SOFTWARE DESIGN & ANALYSIS (Week-5)

USAMA MUSHARAF

MS-CS (Software Engineering)

LECTURER (Department of Computer Science)

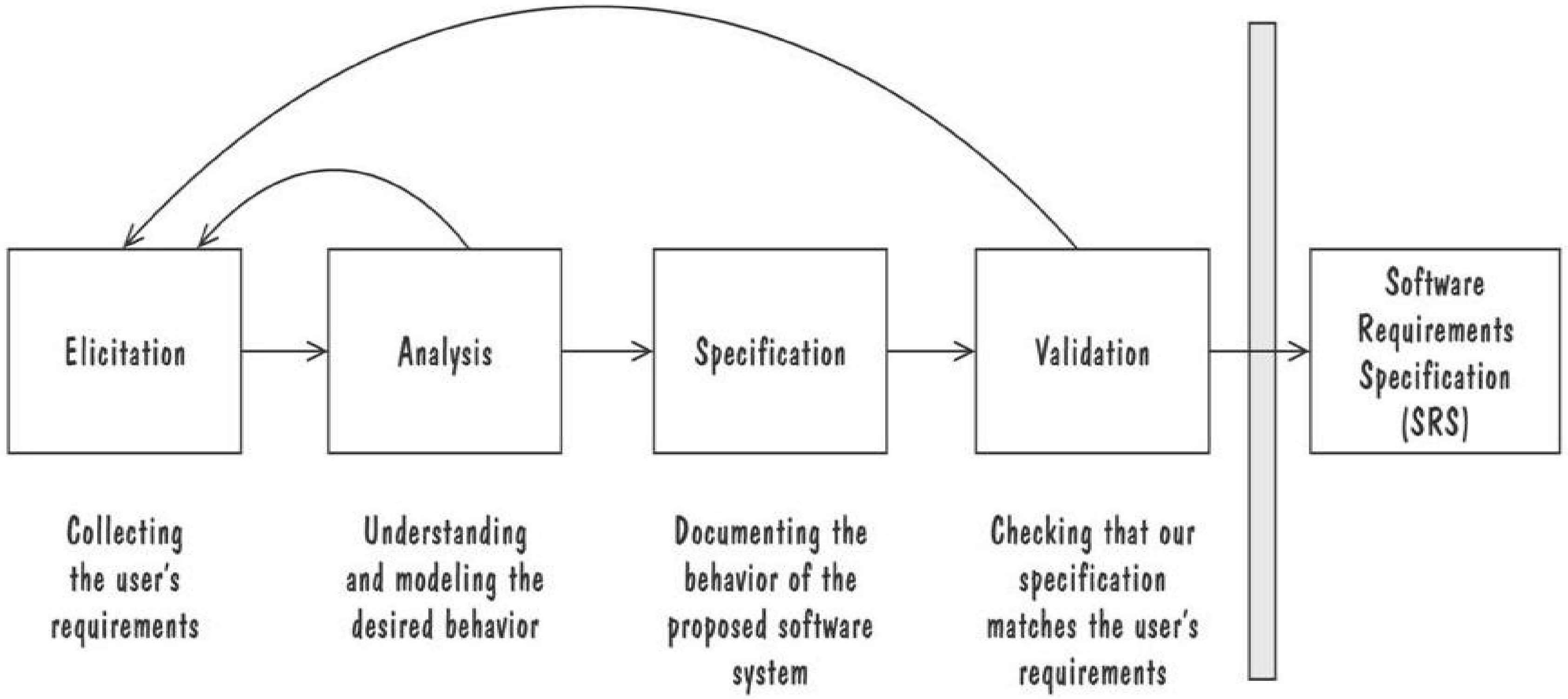
FAST-NUCES PESHAWAR

CONTENTS OF WEEK # 5

- Requirement Engineering (Cont…)
- Analysis Modeling with UML
- Object Oriented Analysis
- Modeling with UML (Static and Dynamic Models)
- Design Principles and Concepts
- Assignment # 1

THE REQUIREMENTS PROCESS
(PROCESS FOR CAPTURING REQUIREMENTS)

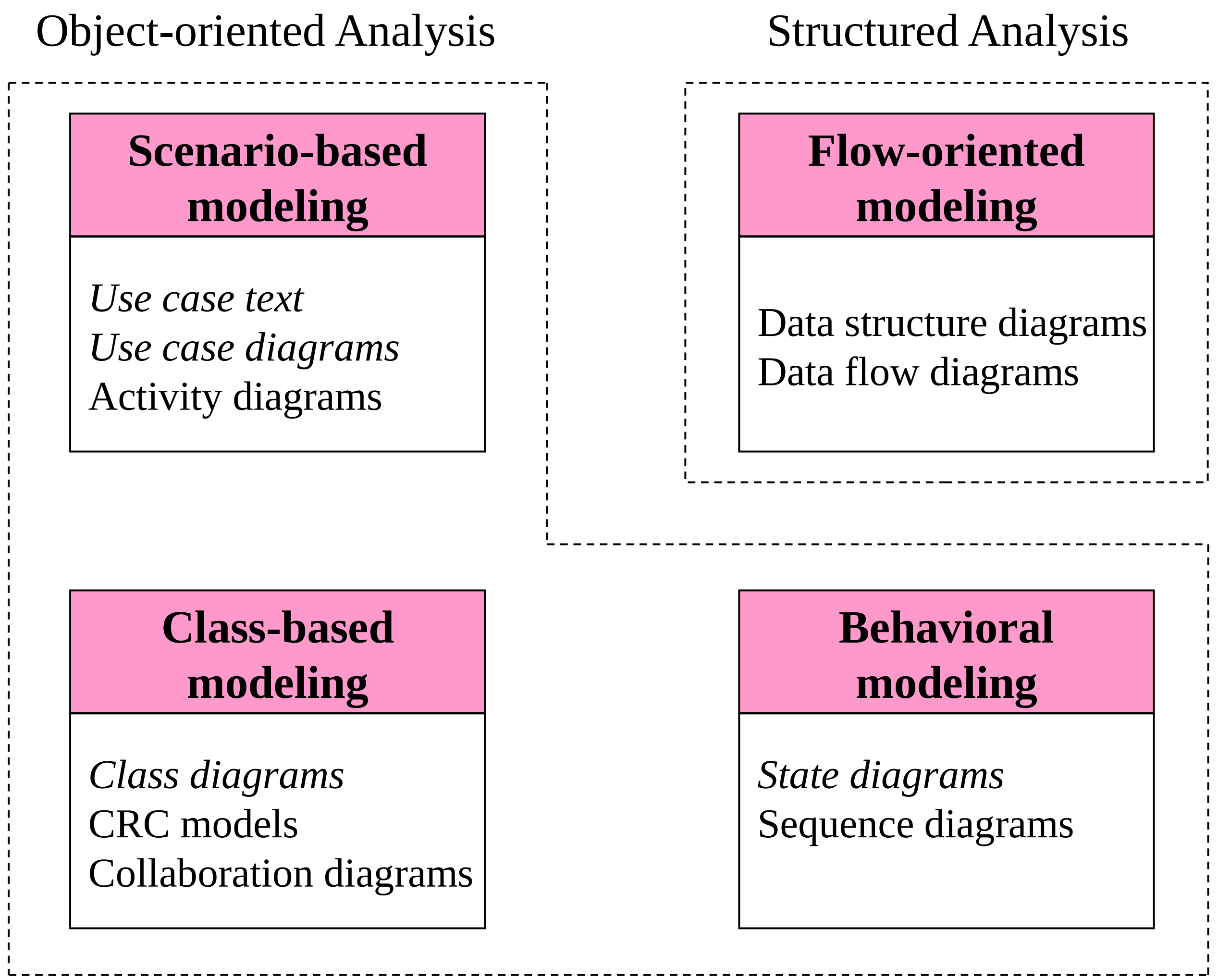
- Performed by the req. analyst or system analyst
- The final outcome is a Software Requirements Specification (SRS) document



Elicitation
Analysis
Specification
Validation
Software Requirements Specification (SRS)
Collecting the user's requirements
Understanding and modeling the desired behavior
Documenting the behavior of the proposed software system
Checking that our specification matches the user's requirements

ANALYSIS MODELING

ELEMENTS OF THE ANALYSIS MODEL



UNIFIED MODELING LANGUAGE (UML)

WHAT IS UML

- UML is the “Unified” Modelling Language”.
- Standard modelling language for software development.
- Maintained by Object Management Group.

