

# Behavioral Modeling



## LECTURE 6

# Key Ideas

- Behavioral models describe the internal dynamic aspects of an information system that supports business processes in an organization
- Key UML behavioral models are: sequence diagrams and behavioral state machines (state chart diagrams)

# Objectives

- Understand the rules and style guidelines for sequence diagrams and behavioral state machines.
- Understand the processes used to create sequence diagrams and behavioral state machines.
- Be able to create sequence diagrams and behavioral state machines.
- Understand the relationship between the behavioral models and the structural and functional models.

# BEHAVIORAL MODELS

# Purpose of Behavioral Models

- Show how objects collaborate to support each use case in the structural model
- Depicts internal view of the business process
- To show the effects of varied processes on the system

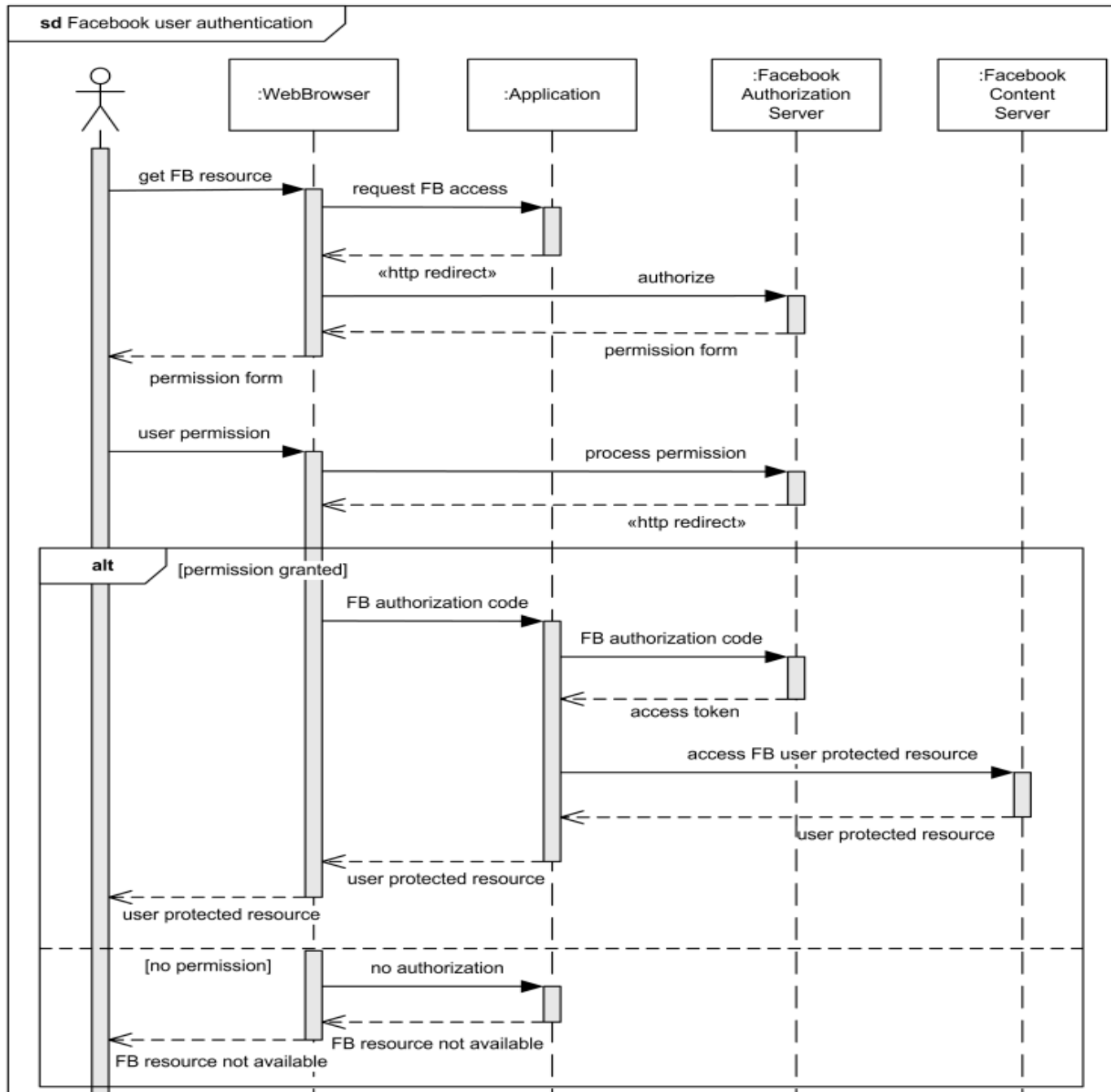
# Diagram Components

- **Objects**
  - Instance of a class
- **Operations**
  - Send and receive messages
- **Messages**
  - Tell object to execute a behavior

