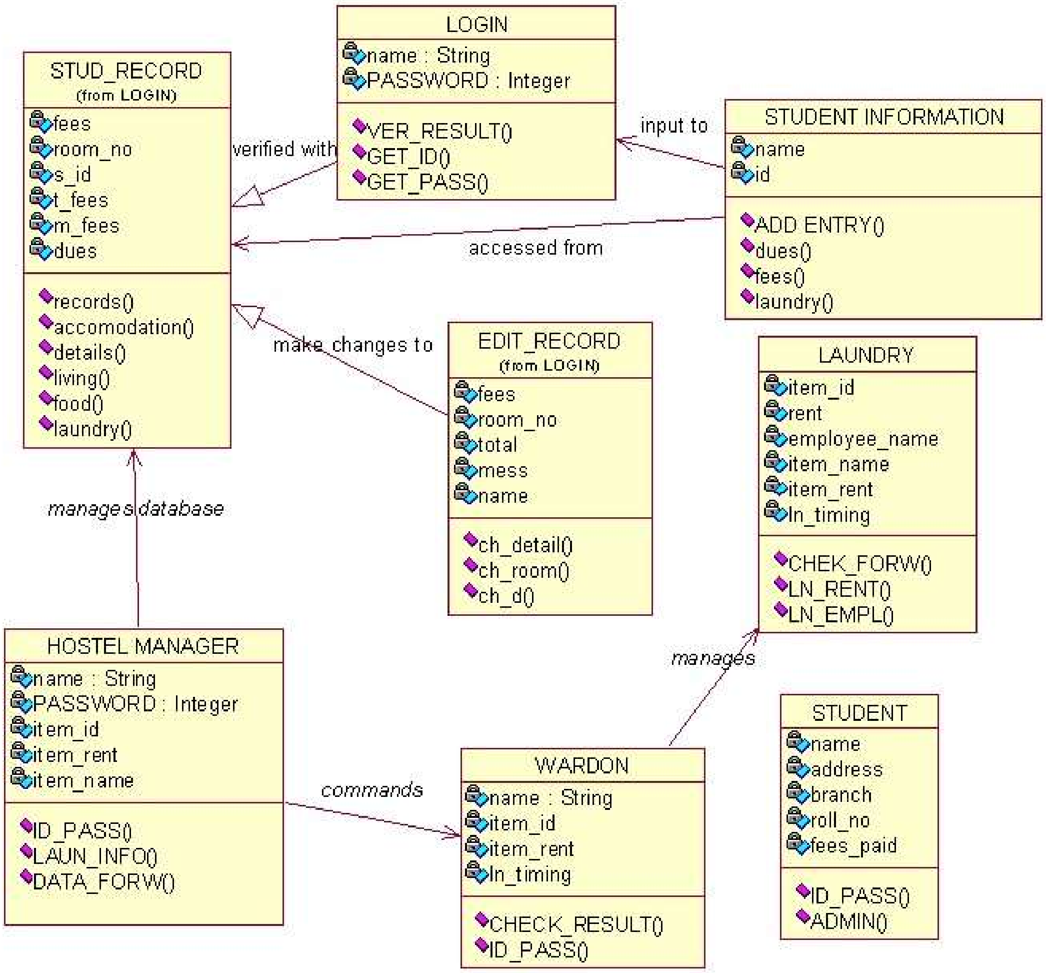
The *supertype* in the generalization relationship is also known as the *"parent"*, *superclass*, *base class*, or *base type*.

The subtype in the generalization relationship is also known as the *"child"*, *subclass*, *derived class*, *derived type*, *inheriting class*, or *inheriting type*.