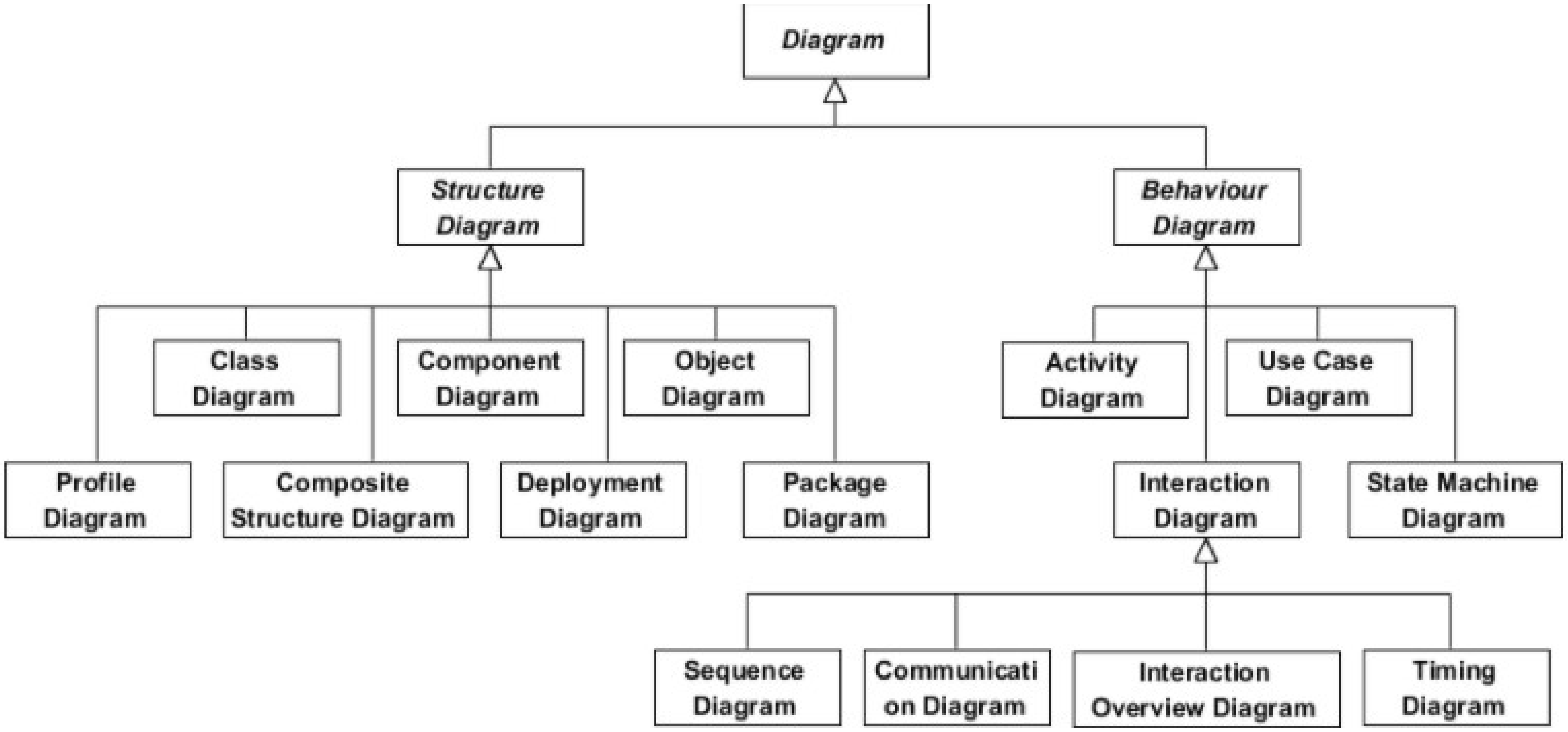
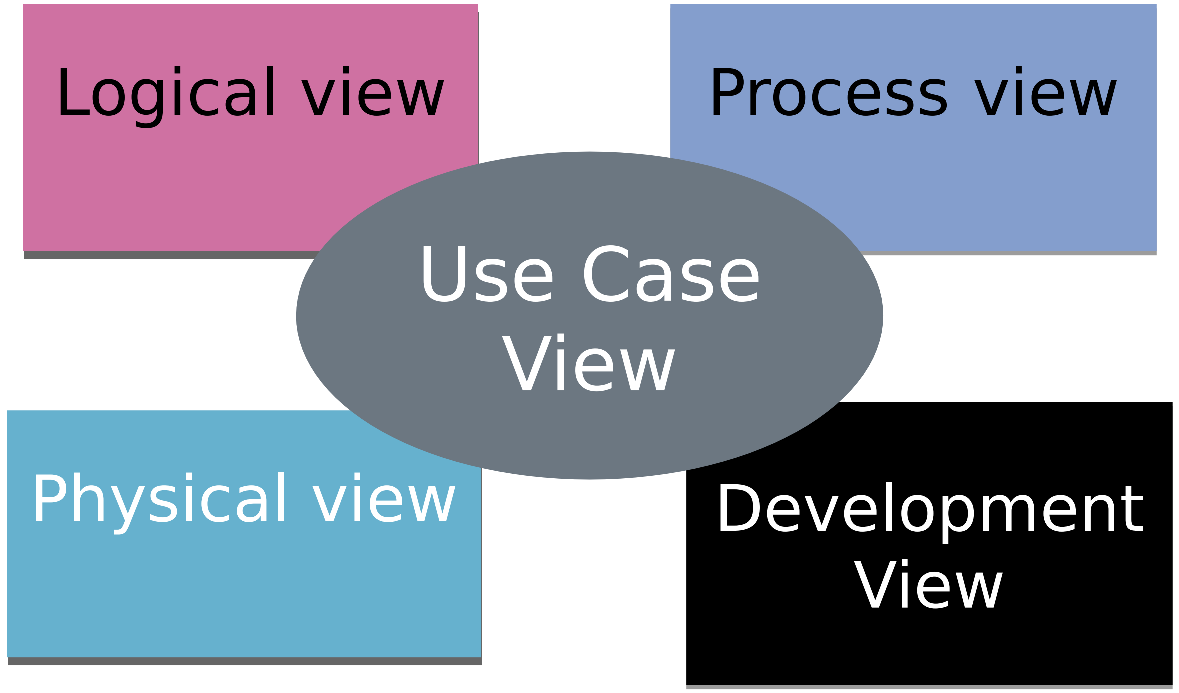


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Activity Diagram
Use Case Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram

VIEW OF THE MODEL

- UML model diagrams can be broken into different perspective or view.
- Can be explained through kruchen's 4+1 Model.



LOGICAL VIEW

- The logical view shows the parts that make up the system and how they interact with each other.
- It represents the abstractions that are used in the problem domain
  - These abstractions are classes and objects
- Different UML diagrams show the logical way such as class diagram state diagram sequence, diagram communication diagram and object diagram.

Logical view

- class diagram
- state diagram
- Sequence diagram
- communication diagram
- object diagram

PROCESS VIEW

- Then we have the process view
- Through this view, we can describe the processes of the system and how they communicate with each other using process
- Using process view, we can find out what needs to happen to the system
- So using process view we can understand the overall functioning of the system
- Activity diagram in UML represents the process view

Process view

- Activity Diagram

PHYSICAL VIEW

- Next is physical view
- The physical view is the view that models the execution environment of the system
- Using this view, we can model the software entities onto the hardware that will host and run the entities
- The physical view in UML is represented through deployment diagrams

Physical view

- Deployment Diagrams

DEVELOPMENT VIEW

- The development view describes the modules are the components of the system.
- This might include packages or libraries.
- It gives a high-level view of the architecture of the system and helps in managing the layers of the system.
- UML provides two diagrams for development view.
  - component Diagram
  - package Diagrams
- All these four views are dependent on Use Case view

Development View

- Component Diagram
- Packages Diagram

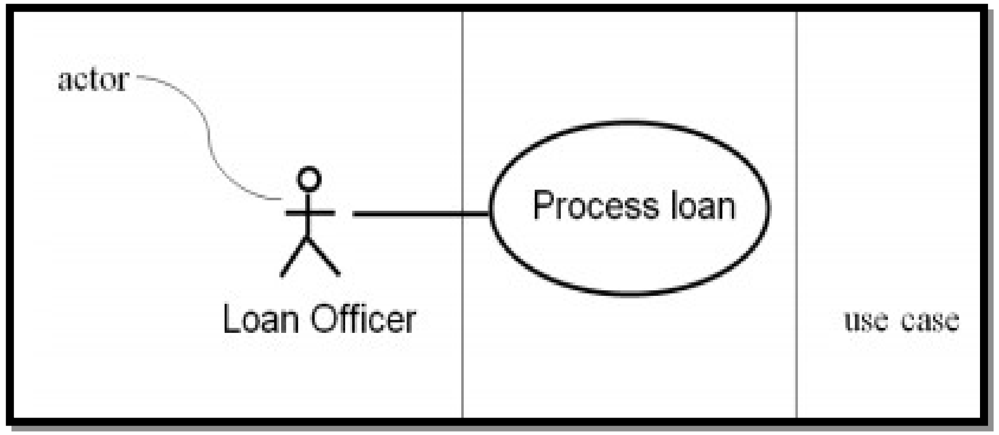
USE CASE

- They use case view illustrates the functionality of the system.
- Using use case we can capture the goals of the user or what the user expects from the system.
- In UML, Use Cases can be created through use case diagrams or use case descriptions (we will discuss it later).
- Use cases can be created by analysts' architects or even by the users.

SCENARIO-BASED MODELING

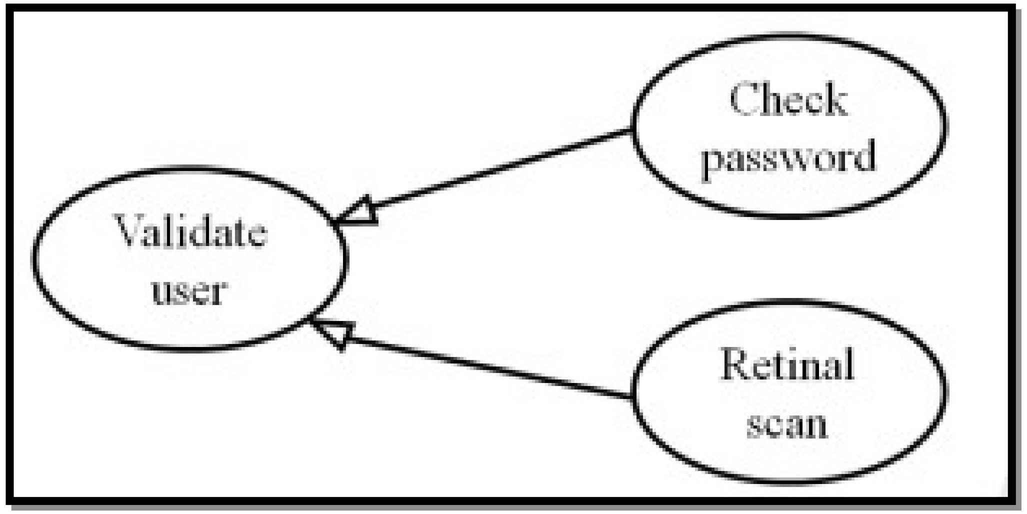
DEVELOPING USE CASES

- – Define the set of actors that will be involved in the story
  - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
  - Actors are anything that communicate with the system or product and that are external to the system itself



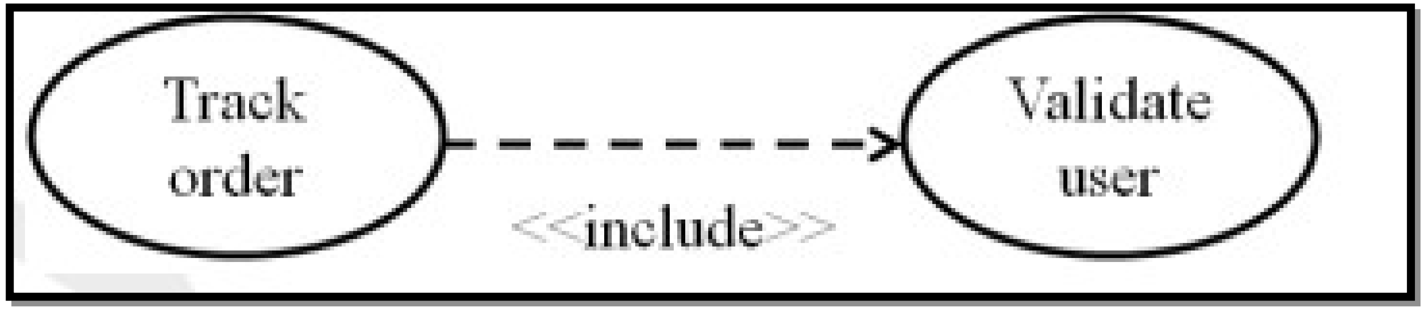
SPECIALIZED USE CASES

You may have two specialized children of this use case (Check password and Retinal scan).



