

Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming

-The ERD (Entity-Relationship Diagram) is presented to define and clarify entities, attributes, and their relationships, and constraints applied to them.

-DFD (Data Flow Diagram) that will be discussed in the next chapter depicts the system components—processes, external entities that interact with the system, and information flow in the system.

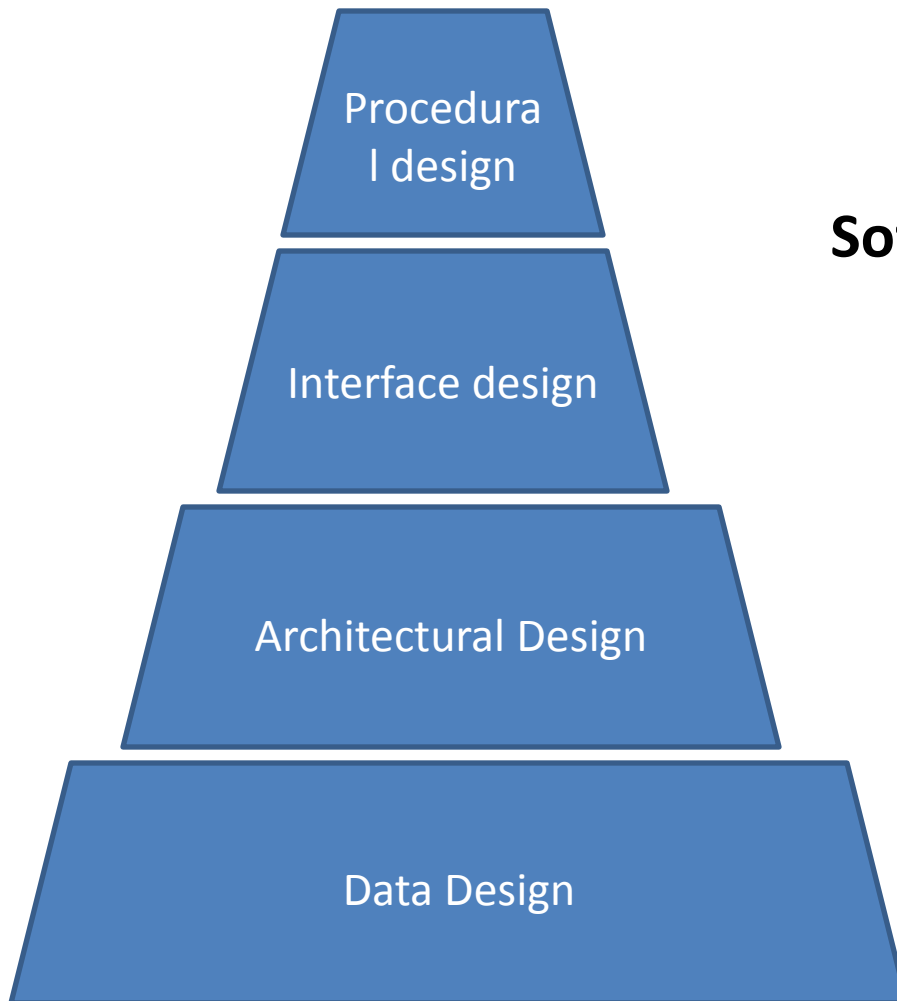
Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming

- After you understand the problem clearly, it is the time to solve it.
- Software design defines a solution to the problem.
- Structured design tasks are performed to translate what produced in system analysis to blueprint of the system that can be implemented.
- During structured design, the four design models are presented.

Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming



Software Design Model

Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming

- After you understand the problem clearly, it is the time to solve it.
- Software design defines a solution to the problem.
- Structured design tasks are performed to translate what produced in system analysis to blueprint of the system that can be implemented.
- During structured design, the four design models are presented—data design, architectural design, interface design, and procedural or component level design.

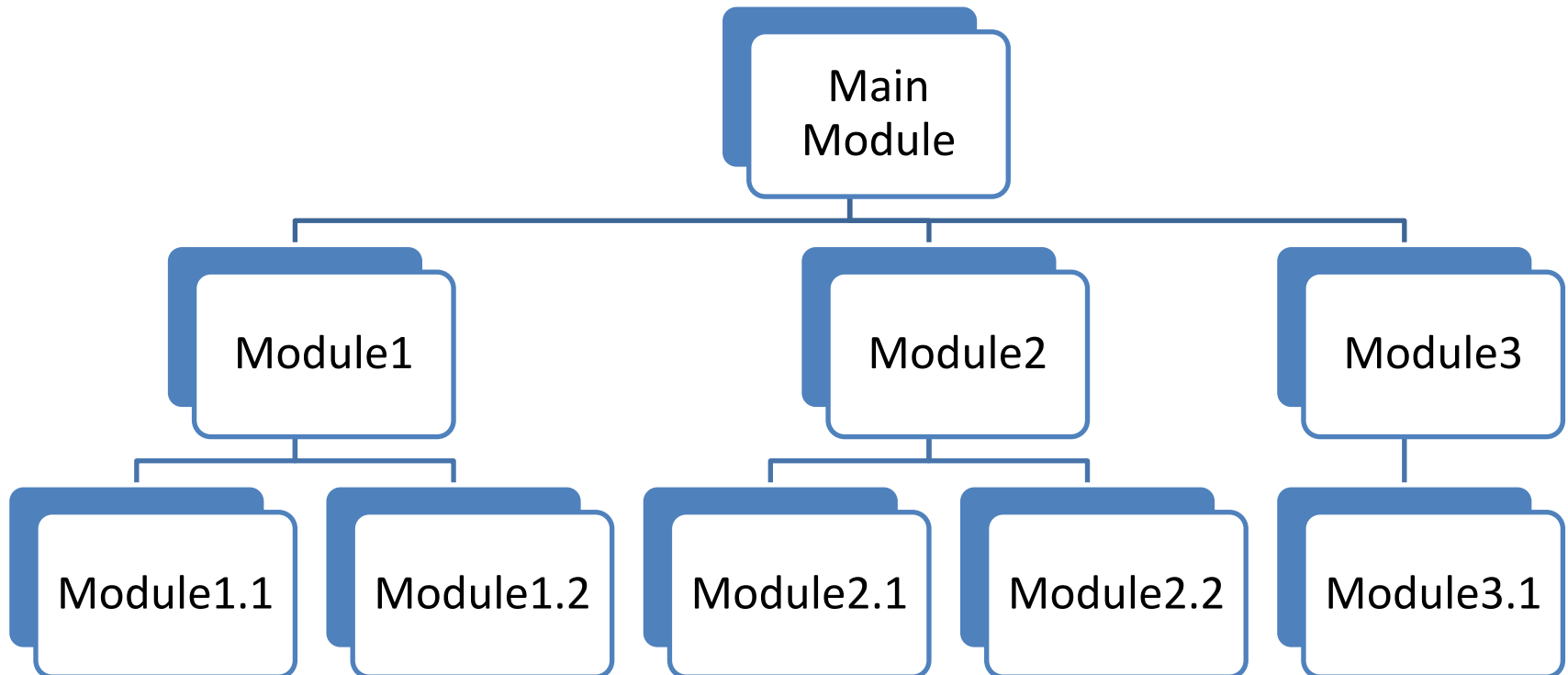
Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming

- Data design** transforms ERD in to data structures required to implement the software.
- Entities, attributes, and relationships are the basic for design activities.
- Architectural design** define a higher-level structure of the system. The higher-level structure of the system depicts the relationship between the structural components or modules of the system.
- Structure Chart is a good visual presentation of Architectural design.