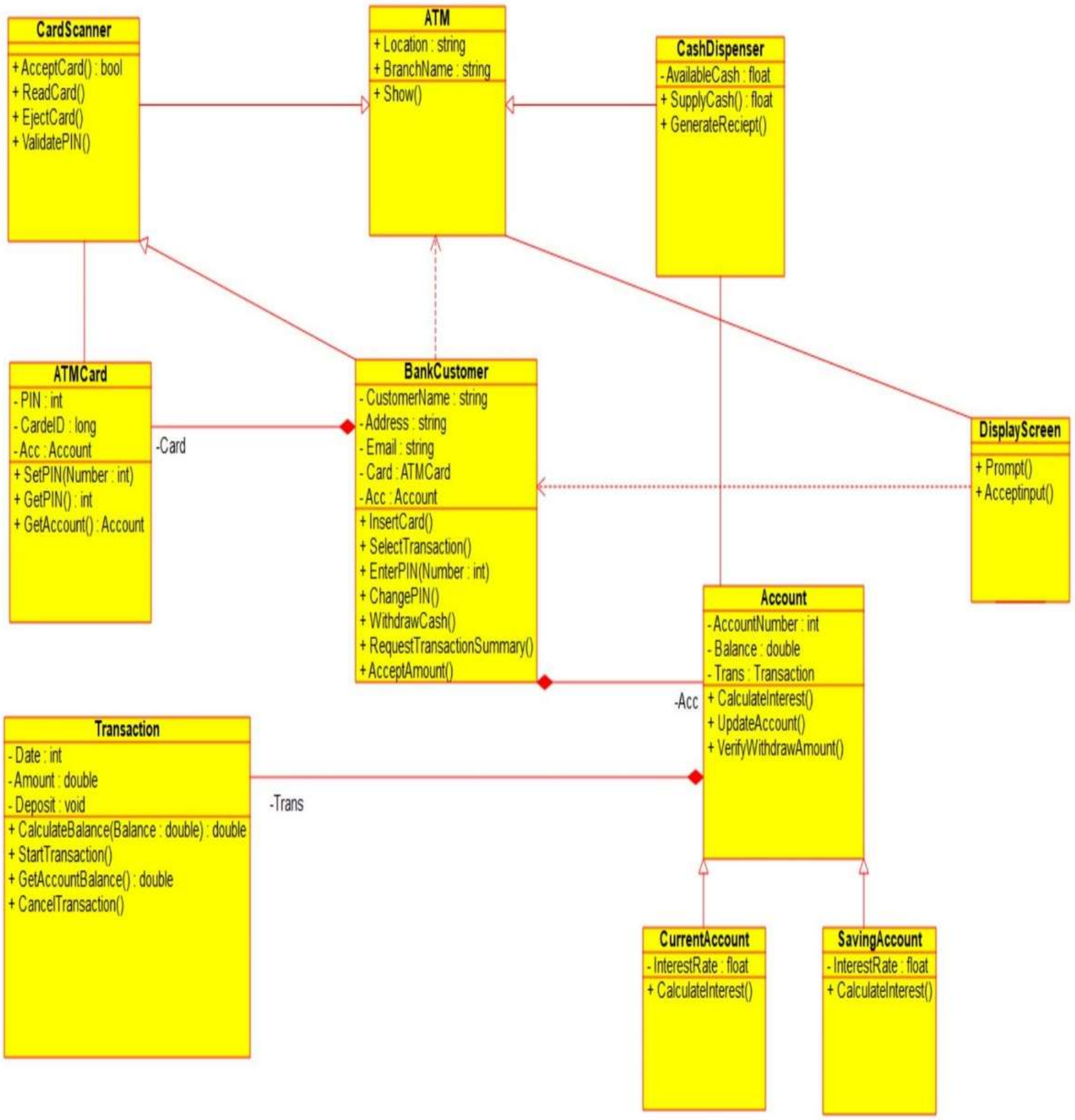


MULTIPLICITY

The association relationship indicates that (at least) one of the two related classes makes reference to the other.


LOGIN
name : String
PASSWORD : Integer
VER_RESULT()
GET_ID()
GET_PASS()
STUD_RECORD
(from LOGIN)
fees
room_no
s_id
t_fees
m_fees
dues
records()
accomodation()
details()
living()
food()
laundry()
verified with
input to
STUDENT INFORMATION
name
id
ADD ENTRY()
dues()
fees()
laundry()
accessed from
make changes to
EDIT_RECORD
(from LOGIN)
fees
room_no
total
mess
name
ch_detail()
ch_room()
ch_d()
LAUNDRY
item_id
rent
employee_name
item_name
item_rent
In_timing
CHEK_FORW()
LN_RENT()
LN_EMPL()
manages database
manages
HOSTEL MANAGER
name : String
PASSWORD : Integer
item_id
item_rent
item_name
ID_PASS()
LAUN_INFO()
DATA_FORW()
commands
WARDON
name : String
item_id
item_rent
In_timing
CHECK_RESULT()
ID_PASS()
STUDENT
name
address
branch
roll_no
fees_paid
ID_PASS()
ADMIN()

Class Diagram of ATM Management System:



Conclusion: The Class diagram of ATM Management System has been drawn successfully by following the steps described above.