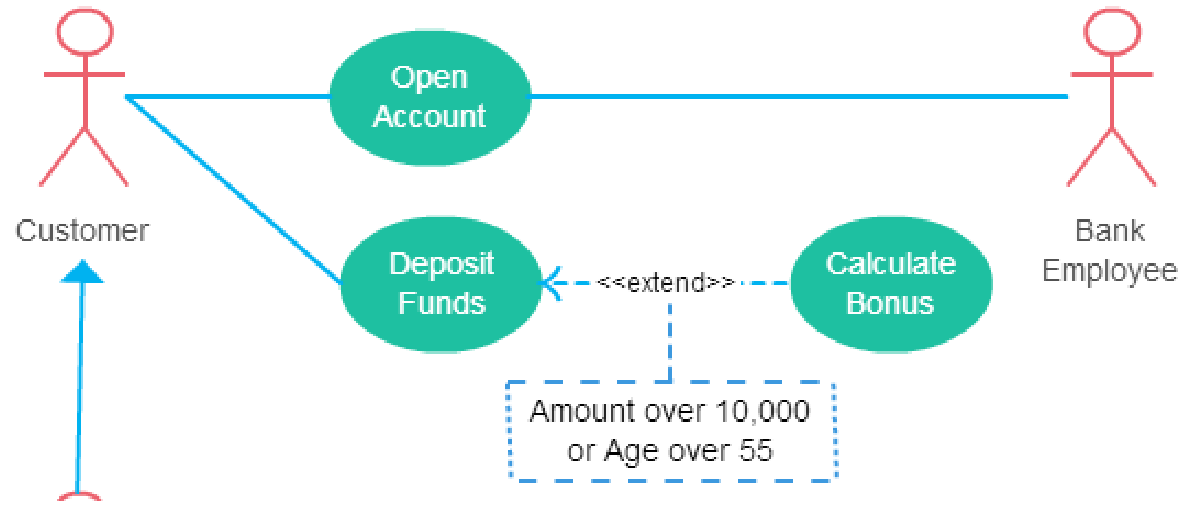
INCLUDE

- Include relationship is used to avoid describing the same flow of events several times, by putting the common behavior in a use case of its own
- This is an example of dependency

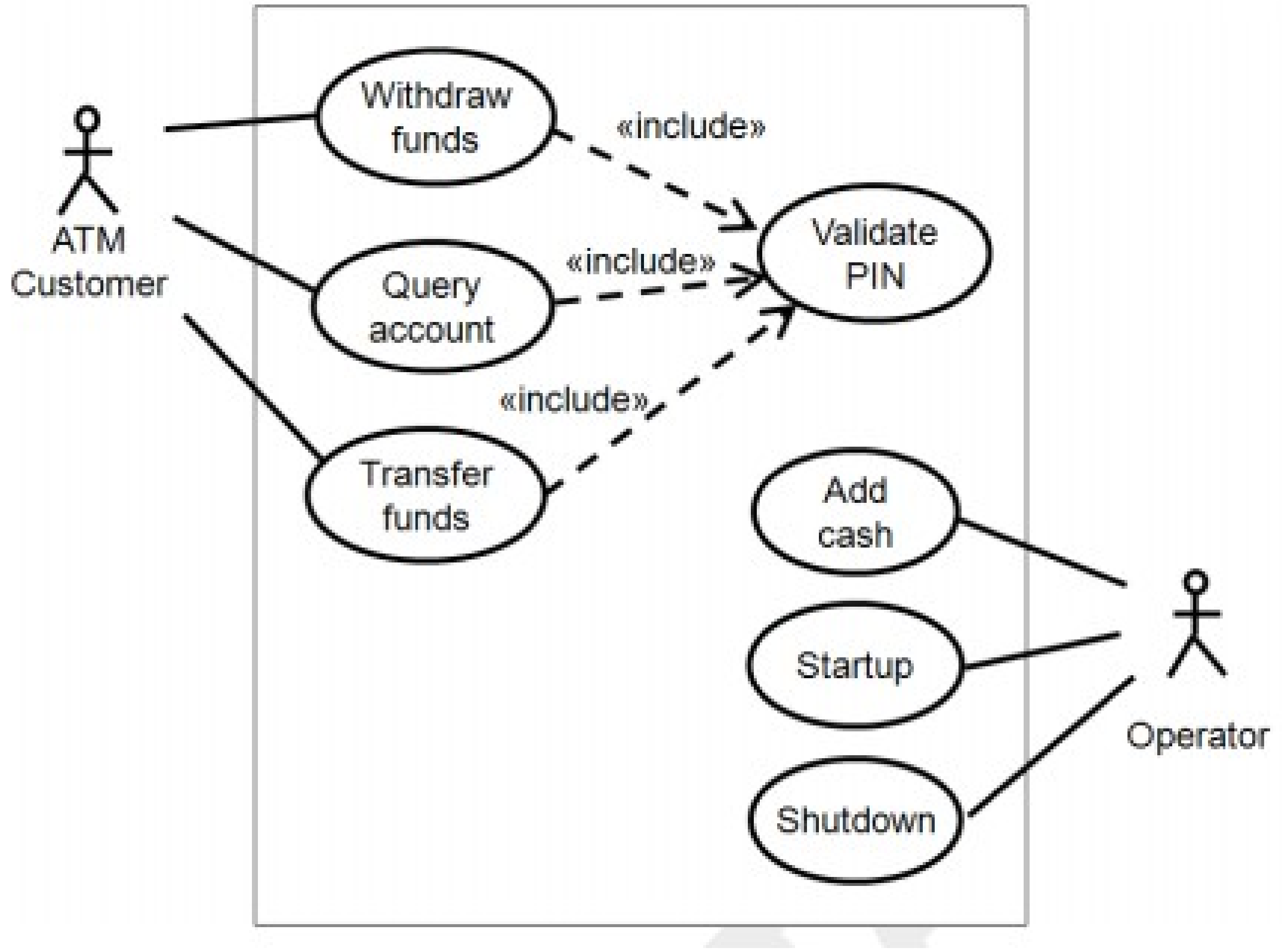


EXTEND

- **The extending use case is usually optional** and can be triggered conditionally. In the diagram, you can see that the extending use case is triggered only for deposits over 10,000 or when the age is over 55.



USE CASE DIAGRAM FOR ATM



Use Case Example

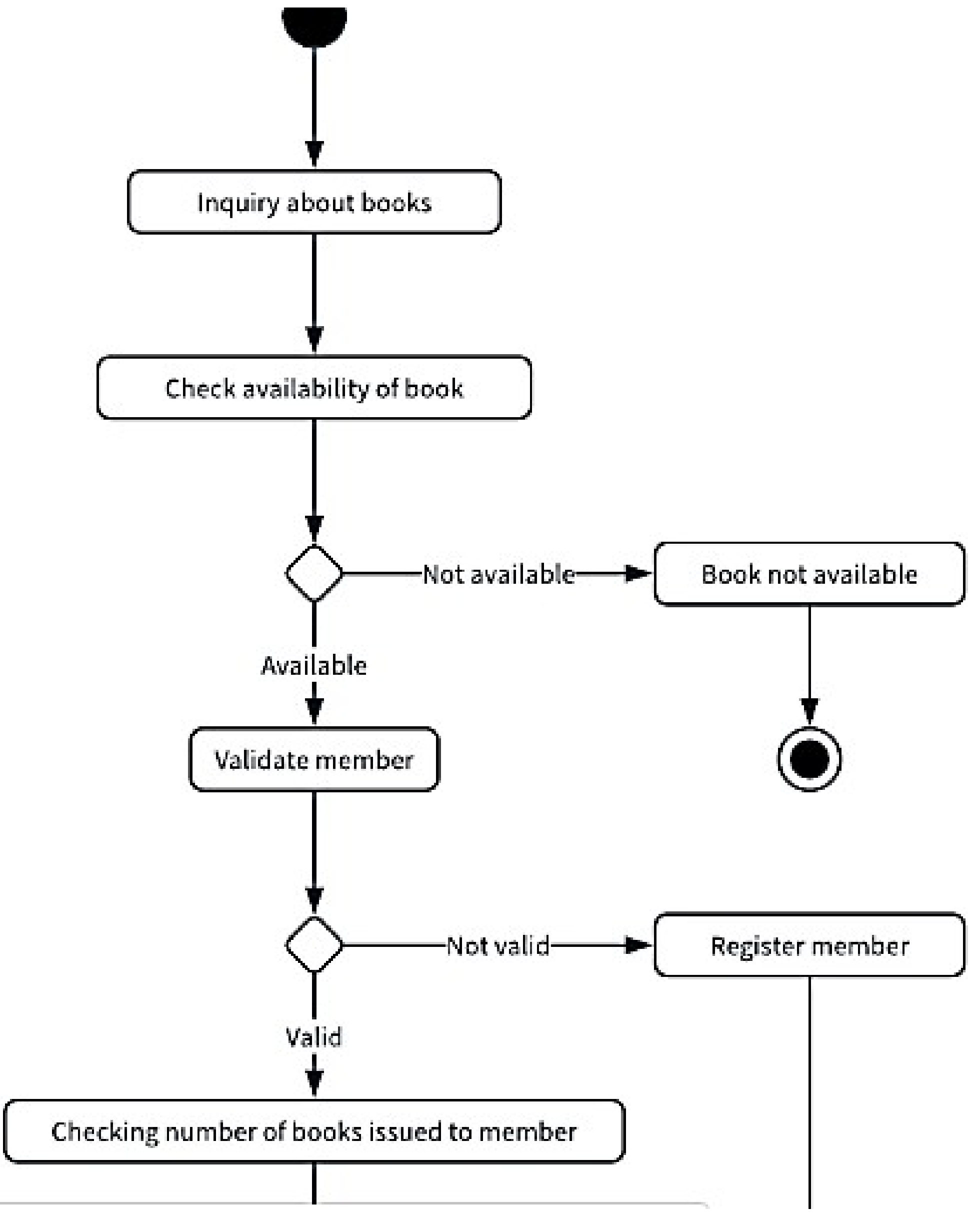
Name	Validate PIN
Summary	System validates customer PIN
Dependency	none
Actors	ATM, Customer
Preconditions	ATM is idle, displaying a Welcome message.
Flow of Events	Activity Diagram
Alternatives	• If the system does not recognize the card, the card is ejected. • If the system determines that the card date has expired, the card is confiscated. • If the system determines that the card has been reported lost or stolen, the card is confiscated. • If the customer-entered PIN does not match the PIN number for this card, the system re-prompts for PIN. • If the customer enter the incorrect PIN three times, the system confiscates the card. • If the customer enters Cancel, the system cancels the transaction and ejects the card
Post condition	Customer PIN has been validated.

BEHAVIORAL MODELING

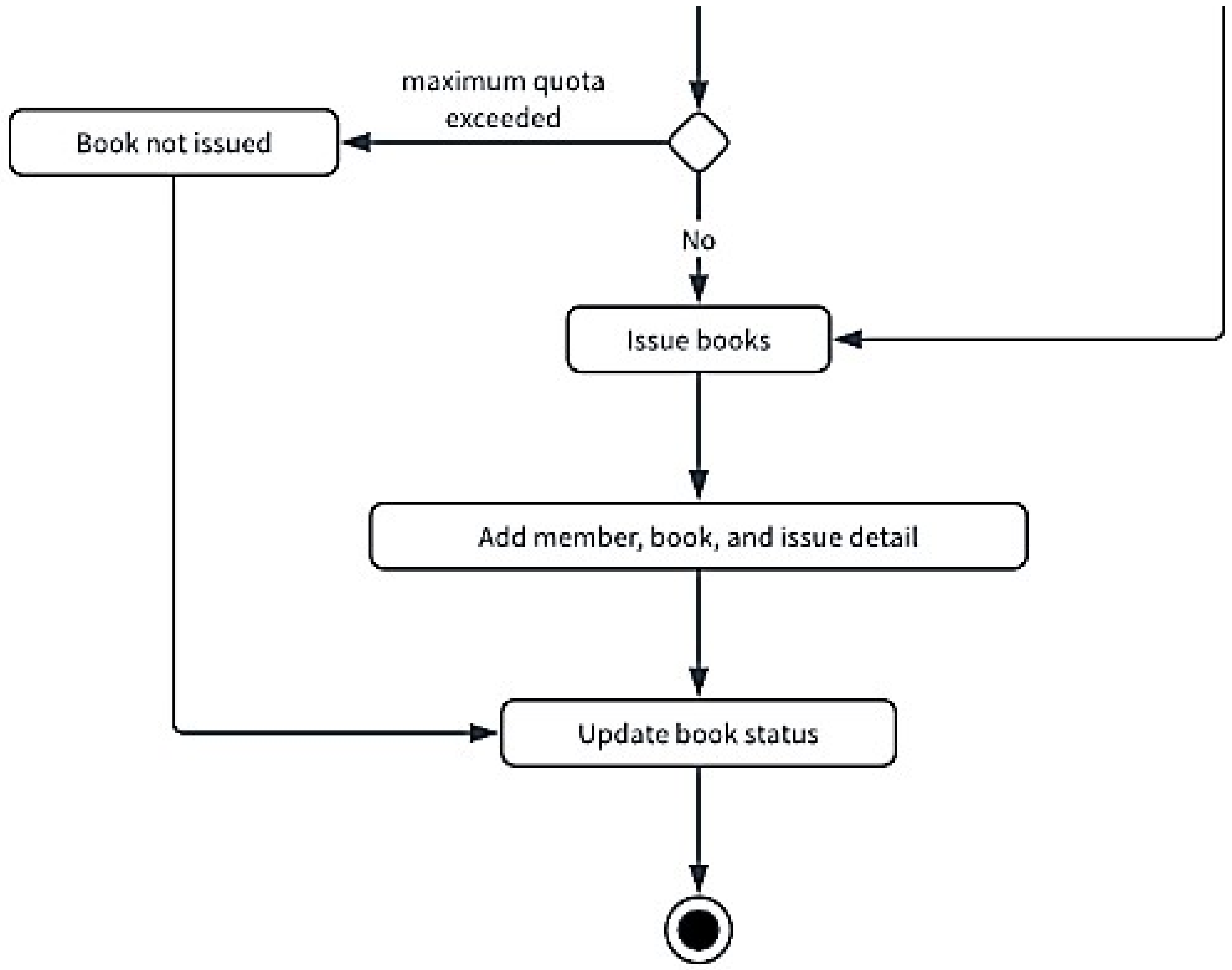
ACTIVITY DIAGRAM

- An activity diagram in the use-case model can be used to capture the activities and actions performed in a use case.
- It expresses the dynamic aspect of the system.