Chapter IV: System Development Methods

1. Structured Analysis, Design, and programming



Structure chart example

Chapter IV: System Development Methods

1. Structured Analysis, Design, and Programming

- The Interface design** (discussed later in other chapter) defines interaction of people with the system, and the system with other systems that work with it . It provides the best way for people and other systems to interact with the system.
- Data Flow Model can be a good input for interface design.
- Procedural design** transforms the structural components or modules of the system architecture in to procedural description of the components.

Chapter IV: System Development Methods

1. Structured Analysis, Design, and Programming

-Pseudo-code or Flow Chart is a popular tool for procedural design.

-Pseudo-code example to calculate a parking fee of a parking lot:

+1 input vehicle type, time in, and time out

+2 set parking rates :

-For car and $\text{duration} \leq 2$, $\text{rate} = 0\$$

-For car and $\text{duration} > 2$, $\text{rate} = 1\$$ per hour

-For truck and $\text{duration} \leq 2$, $\text{rate} = 1\$$

-For truck and $\text{duration} > 2$, $\text{rate} = 2\$$ per hour

-For bus and $\text{duration} \leq 2$, $\text{rate} = 2\$$

Chapter IV: System Development Methods

1. Structured Analysis, Design, and Programming

-Pseudo-code or Flow Chart is a popular tool for procedural design.

-Pseudo-code example to calculate a parking fee of a parking lot:

+3 Compute parking fee:

-If $v_type=c$ & $duration \leq 1$ then $p_fee=crate1$

-If $v_type=c$ & $duration > 1$ then $p_fee=0+(duration-1)*crate2$

-If $v_type=t$ & $duration \leq 2$ then $p_fee=trate1$

-If $v_type=t$ & $duration > 2$ then $p_fee=1+(duration-2)*trate2$

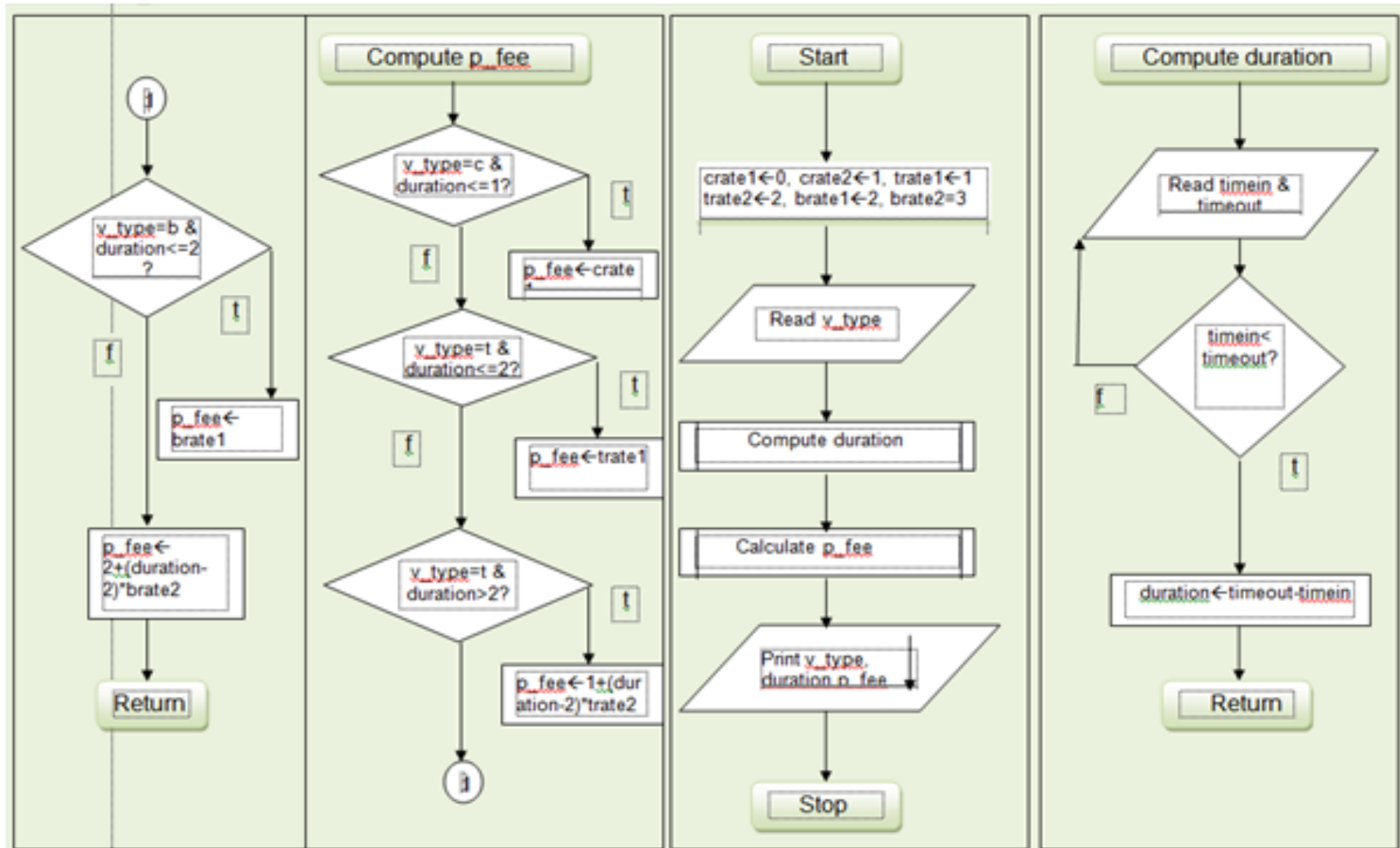
-If $v_type=b$ & $duration \leq 2$ then $p_fee=brate1$

-If $v_type=b$ & $duration > 2$ then $p_fee=2+(duration-2)*brate2$

-Return p_fee

System Analysis & Design

-Flow Chart example of parking system:



Chapter IV: System Development Methods

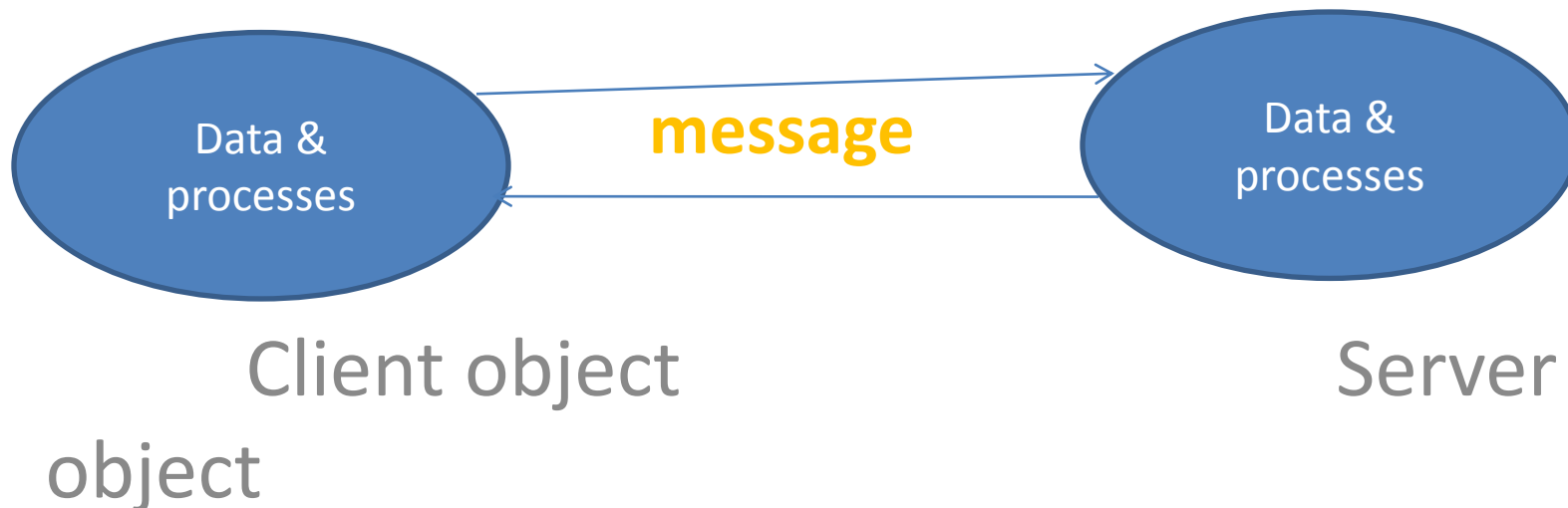
1. Structured Analysis, Design, and Programming