Activity diagram



Activity diagram

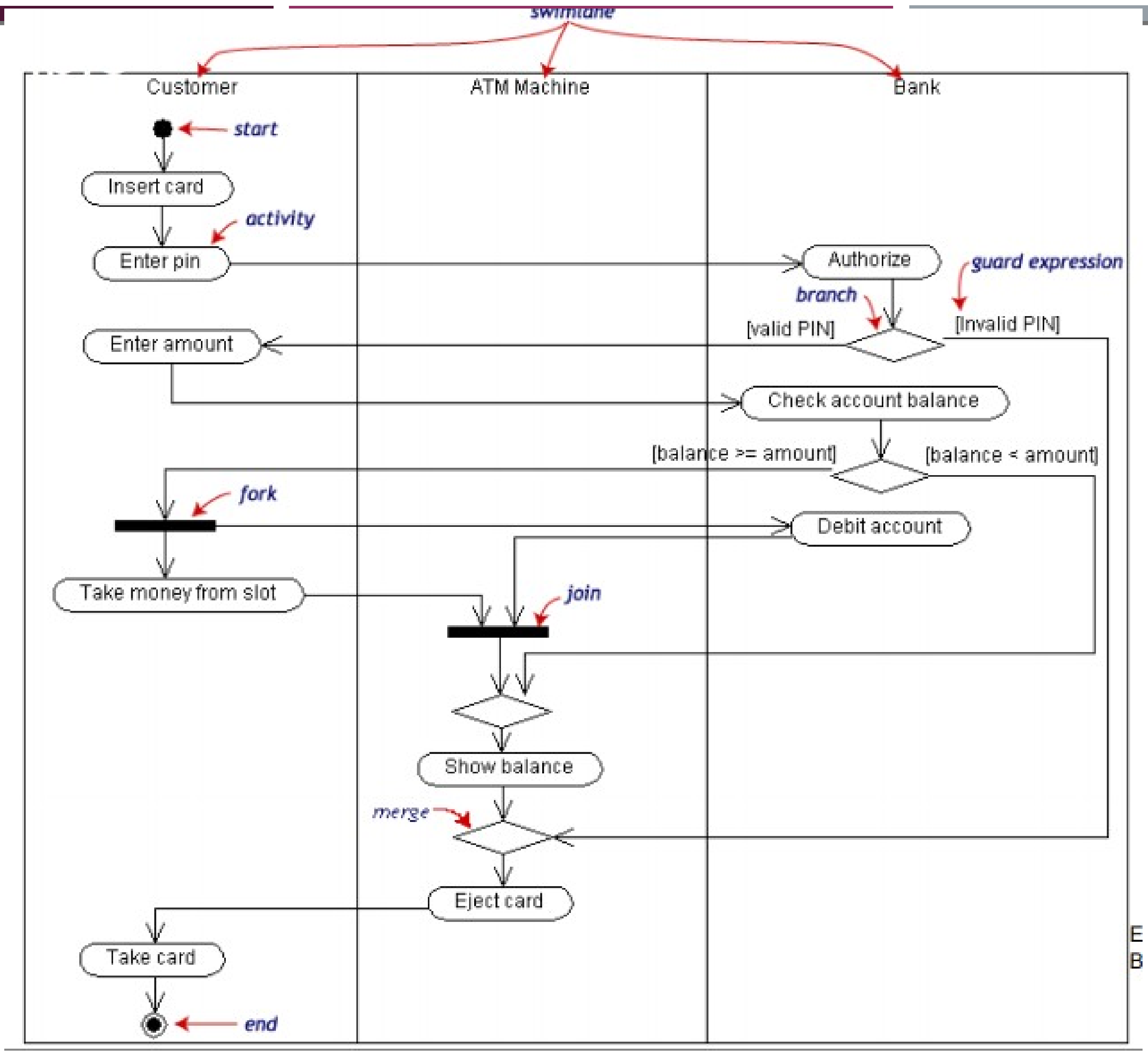


SWIMLANE DIAGRAM

Mapping Who Does What to Whom

- You are assigning a responsibility to an actor.
- Note, we did not say to an object - to an actor.

Swimlane Diagram



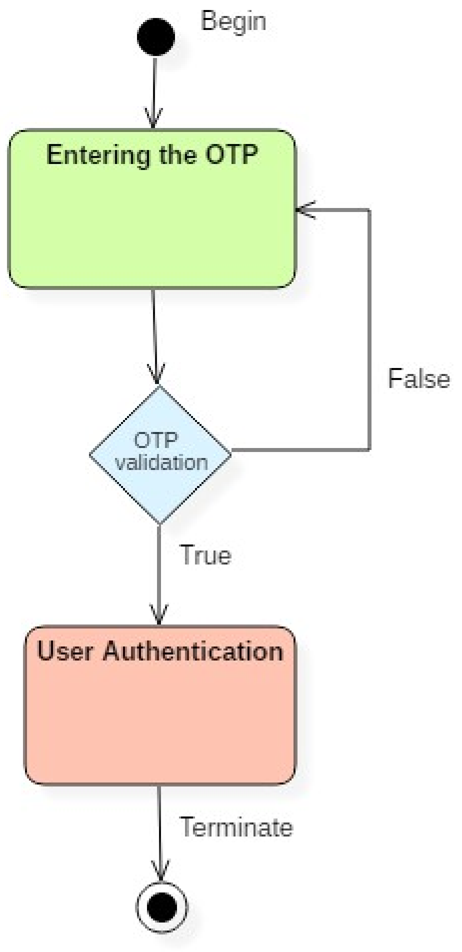
STATE MACHINE DIAGRAM

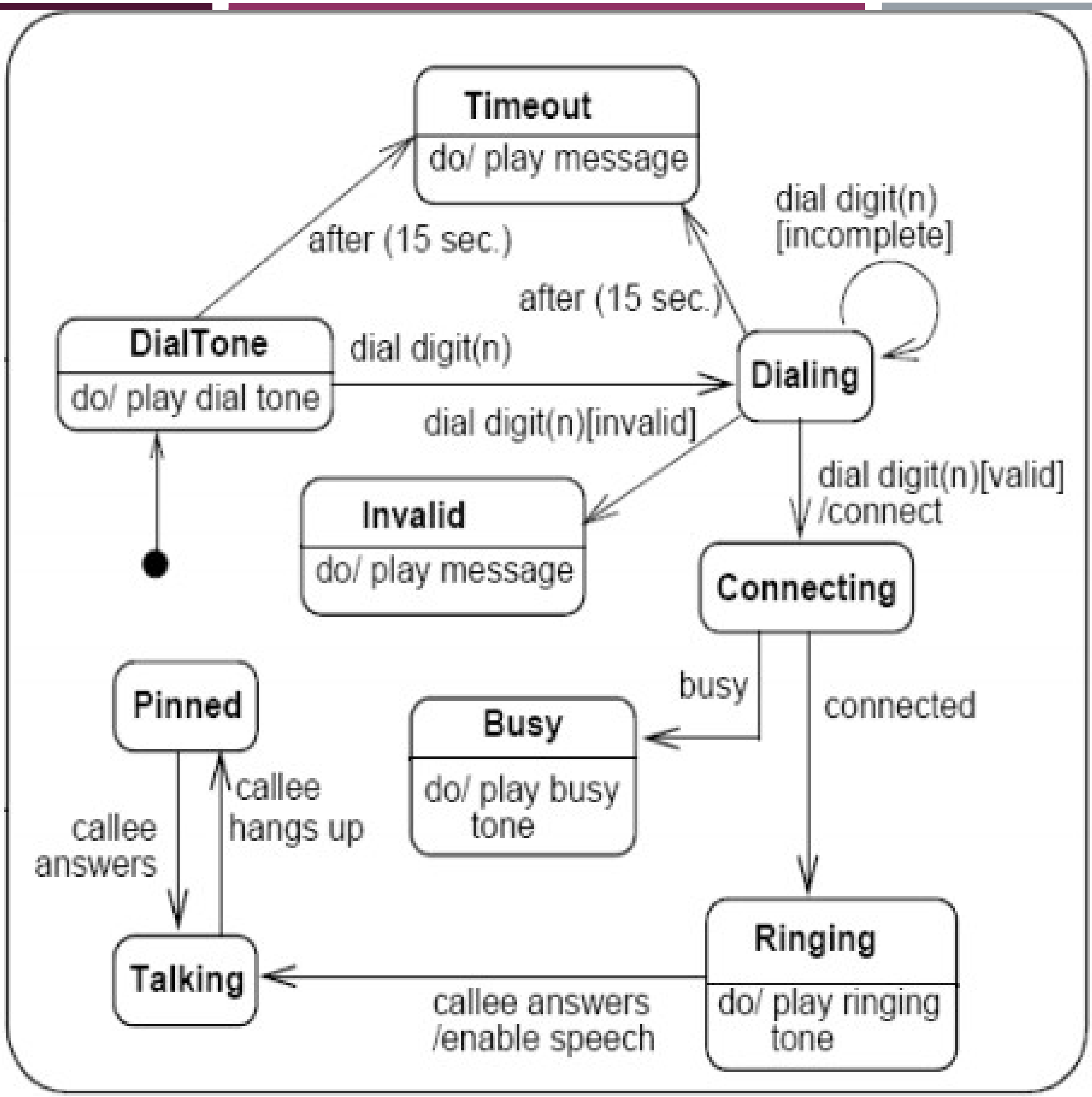
- A state machine diagram models dynamic behavior.
- It specifies the sequence of states in which an object can exist:
- The events and conditions that cause the object to reach those states.
- The actions that take place when those states are reached.

ELEMENT OF STATE MACHINE DIAGRAM

- States
- Events
- Transition

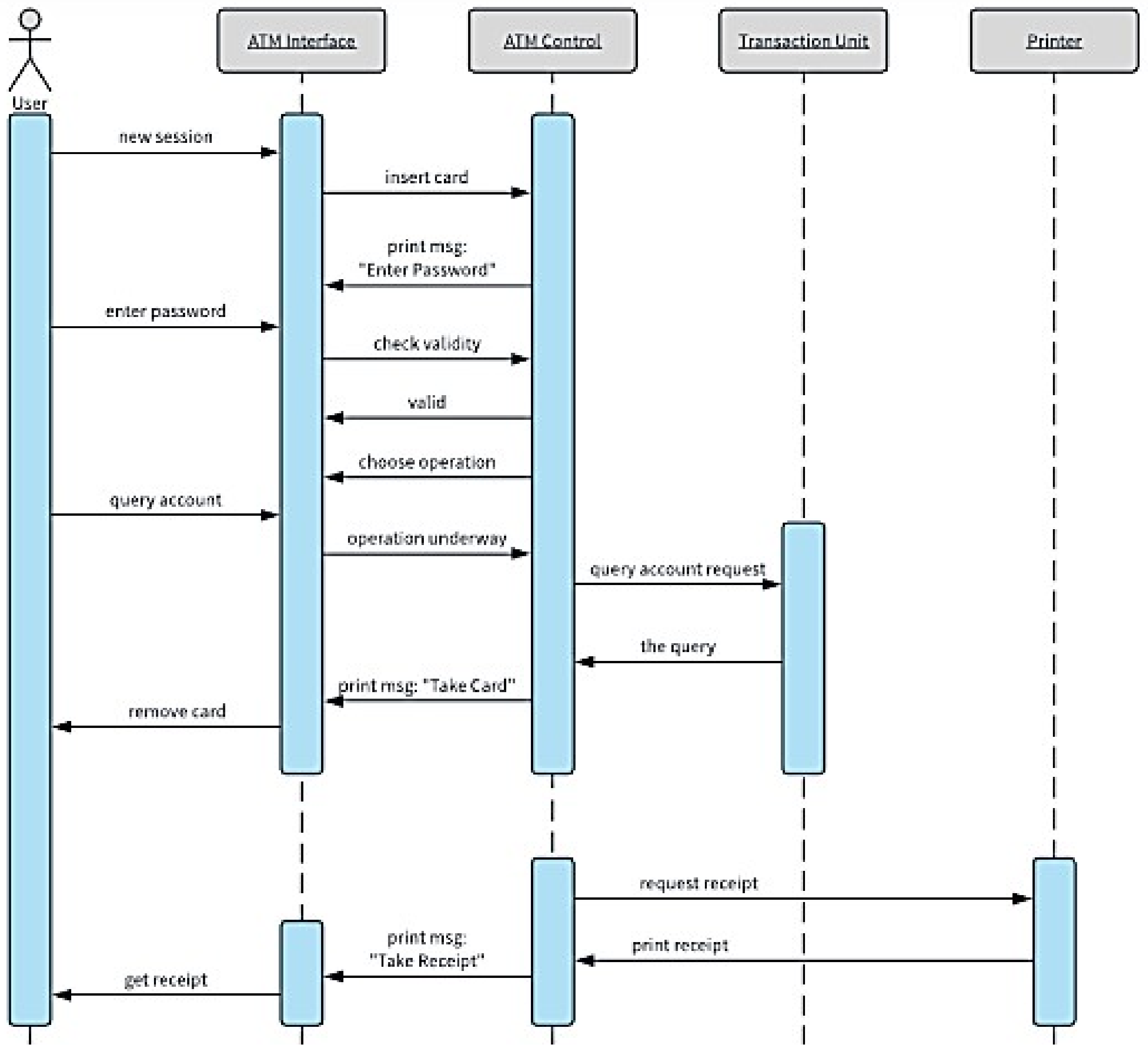
- An object has state, behavior, and a unique identity.
- An object goes through various states during its lifespan.



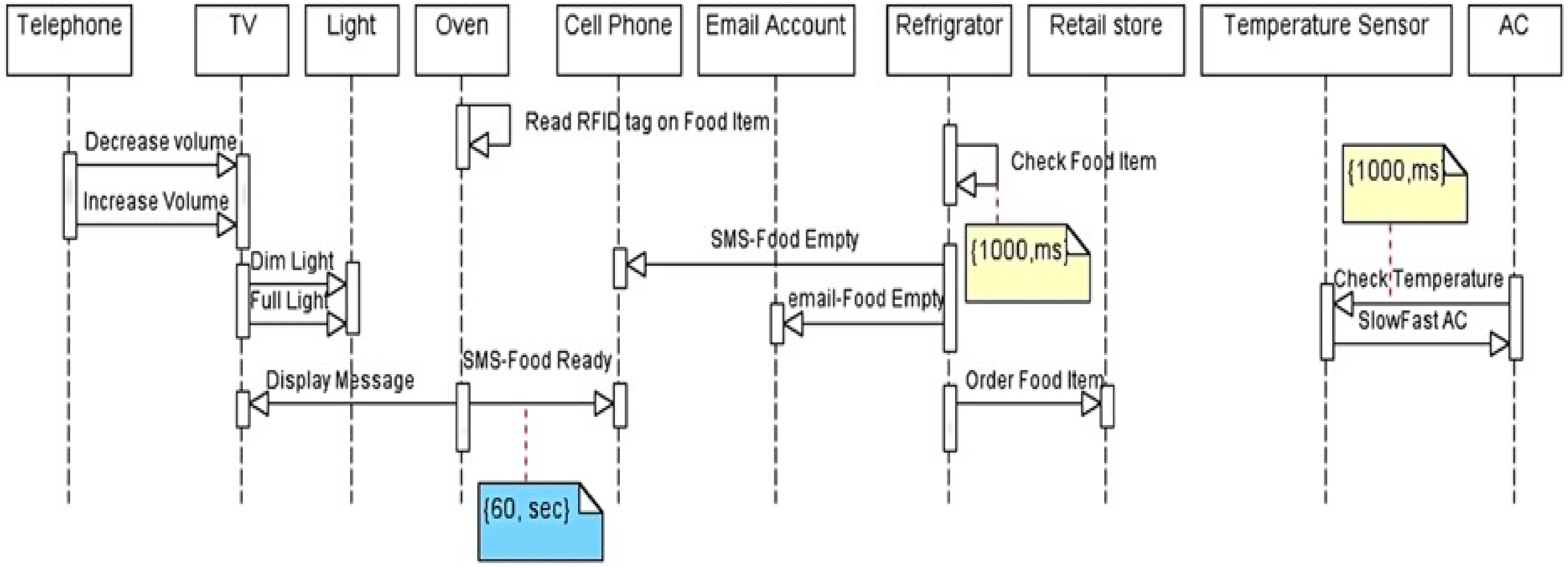

Timeout
do/ play message
dial digit(n)
[incomplete]
after (15 sec.)
after (15 sec.)
DialTone
do/ play dial tone
dial digit(n)
Dialing
dial digit(n)[invalid]
dial digit(n)[valid]
/connect
Invalid
do/ play message
Connecting
busy
connected
Pinned
Busy
do/ play busy tone
callee hangs up
callee answers
Talking
Ringing
do/ play ringing tone
callee answers
/enable speech

SEQUENCE DIAGRAM

- A **sequence diagram** shows object interactions.
- It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.


User
ATM Interface
ATM Control
Transaction Unit
Printer
new session
insert card
print msg: "Enter Password"
enter password
check validity
valid
choose operation
query account
operation underway
query account request
the query
print msg: "Take Card"
remove card
request receipt
print receipt
print msg: "Take Receipt"
get receipt

SYSTEM SEQUENCE DIAGRAM



CASE STUDY
(HOSPITAL MANAGEMENT SYSTEM)

HOSPITAL MANAGEMENT SYSTEM

- The Hospital Management System (HMS) is a web application, which used for the control of hospital services.
- The HMS web application can be accessed by either mobile or computer browser.
- The HMS application combines all details regarding doctors, patients, nurses, hospital administrative, etc. into one software.

HOSPITAL MANAGEMENT SYSTEM

- HMS System allows the patients to register via a registration module (form), which gathers and stores all required patient’s data such as name, e-mail, gender, etc.

- Registered patients can skip this step and login directly using their username and password through the login module. Nevertheless, unregistered users can only take advantage of major system features such as viewing the hospital timings.

- After the patient creates an account and register, he can access the allowed system features/functionalities for patients.

- Patients can view available appointments, book an appointment and manage his/her own profile. After the patient book an appointment, he can visit the hospital according to his appointment.

HOSPITAL MANAGEMENT SYSTEM

- Once the patient reaches the hospital, the receptionist will issue a clinic number for him since the receptionist has access to the system to view the appointments list and status with nurses and doctors.

- The HMS system also allows the receptionist to create patient accounts and book an appointment, referring to the doctors’ schedule, for unregistered patients.

- Once patient’s turn came, the patient can explain his condition to the consulting nurse, so that the nurse performs the pre-assessment examinations to diagnose the problem and then redirect him to the concerned doctor/clinic.

- The HMS system enables the nurse to allot patients for the concerned doctors, to view doctors’ status and to update patients’ account.

HOSPITAL MANAGEMENT SYSTEM

- Then, the concerned doctor will diagnose the patient, and then enter the prescription needed for the patient. If the doctor sees that the patient needs any further examinations like collecting and processing specimens, the system allows the doctor to redirect the patient to the Nurse again.

- After the nurse collects the specimens, the specimens will be sent to the laboratory so that the lab assistant can process, analyze the specimens, and then generate and enter the test results into the system.

- Furthermore, the doctor can redirect the patient to the lab assistant if there is a need to perform examinations such as X-Ray images, CT scan, MRI. The lab assistant can access the system and generate test reports regarding the examinations or test performed.

HOSPITAL MANAGEMENT SYSTEM

- On the other hand, the doctor keeps track of the examination results entered by the lab assistant and then recommend further actions to be taken if required, as well as enters a new prescription for the patient.
- The system also allows the patient to access his account to see prescription details and view his reports along with doctor advice.
- This feature is very useful since test reports usually take a long time to be generated, so that the patient may leave the hospital and view the results along with doctor’s advice through his account without the need of going to the hospital again.

HOSPITAL MANAGEMENT SYSTEM

- Once the prescription is ready, the pharmacist will prepare the medicines for the patient and enters the dose and guidelines of each medicine into the system.
- When the patient goes to the pharmacy of the hospital, he/she will find the medicines ready so that he/she can pick and go easily.
- The patient has two options to know the dose and guidelines of each medicine, either by asking the pharmacist directly or by accessing his/her account to see it.
- This will help the patient be aware of the medicines' dose if he/she forgets it. Finally, the patient will need to go to the cashier to pay for his/her visit.
- The system allows the cashier to create and order invoice for payment through the billing module. In addition, the cashier can watch the payment history of the patients.

DESIGN PRINCIPLES

DESIGN PRINCIPLES

1- The design process should not suffer from “tunnel vision.”

A good designer should consider alternative approaches, judging each based on the requirements of the problem, the resources available to do the job, and the design concepts.

2- The design should be traceable to the analysis model.

DESIGN PRINCIPLES

3- The design should not reinvent the wheel.

- Systems are constructed using a set of design patterns.
- These patterns should always be chosen as an alternative to reinvention.
- Time is short and resources are limited!

DESIGN PRINCIPLES

4- The design should “minimize the intellectual distance” between the software and the problem as it exists in the real world.