- When program modules are designed, coding begins.
- Structured programming use standard control structures to improve clarity, and maintenance.
- Structured programming encourages top-down development. A top-level function is expanded down in to more detail components.
- Structured programming use SEQUENCE, IF ELSE, and LOOP to facilitate top-down development.
- Structured programming languages initially used are ALGOL, Pascal, PL/I, and Ada. C is the most popular structured programming language today.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

- Object-Oriented analysis and design do not model a system by E-R diagram, and DFD diagram. These components are integrated in to objects. This can be inferred that an object contains data and processes/methods.
- Objects interact with each other through message to reach the common goal of a large system



Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

- Object modeling offers a number of advantages for the system development.
- +Object modeling leads to an easy conversion of models (developed in analysis and design) to an implementation.
- +Objects can be reusable. They are easy to plug in the system.

Chapter IV: System Development Methods

1. Object-Oriented Analysis, Design, and Programming

Object structure

Objects contain data and processes that operate on the data.

Project
-ProNumber = p0012 -AssPers : Persons ref -StartDate = 09-02-2013 -Manager = Mr. A -Tk : Tasks -Budget = 20000
+AddPer() +DeletePer() +SetTask() +ChangeBud()

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

For example, the PROJECT object contains the following data and methods:

Data:

- ProNumber: p0012
- AssPers: Persons ref
- Manager: Mr.A
- StartDate:09-02-2013
- Budget: 12000
- Tk: Tasks

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

For example, the PROJECT object contains the following data and methods:

Methods:

- AddPer()

- DeletePer()

- SetTask()

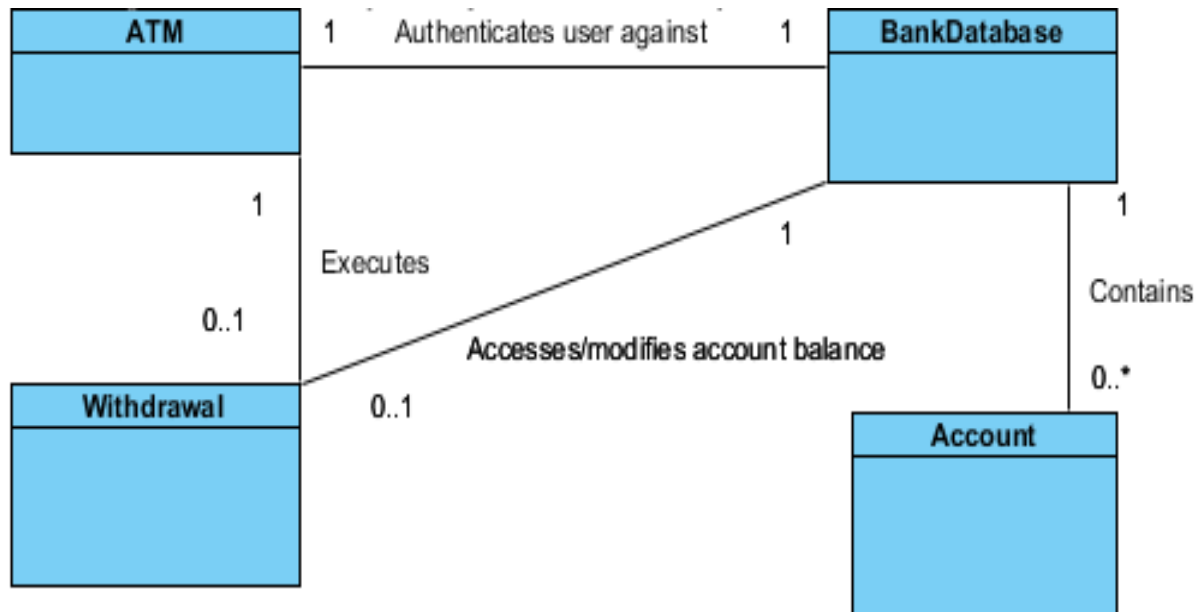
- ChangeBud()

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Objects Association and Containment

An object model usually contains many objects related to each other.



Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

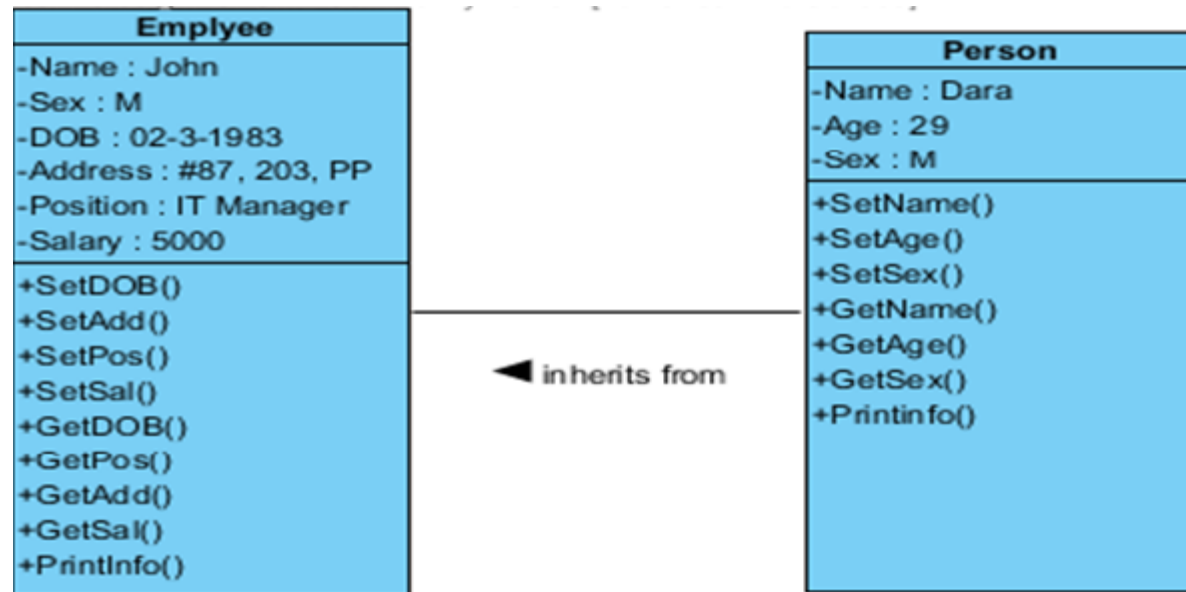
- The ATM object has one-to-one relationship with the BankDatabase object.
- The ATM participates a composition relationship with a zero or one of object Withdrawal.
- The BackDatabase participates a composition relationship with zero or one of the Withdrawal object.
- The BankDatabase contains zero or more Account.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Objects Inheritance

An object can inheritance features from another object. I can have additional features or override features of the object.



Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

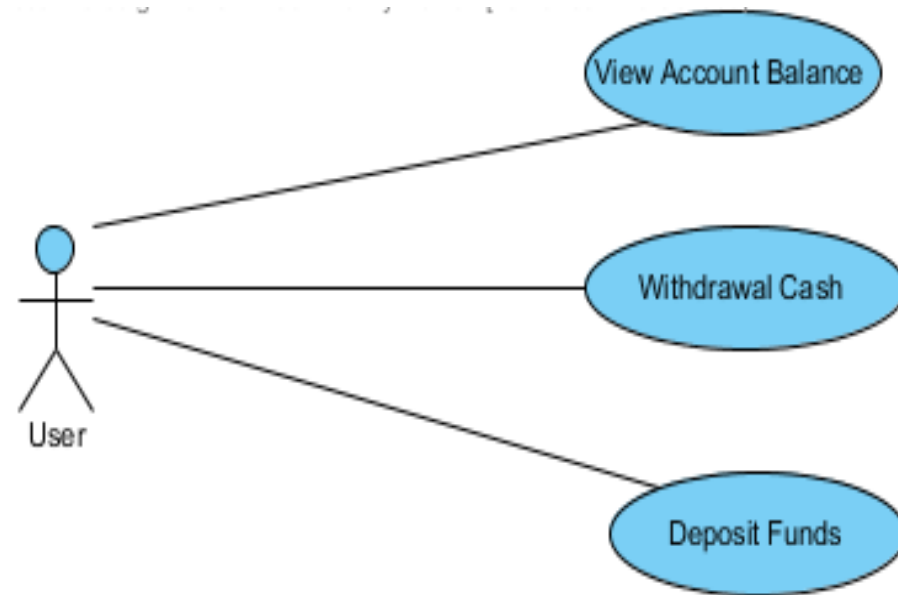
- Employee object inherits SetName(), SetSex(), GetName(), and GetSex() methods from Person object.
- Employee object overrides the Printinfo() method of the Person object to include additional information (DOB, Address, Position, and Salary) to print out.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Analysis

- The most common diagram to be produced during system analysis using UML object-oriented method is USE CASE.
- USE CASE presents the activities (scenarios) of the users (actors) with the system.



System Analysis & Design

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Analysis

- The picture above is an example of USE CASE. This USE CASE shows the interactions of the user (actor) with the ATM system.
- The activities of the user with the system can be account balance view, cash withdrawal, and funds deposit.

-In USE Case the human image  represents the actor.

-Each use case is represented by val symbol.

-The actors and activities in USE CASE diagram are collected from requirements document.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Design

a). Defining classes

- In the object-oriented method, the design tasks involve defining system structure/architecture and system behavior.
- System structure/architecture defines the system components that make up the system and their relationships.
- There are number of UML models that relate to the system structure. The most popular one covered in this tutorial is **Class Diagram**.
- Class Diagram specifies the structural components and their relationships

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Design

- The structural components of the system are classes.
- You can define the classes by defining the things (noun) from the requirements document. For example, things or nouns that can be collected from the ATM requirements document are *ATM, account number, PIN code, transactions, balance enquiry, withdrawal, deposit, customer, user, bank database, screen, keypad, case dispenser, fund or money, deposit slot.*
- You also need to distinguish nouns that can be classes and attributes.
- A class in broader sense can be break down in to more specific classes. For example, transaction can be break in to balance enquiry, withdrawal, and deposit.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Design

- The nouns can be classes are ATM, Account, Balance Enquiry, Withdrawal, Deposit, Bank Database, Screen, Keypad, Cash Dispenser, Deposit slot.
- The nouns can be attributes are account number, pin code, amount/fund, balance.

Chapter IV: System Development Methods

2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Design

- After classes are defined, their relationships are specified.
 - +Screen, Keypad, Case Dispenser, Deposit slot are in ATM class.
 - +ATM authenticates a user against the Bank Database.
 - +ATM executes account enquiry, withdrawal, or, deposit.
 - +Withdrawal modifies account balance through Bank Database.
 - +Back Database contains accounts.

Chapter IV: System Development Methods

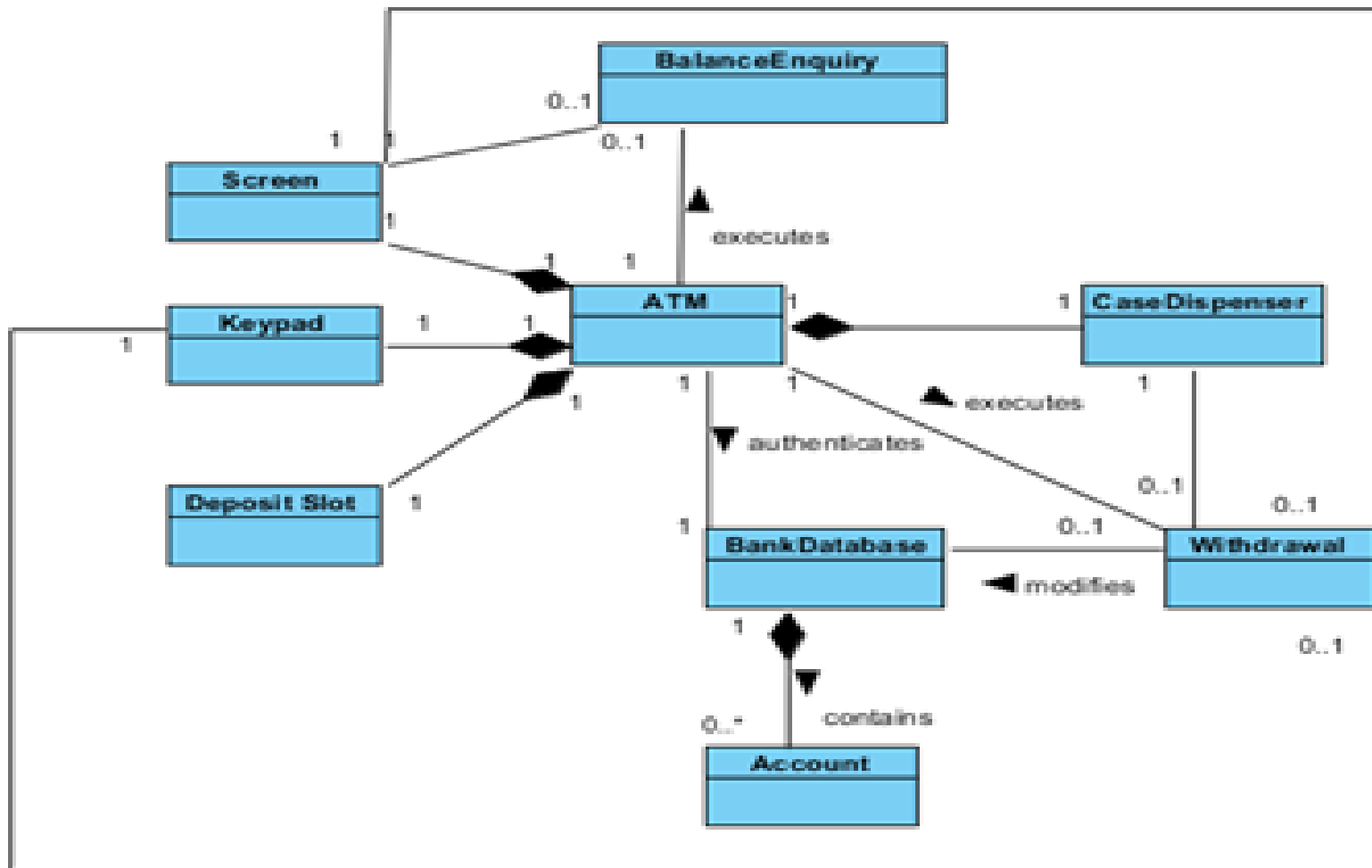
2. Object-Oriented Analysis, Design, and Programming

Object-Oriented Design

- +Balance enquiry requires screen to display the account balance.
- +Withdrawal requires screen, and keypad to prompt for information and receive input, and cash dispenser to dispense cash to the user.
- +Deposit also requires screen, and keypad to prompt for information and receive input, and deposit slot to receive envelope.