DESIGN PRINCIPLES

5- The design should be structured to accommodate change

6- The design should be assessed for quality as it is being created

DESIGN CONCEPTS

FUNDAMENTAL CONCEPTS OF DESIGN

- **abstraction**—data, procedure, control
- **refinement**—elaboration of detail for all abstractions
- **modularity**—compartmentalization of data and function
- **architecture**—overall structure of the software
  - Styles and patterns
- **procedure**—the algorithms that achieve function
- **hiding**—controlled interfaces

ABSTRACTION

“Capture only those details about an object that are relevant to current perspective”

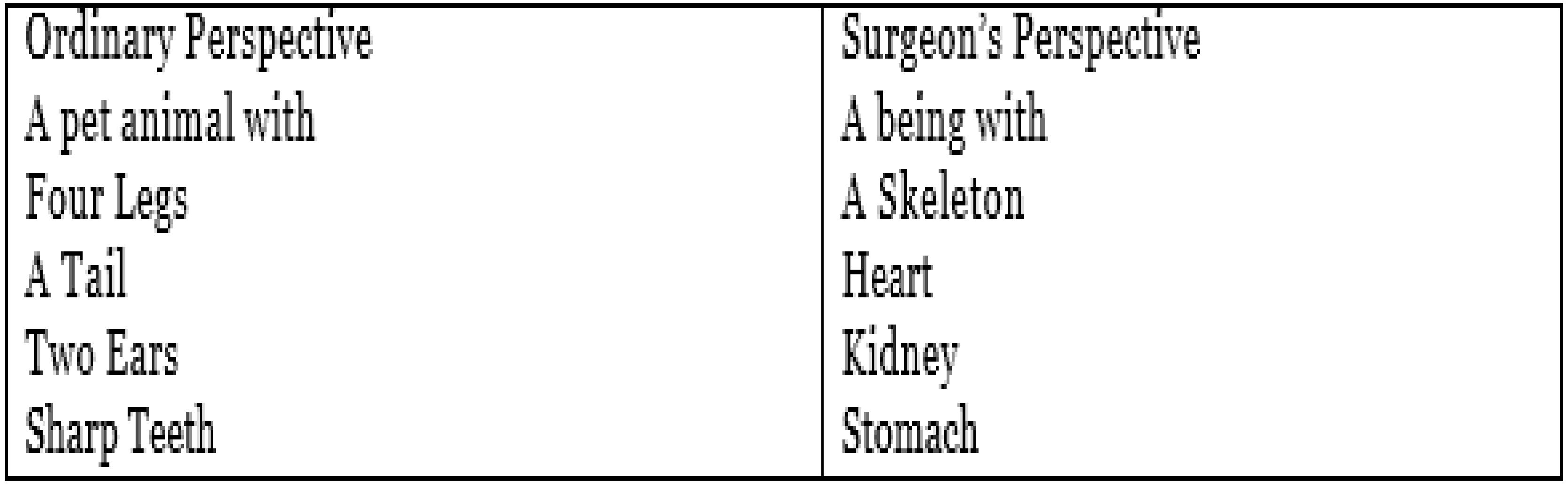
Suppose we want to implement abstraction for the following statement,

“Ali is a PhD student and teaches BS students”

Here object Ali has two ***perspectives*** *one is his* ***student perspective*** *and second is his* ***teacher perspective.***

ABSTRACTION

A cat can be viewed with different perspectives.



Ordinary Perspective	Surgeon’s Perspective
A pet animal with	A being with
Four Legs	A Skeleton
A Tail	Heart
Two Ears	Kidney
Sharp Teeth	Stomach

ABSTRACTION

Driver's View	Engineer's View

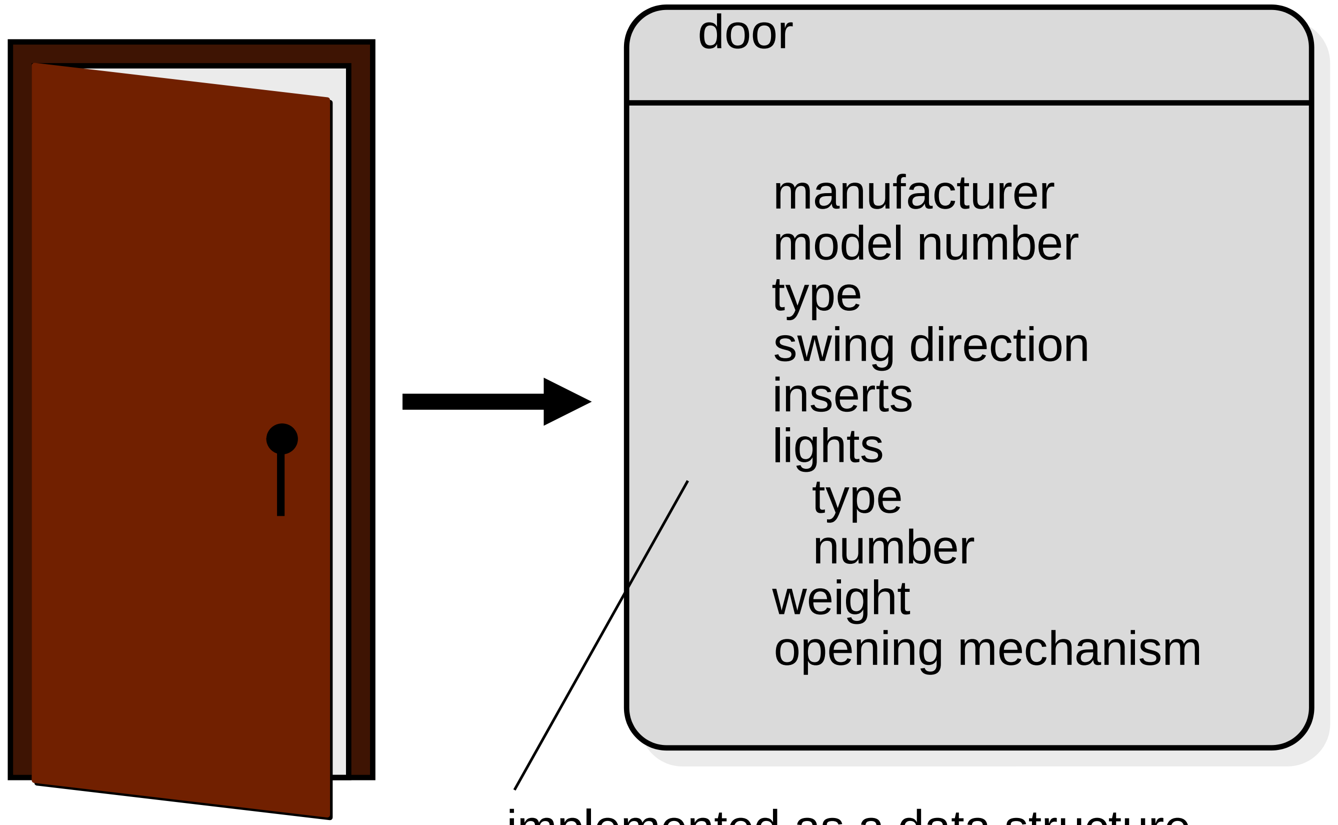
ABSTRACTION ADVANTAGES

- It helps us understanding and solving a problem using object oriented approach as it hides extra irrelevant details of objects.

- Focusing on single perspective of an object provides us freedom to change implementation for other aspects of for an object later.

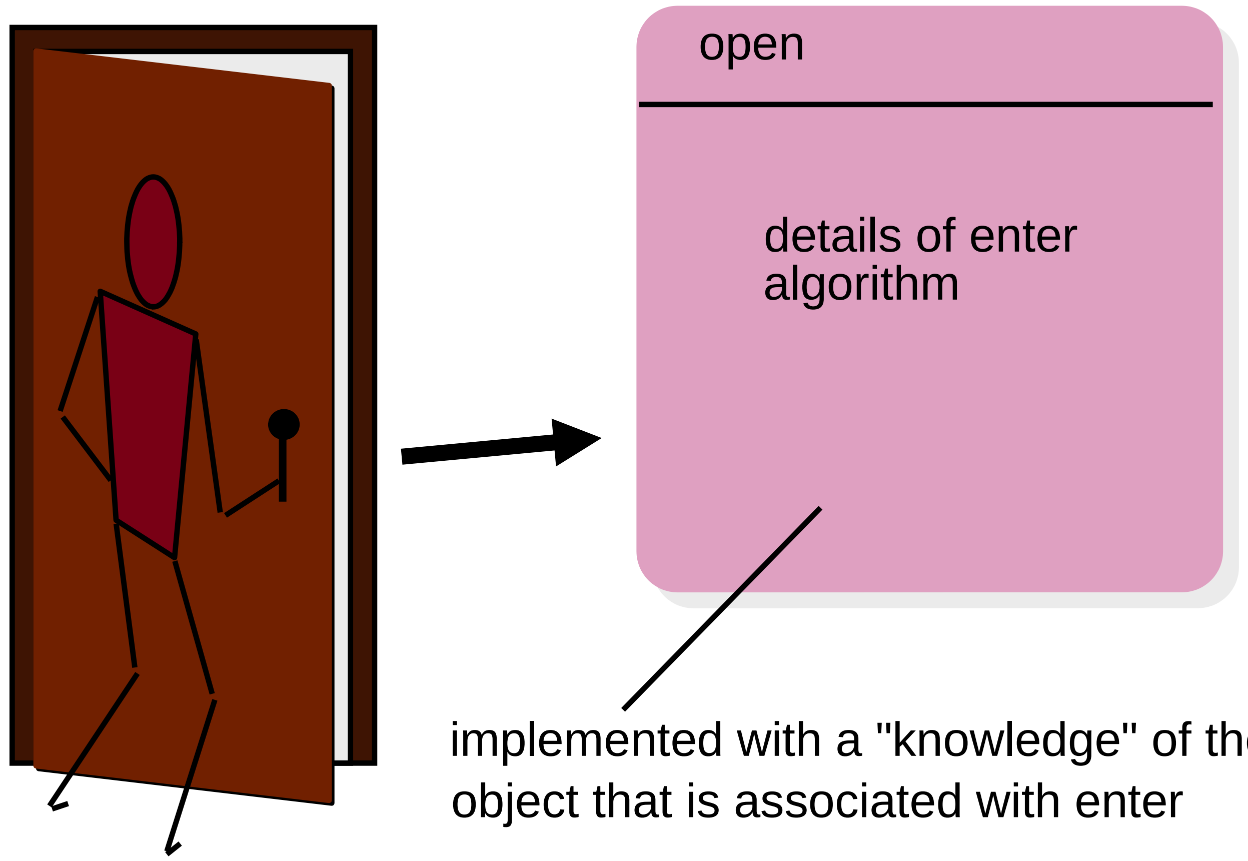
- *Abstraction is used for achieving information hiding as we show only relevant details to related objects, and hide other details.*

DATA ABSTRACTION



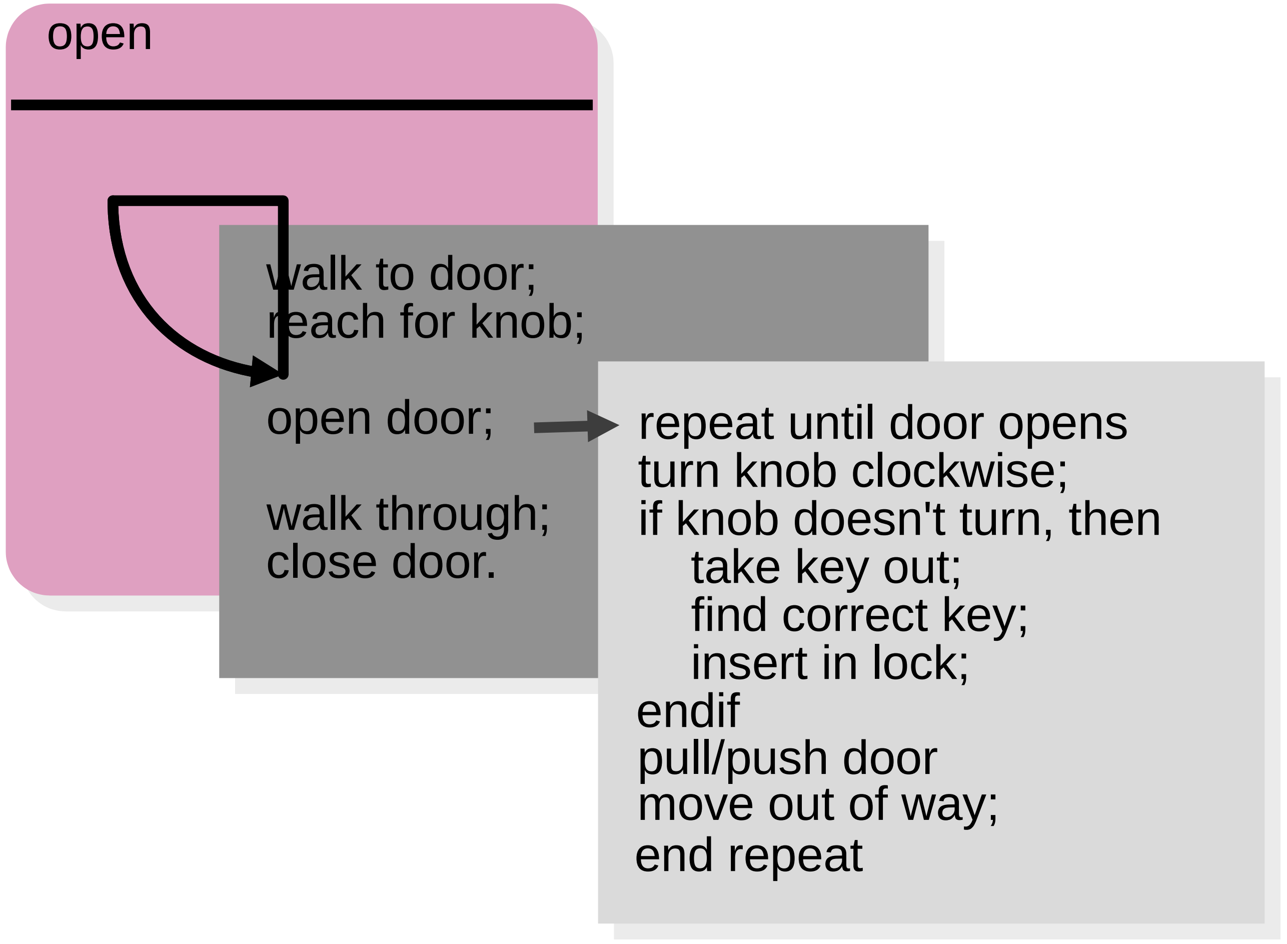
implemented as a data structure

PROCEDURAL ABSTRACTION



implemented with a "knowledge" of the object that is associated with enter

STEPWISE REFINEMENT

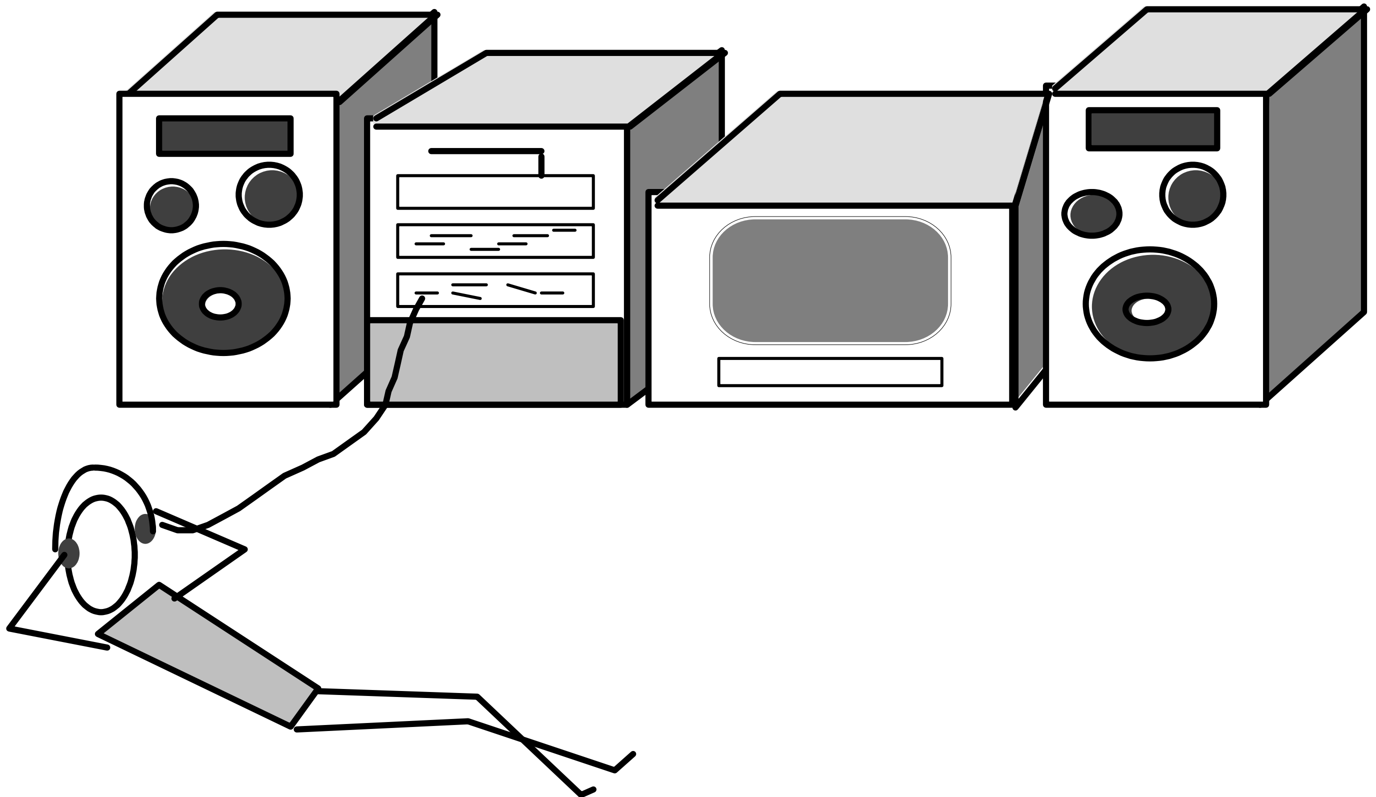


```
walk to door;
reach for knob;

open door;

walk through;
close door.
```

```
repeat until door opens
turn knob clockwise;
if knob doesn't turn, then
    take key out;
    find correct key;
    insert in lock;
endif
pull/push door
move out of way;
end repeat
```

MODULAR DESIGN

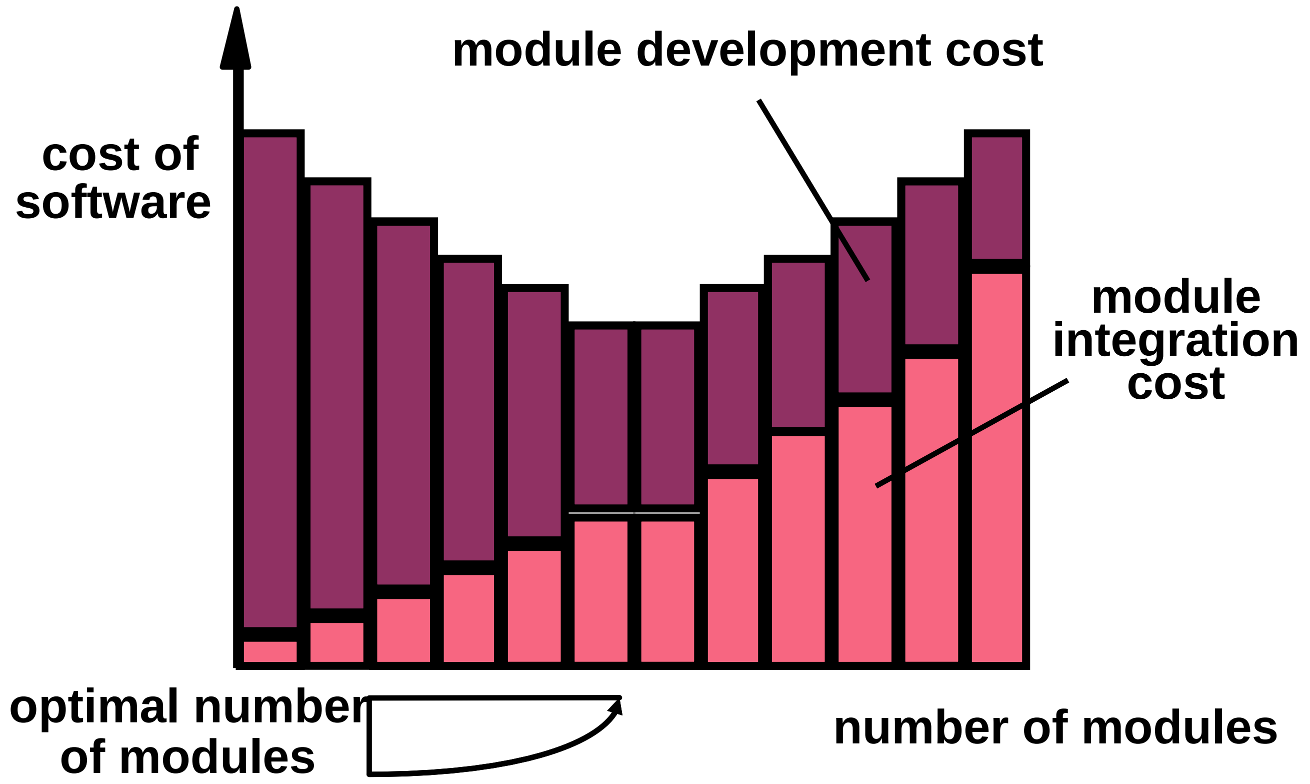
easier to build, easier to change, easier to fix ...