Chapter IV: System Development Methods

ATM system: Class Diagram



Chapter IV: System Development Methods

The Deposit class is not included in the Class Diagram because we want to keep the diagram simple and the Deposit class is much like the Withdrawal.

b.) Defining attributes

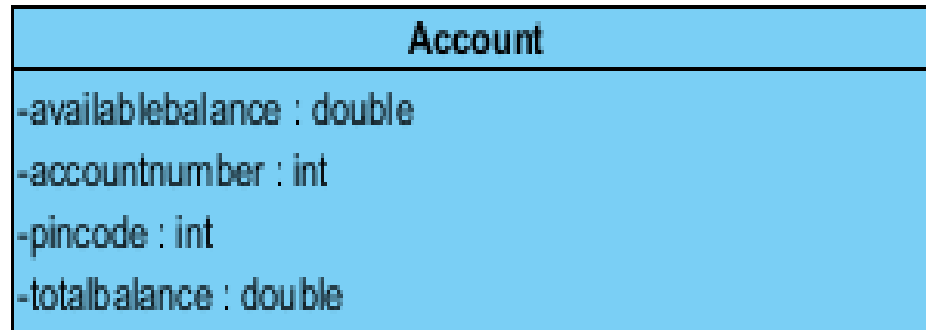
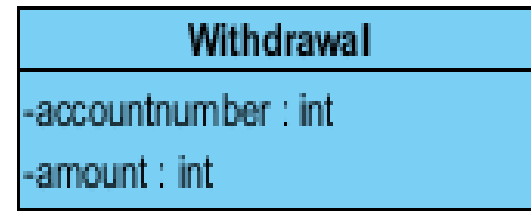
Attributes describe the class. Each attribute can have a value. Attributes of class are implemented as data members in C++, C#, or Java. The table below shows the classes and their attributes.

Chapter IV: System Development Methods

Class	Attributes
ATM	Authenticated
BalanceEnquiry	Account number
Withdrawal	Account number, amount
Deposit	Account number, amount
BankDatabase	
Account	Account number, PIN code, Available balance, total balance
Screen	
Keypad	
CashDispenser	Max number of bills/per day
DepositSlot	

Chapter IV: System Development Methods

Then you will model attributes of classes as shown below:



Chapter IV: System Development Methods

The Deposit class is not included in the Class Diagram because we want to keep the diagram simple and the Deposit class is much like the Withdrawal.

c.) Defining class operations

Classes/Objects perform operations. For example, ATM object executes balance enquiry, withdrawal, or deposit. You can identify the operations by collecting the verbs or verb phrases from the requirements document.

System Analysis & Design

The table below shows operation that can be identified from the requirements document.

Class	Operations
ATM	authenticate user Executes transactions
BalanceEnquiry	NA
Withdrawal	NA
Deposit	NA
BankDatabase	Retrieve account balance, debit amount from account, , credit amount to account
Account	NA
Screen	Display messages
Keypad	Get user input
CashDispenser	
DepositSlot	

Chapter IV: System Development Methods

Then you will model operations of classes as shown below:

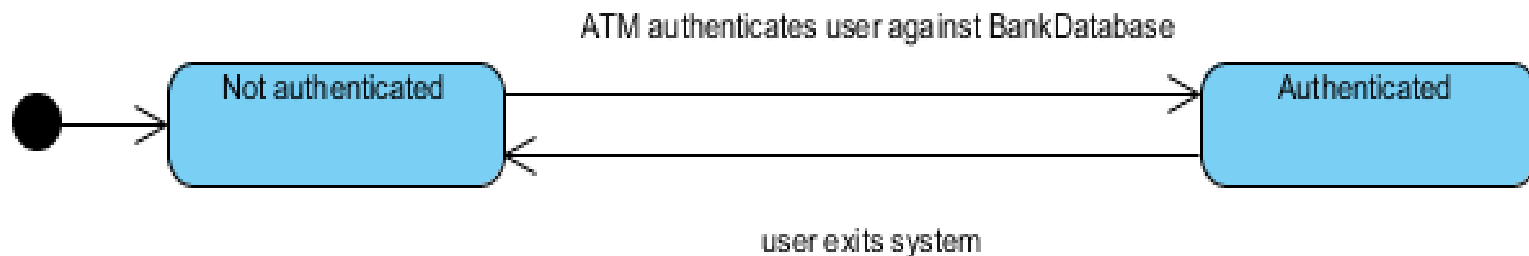
Account
-availablebalance : double -accountnumber : int -pincode : int -totalbalance : double
+validatePin(PinCode : int) : boolean +getAvaiBal(AccNumber : int, PinCode : int) : double +credit(AccNum : int, PinCode : int, Amount : double) : void +getTotBal(AccNumber : int, PinCode : int) : double +debit(AccNum : int, PinCode : int, Amount : double) : void

Withdrawal
-accountnumber : int -amount : int
+executeWithdrawal(AccNumbe...

Chapter IV: System Development Methods

c). Defining system behavior

-State Machine Diagram describes the state changes of some components of the system to response to the events occurring in the system. For example, the ATM object may change from *not authenticated* (initial state) to *authenticated* when the user enters valid account number, and PIN code. When the user exits the system, the stage changes from authenticated to not authenticated.

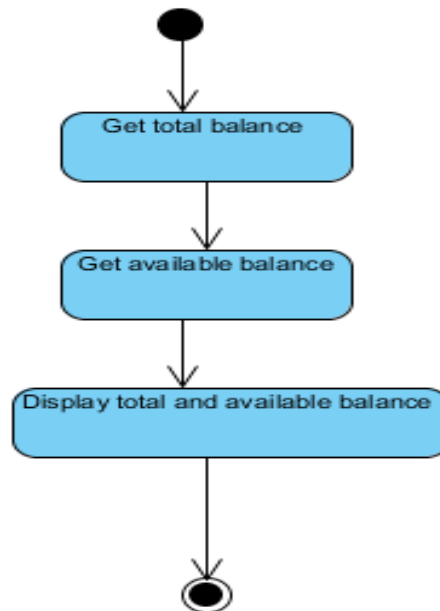


Chapter IV: System Development Methods

d). Defining system behavior

-Activities Diagram describes the activities (and their order) carried out by objects in the system.