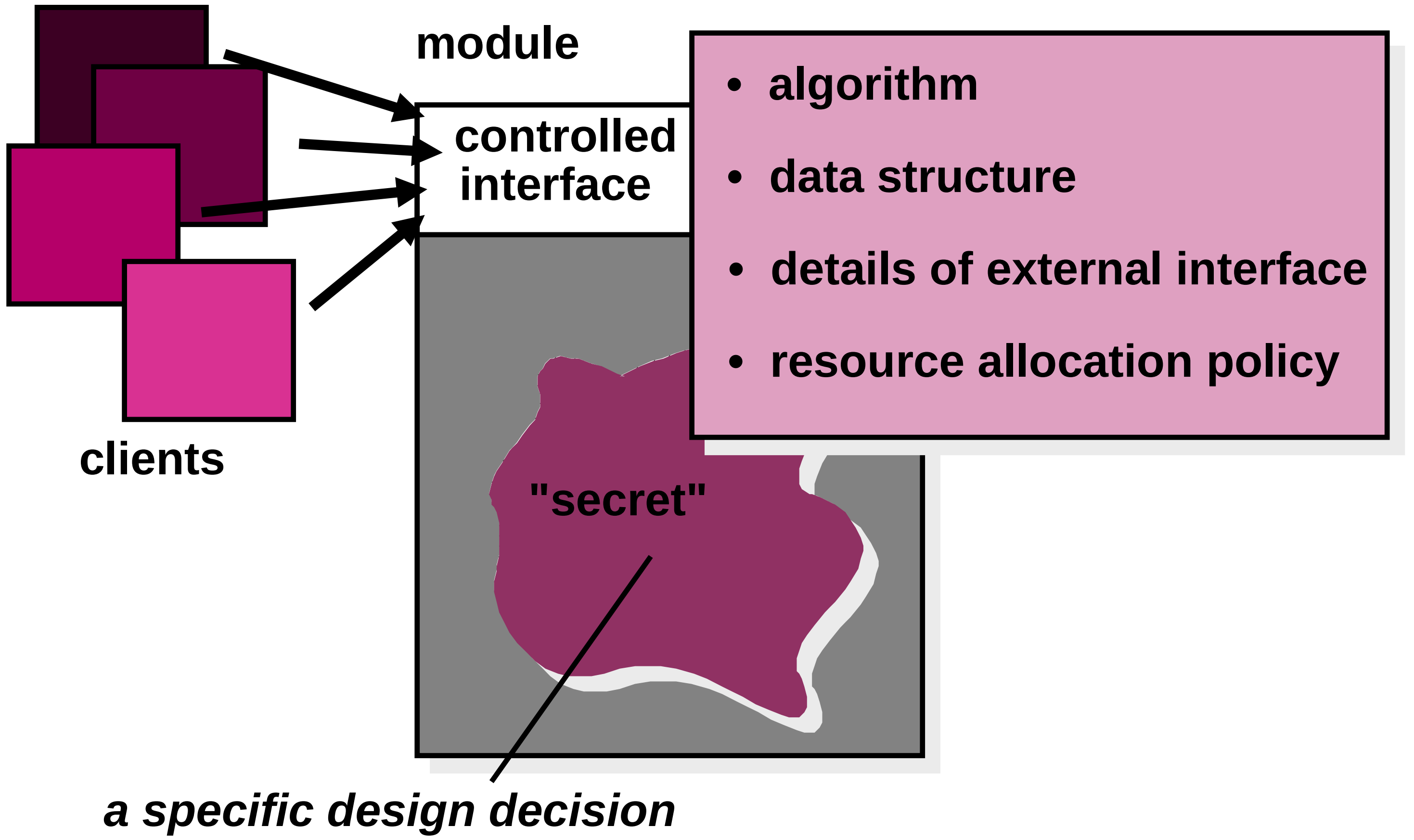
- Easier to manage
- Easier to understand
- Reduces complexity
- Delegation / division of work
- Fault isolation
- Independent development
- Separation of concerns
- Reuse

MODULARITY: TRADE-OFFS

What is the "right" number of modules for a specific software design?



INFORMATION HIDING



- algorithm
- data structure
- details of external interface
- resource allocation policy

INFORMATION HIDING

- Design the modules in such a way that information (data & procedures) contained in one module is inaccessible to other modules that have no need for such information.
- Independent modules.

Benefits:

when modifications are required, it reduces the chances of propagating to other modules.

EFFECTIVE MODULAR DESIGN

FUNCTIONAL INDEPENDENCE

COHESION - the degree to which a module performs one and only one function.

COUPLING - the degree to which a module is "connected" to other modules in the system.

COUPLING

Coupling is a measure of independence of a module or component.

Loose coupling means that different system components have loose or less reliance upon each other.

Hence, changes in one component would have a limited affect on other components.

COUPLING

High coupling causes problems

- Change propagation- ripple effect
- Difficulty in understanding
- Difficult reuse

COHESION

Cohesion is a measure of the degree to which the elements of the module are functionally related.

It is the degree to which all elements directed towards performing a single task are contained in the component.

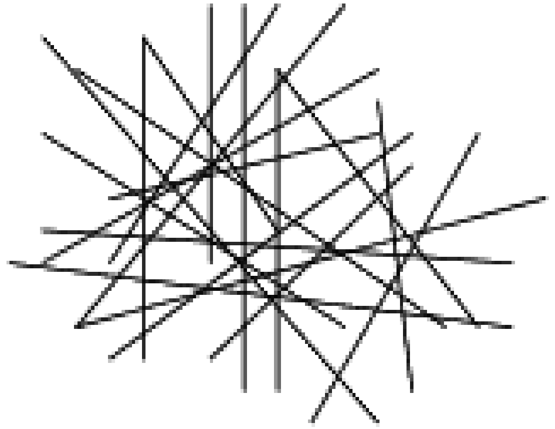
Basically, cohesion is the internal glue that keeps the module together.

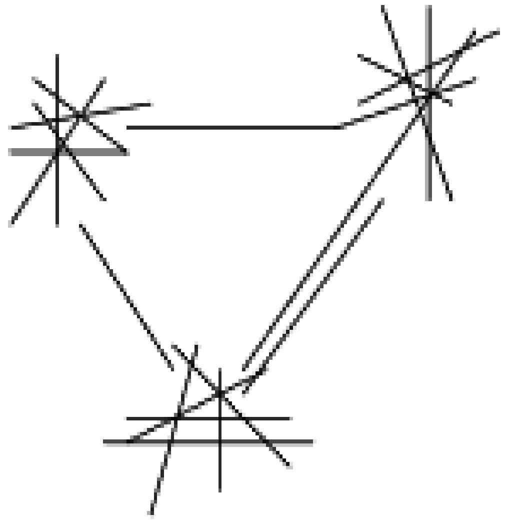
A good software design will have high cohesion

COUPLING & COHESION

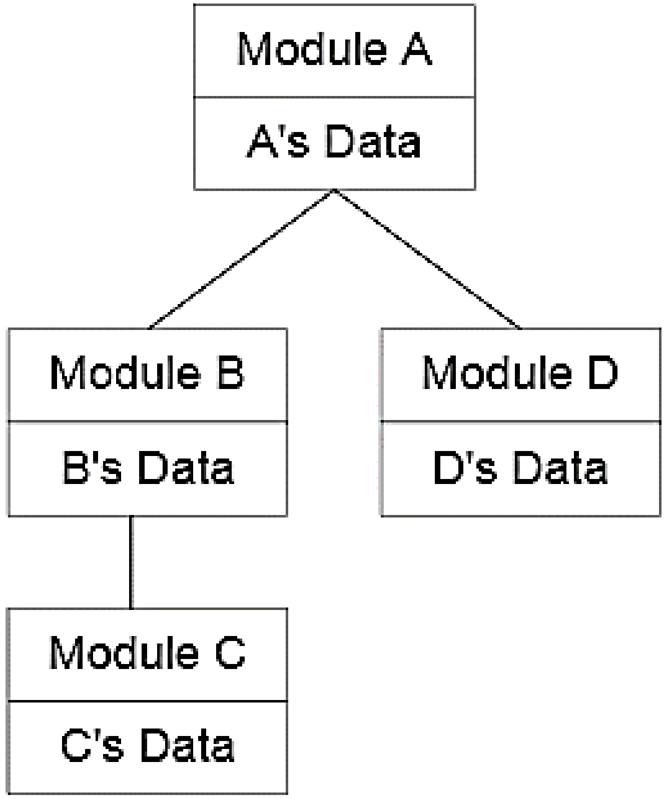
A Software should be Lesly coupled and highly cohesive.

COUPLING

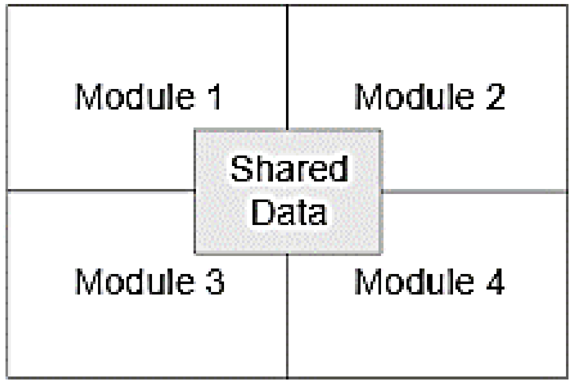

High Coupling


Low Coupling

COUPLING

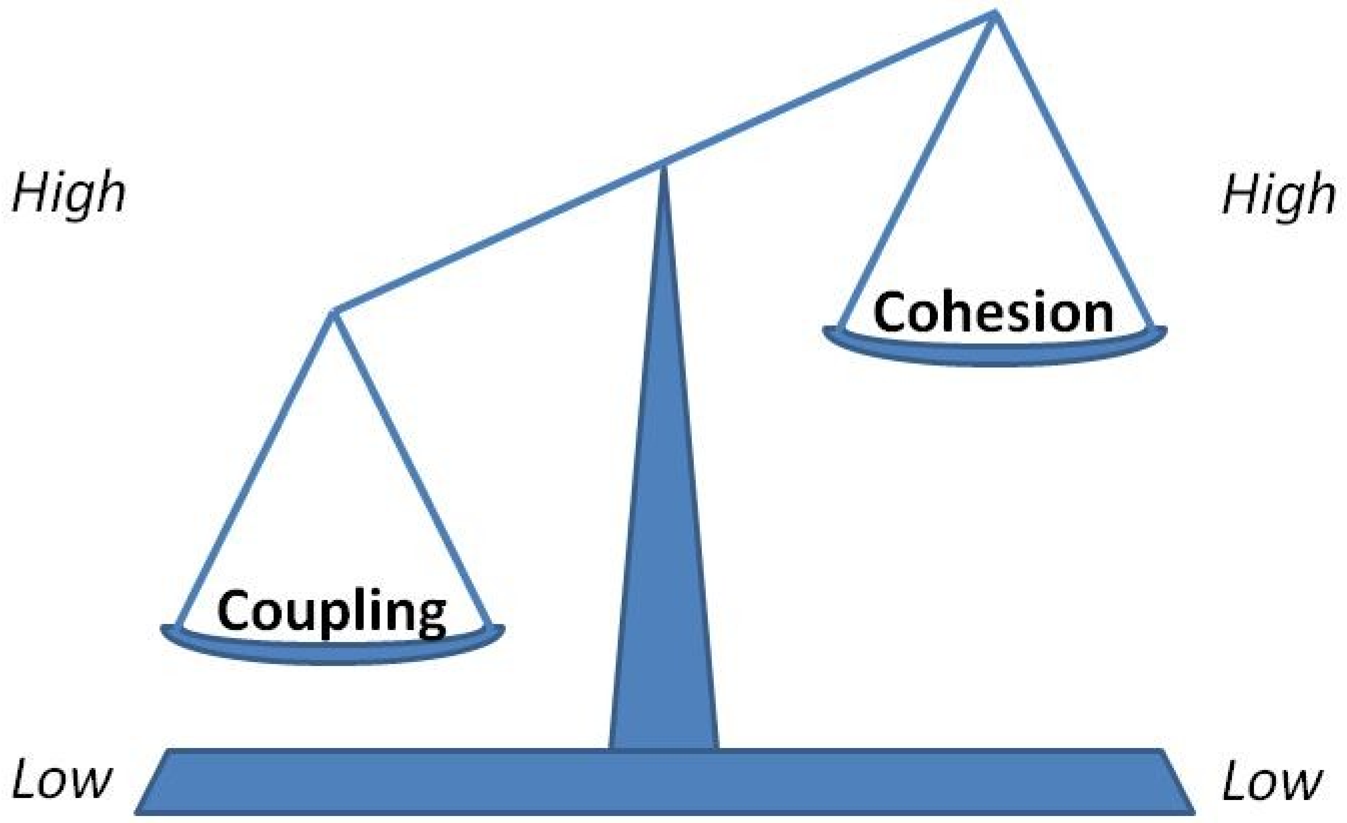


Low Coupling



High Coupling

RELATIONSHIP BETWEEN COUPLING AND COHESION



HAVE A GOOD DAY!