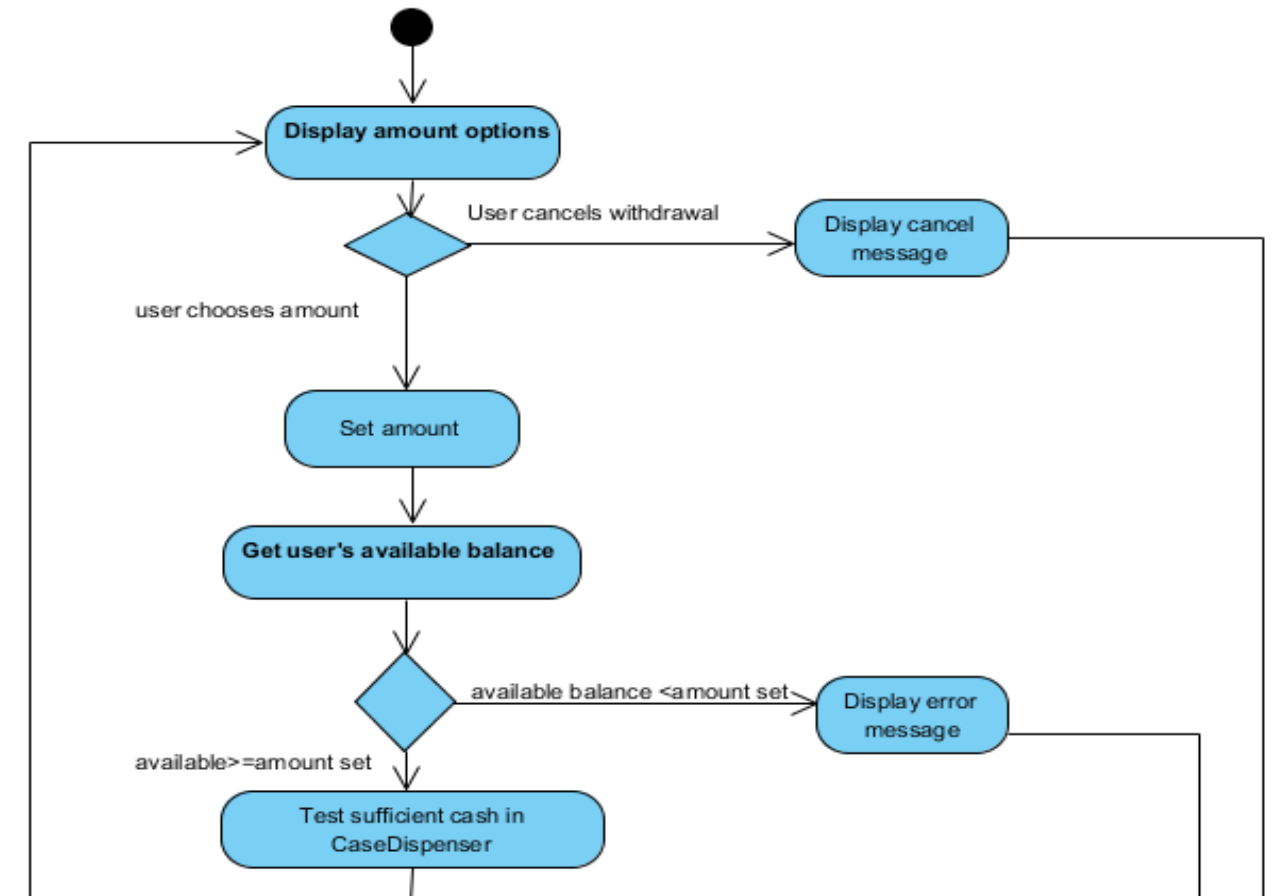
Activity Diagram for Balance Enquiry



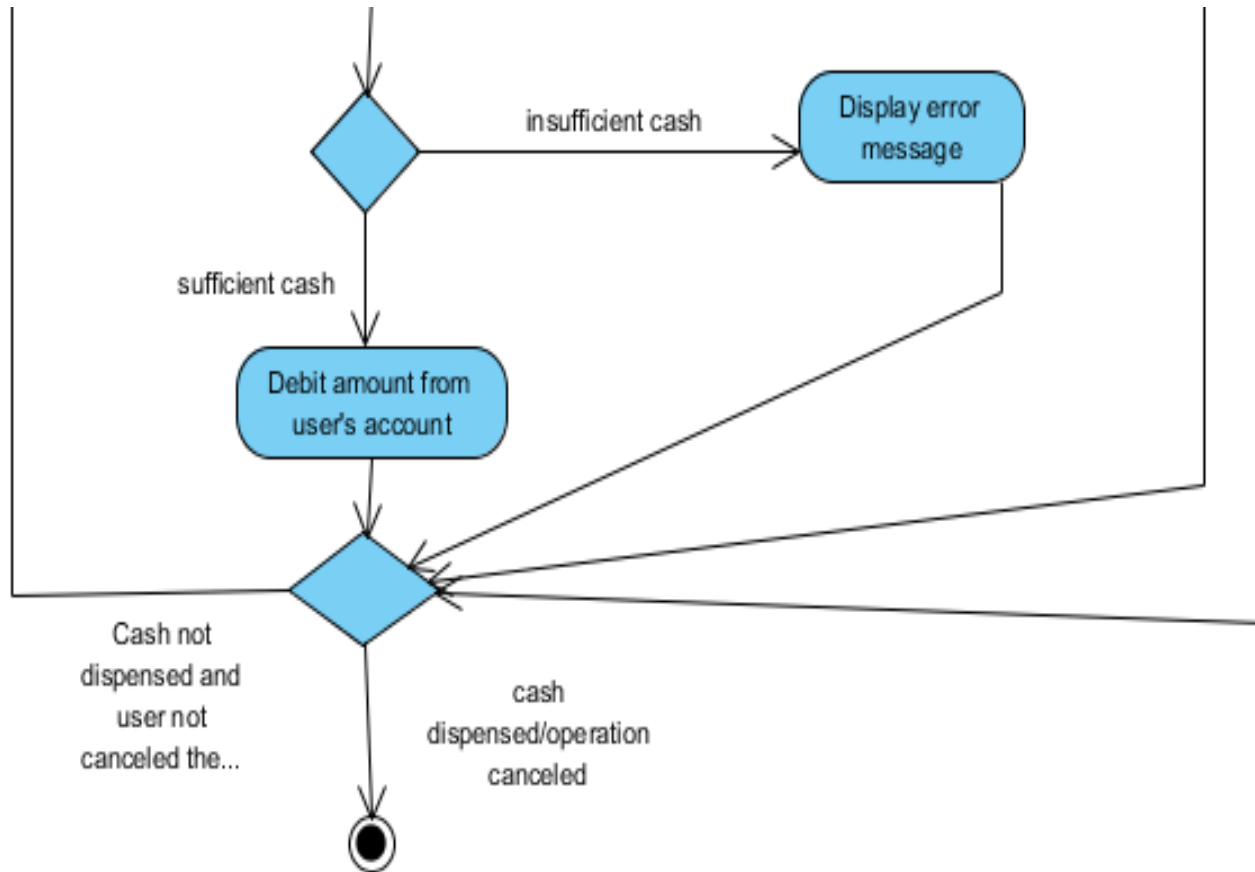
Chapter IV: System Development Methods

d). Defining system behavior

Activity Diagram for Withdrawal



Chapter IV: System Development Methods



Chapter IV: System Development Methods

d). Defining system behavior

-Communication Diagram models the communications between objects in the system. An object communicates or interacts with another through messages. Messages in this context are the same as function calls in C++, or method invokes in C# or Java. The order or sequence in which the messages are sent to each other also specified in the diagram in numbers.

System Analysis & Design

Objects communication table (ATM)

Object sends	Messages	To object
ATM	displayMessage getInput authenticateUser executeBalanceEnquiry executeWithdrawal executeDeposit	Screen KeyPad BankDatabase BalanceEnquiry Withdrawal Deposit
BalanceEnquiry	getTotalBanace getAvailableBalance displayMessage	BankDatabase BankDatabase Screen
Withdrawal	displayMessage getInput getAvailableBalance checkSufficient debitAccount dispenseCash	Screen KeyPad BankDatabase CaseDesipenser BankDatabase CashDispenser

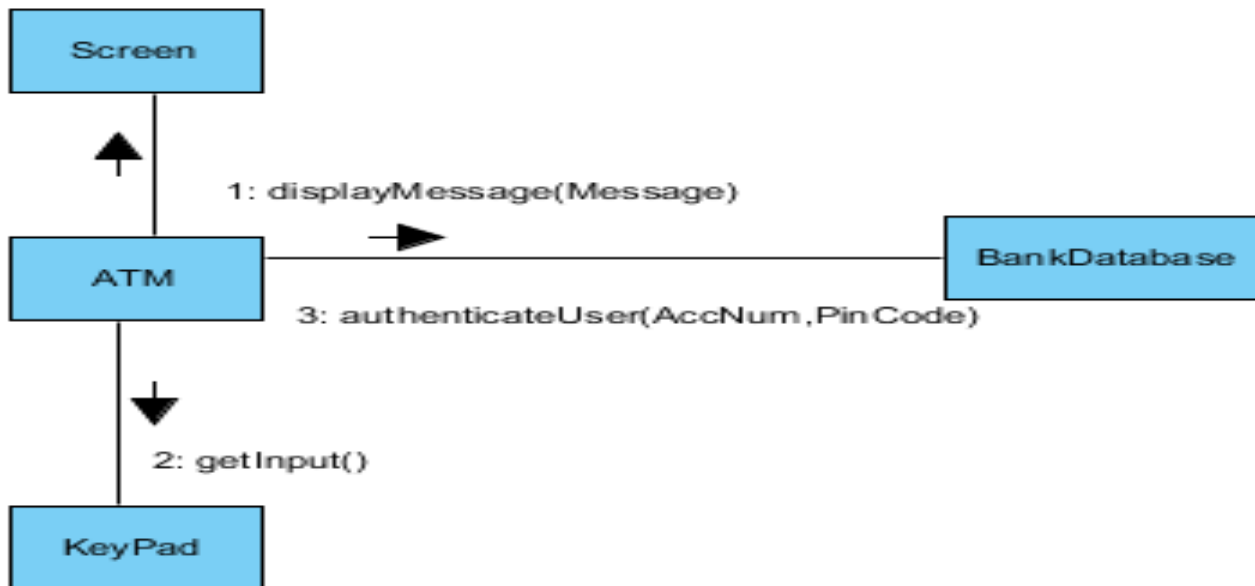
System Analysis & Design

Objects communication table (ATM)

Object sends	Messages	To object
Deposit	displayMessage getInput checkEnvelop crediAccount	Screen KeyPad DepositSlot BankDatabase
BankDatabase	validatePin getTotalBalance getAvailbleBalance creditAccount debitAccount	Account Account Account Account Account

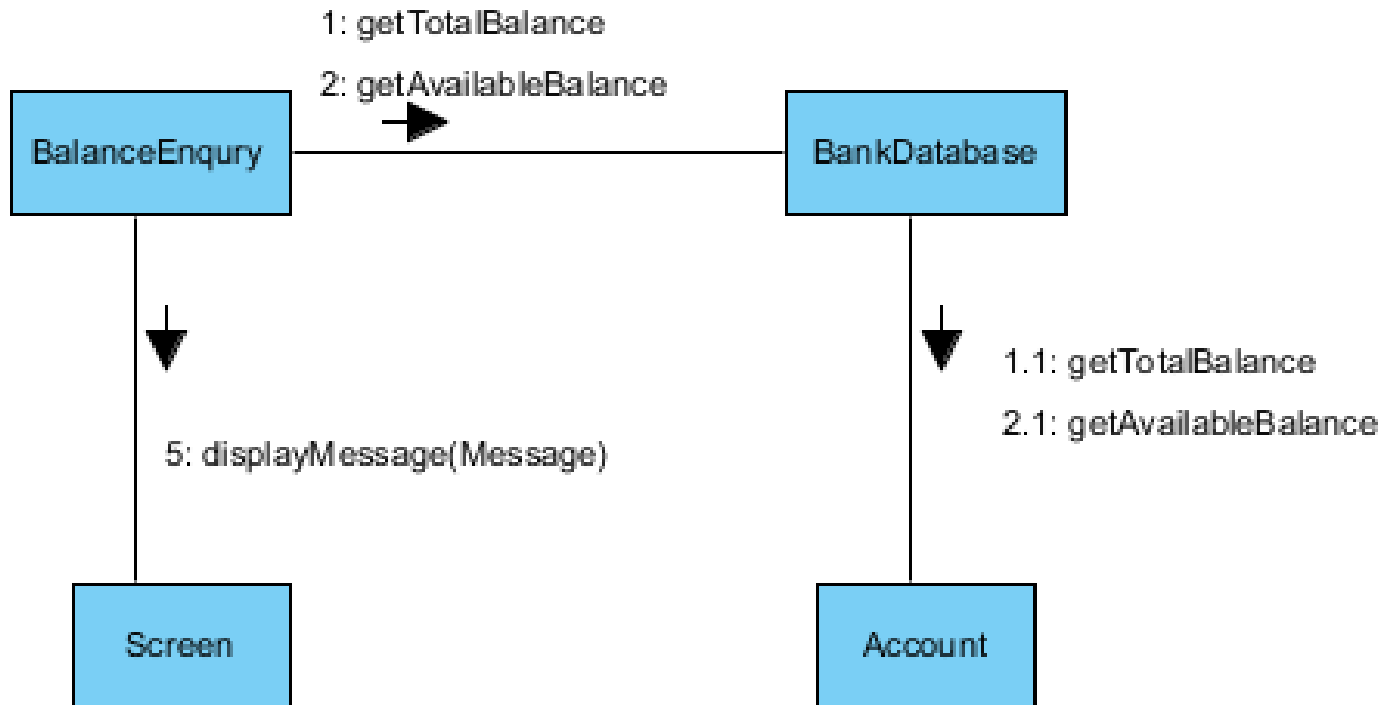
System Analysis & Design

Communication Diagram: communications between ATM, Screen, KeyPad, And BankDataBase objects.



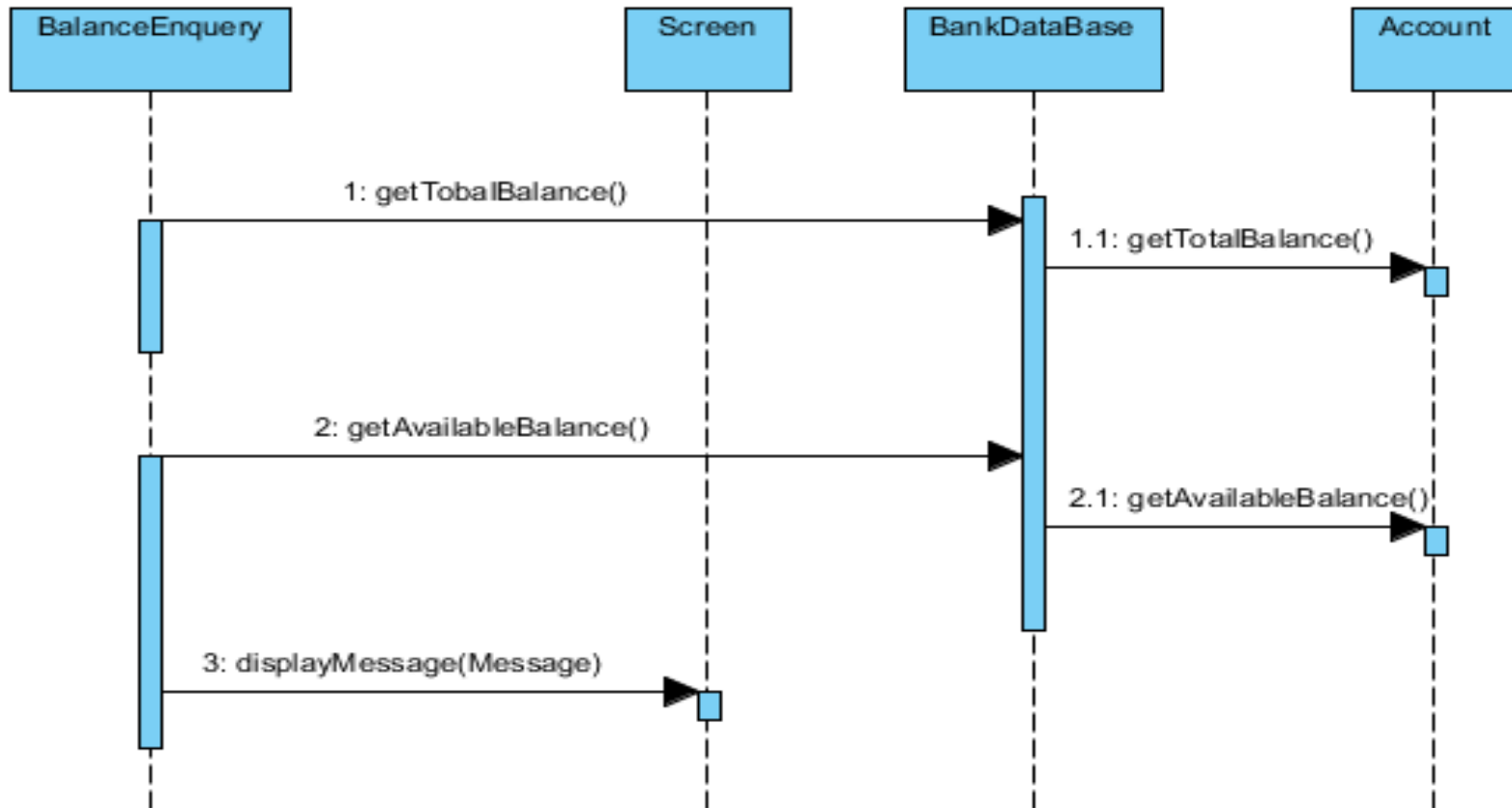
System Analysis & Design

Communication Diagram: communications between BalanceEnquiry, BankDataBase, and Screen objects to complete balance enquiry transaction.



System Analysis & Design

Sequence Diagram is another model to specify the communication between objects in the system. The Sequence Diagram focuses on the timing message flow rather than on the participants of the model.



System Analysis & Design

- A rectangle represents an object (e.g BalanceEnquiry).
- A dot line drawn from the object is its lifeline that represents the time process.
- A solid arrow filled with arrow head extending from the sender's activation to receiver 's activation is the message between the two objects in communication.
- An activation(thin vertical rectangle) happens a long the time progress of the object to indicate that the object is executing.

Object-Oriented Programming

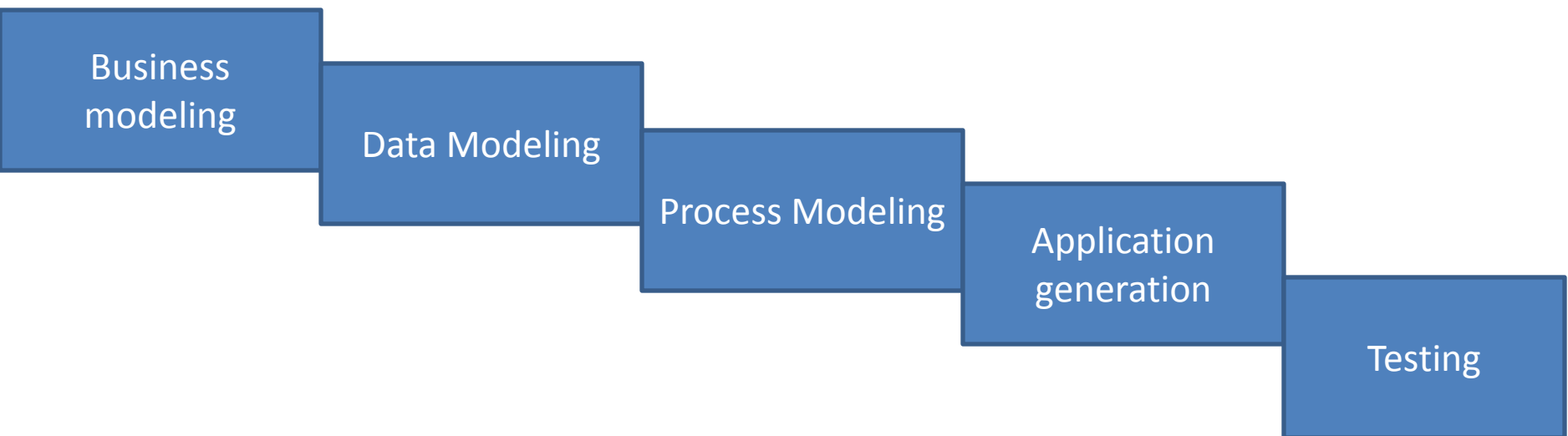
In Object-Oriented Design tasks, classes, objects, data, processes, object states, and objects communication are modeled. Those models can be directly implemented by a programming language that supports object-oriented programming method. This kind of programming language is called object-oriented programming. C++, Java, C#, Ruby, and Smalltalk are examples of popular object-oriented programming language today.

4. RAD Application Development

-RAD is system development model that focuses on the short development cycle (60-90 days) by using component-based construction.

-RAD defines the system development phases as shown below:

+Business modeling: In this phase, the information about business functions are modeled to answers the following questions: What information are input in the business process? What information are generated? Where does the information go? Who processes it?



4. RAD Application Development

+Data modeling: This phase redefines the information generated in business modeling to create E-R diagram that is able to answer the following questions: What are the primary objects to be processed by the system? What are attributes describe each object? What are the relationships between the objects?

+Process modeling: The process model describes the information flow in the system. The data flow diagram (DFD) is graphical representation of the information flow and the transforms that are applied as data move from input to output.

4. RAD Application Development

+Application generation: RAD uses the fourth generation techniques (4GT) to construct the software. The 4GT tools enable software engineers to input the software specification at a high level. Then they automatically generate the source code based on the specification. The 4GT method uses automated tools that are able to use and create reusable objects or components.

+Testing: The 4GT method widely supports the use and creation of reusable components. Existing components were already tested, so the time required to test the software is reduced. However, new components must be tested and all interfaces must be exercised thoroughly.

3. Joint Application Development (JAD)

-JAD is team-oriented approach used in early stages (analysis & design) system development process.

-Users and developers jointly define and clarify requirements and develop the solution.

-JAD team should include the following people:

+*Preventatives* from customer's organization who are responsive for communicating customer support and cooperation, setting project vision, identifying functional area to be addressed by the system, and resolving any conflicts that occur.

+*Well-trained facilitators* who are capable of using tools and techniques to collect requirements. He/she is responsive for organizing and scheduling JAD activities, guiding the sessions, and encourage participation.

3. Joint Application Development (JAD)

+Users from the organization to provide business expertise, direction of the business, and to present the various groups in the organization affected by the project.

+Developer representatives are responsible for transforming business requirements to models, building prototype, ensuring the solution to be delivered at the right time and in the limited budget, and proposing the available technology that is suitable the project.

3. Joint Application Development (JAD)

+Recorder is responsible for recording and distributing the proceedings of the meeting or session.

+Observers are members of software development team. They sit behind the participants and observe the proceedings of the meeting.

System Analysis & Design

3. Joint Application Development (JAD)

-JAD software development has the following phases:

+Planning: in the planning stage, the representatives from organization and facilitator are designated and system needs are defined. The system needs can be defined by answering the following questions: What are the benefits? What problem would it solve? Does it fit the strategic direction of the organization? What are risks associated with the system?

+Preparation: the tasks in preparation for design meetings or sessions involves scheduling design meetings, prepare meeting agendas, training participants, and prepare materials, room, and software aids.

3. Joint Application Development (JAD)

-JAD software development has the following phases:

+*Design meetings/sessions*: to complete this phase, the following tasks have to be done: review project scope (data, functions, performance, and constraints) and system needs, identify system interfaces, and develop a prototype.

+*Finalization*: the final state is to ensure that the design document is complete and signed off, the prototype is demonstrated and got approval from organization's representative.

3. Joint Application Development (JAD)

- Using JAD in system development has the following benefits:
 - +Improve quality of requirements
 - +Promote teamwork with customer
 - +Avoid deficiencies in system analysis and design
 - +Accelerate system design