# Sequence Diagrams

- Illustrate the **objects** that participate in a use-case
- Show the **messages** that pass between objects **for a particular use-case**

# Sequence Diagrams Examples





# Sequence Diagrams...

- A Sequence diagram is a structured representation of behavior as a series of sequential steps over time.
- They are used to:
  - **Depict workflow**, Message passing and how elements in general cooperate over time to achieve a result
  - **Capture the flow of information and responsibility** throughout the system, early in analysis; Messages between elements eventually become **method calls** in the Class model
  - **Make explanatory models for Use Case scenarios**; by creating a Sequence diagram with an Actor and elements involved in the Use Case, you can model the sequence of steps the user and the system undertake to complete the required tasks

# Sequence Diagram...

- The main purpose of a sequence diagram is to define **event sequences** that **result in some desired outcome**.
- The focus is less on messages themselves and more on the **order in which messages occur**;
- The diagram conveys information along the **horizontal and vertical** dimensions:
- **Vertical dimension:** shows, top down, the time sequence of messages/calls as they occur, and
- **Horizontal dimension:** shows, left to right, the object instances that the messages are sent to.

# Construction of Sequence Diagram

- Sequence elements are arranged in a **horizontal sequence**, with **Messages passing back** and **forward** between elements
- Messages on a Sequence diagram can be of several types; the Messages can also be configured to reflect the **operations** and **properties** of the **source** and **target elements**
- An **Actor element** can be used to represent the **user initiating the flow** of events
- Stereotyped elements, such as **Boundary, Control and Entity**, can be used to illustrate **screens, controllers and database items**, respectively
- Each element has a **dashed stem** called a **Lifeline**, where that element exists and potentially takes part in the interactions



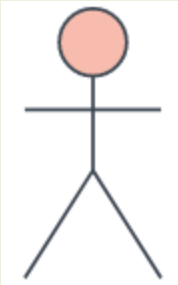
# Benefits of sequence diagrams

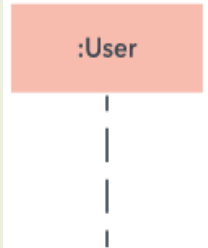
- Sequence diagrams can be useful reference diagrams for businesses and organizations. They are drawn to bring the following benefits:
  - Represent the details of a UML use case.
  - Model the logic of a sophisticated procedure, function, or operation.
  - See how tasks are moved between objects or components of a process.
  - Plan and understand the detailed functionality of an existing or future scenario.

# Benefits...

- Describe the **flow of messages, events**, actions between objects
- Show **concurrent processes** and **activations**
- Show **time sequences** that are not easily depicted in other diagrams
- Typically **used during analysis and design** to document and understand the logical flow of your system

# Basic symbols

Symbol	Name	Description
	Object symbol	This box shape represents a <b>class, or object</b> , in UML. They demonstrate how an object will behave in the context of the system. <b>Class attributes should not be listed in this shape.</b>
	Activation box	Symbolized by a rectangle shape, an activation box <b>represents the time needed</b> for an object to complete a task. <b>The longer the task</b> will take, the longer the activation box becomes.
	Actor symbol	Represented by a stick figure, actors are entities that are both <b>interactive</b> with and <b>external</b> to the system.



Lifeline symbol

A dashed vertical line that represents **the passage of time as it extends downward**. Lifelines **may begin with a labeled rectangle shape or an actor symbol**.



Synchronous message symbol

Represented by a solid line with a solid arrowhead. This symbol is used **when a sender must wait for a response** to a message before it continues. The diagram should show both the call and the reply.



Asynchronous message symbol

Represented by a solid line with a lined arrowhead. Asynchronous messages are those that **don't require a response before the sender continues**. Only the call should be included in the diagram.

# Lifeline

- A sequence diagram is made up of several lifeline notations that should be **arranged horizontally across the top of the diagram**.
- No two lifeline notations should overlap each other.
- **They represent different objects or parts that interact** with each other in the system during the sequence.

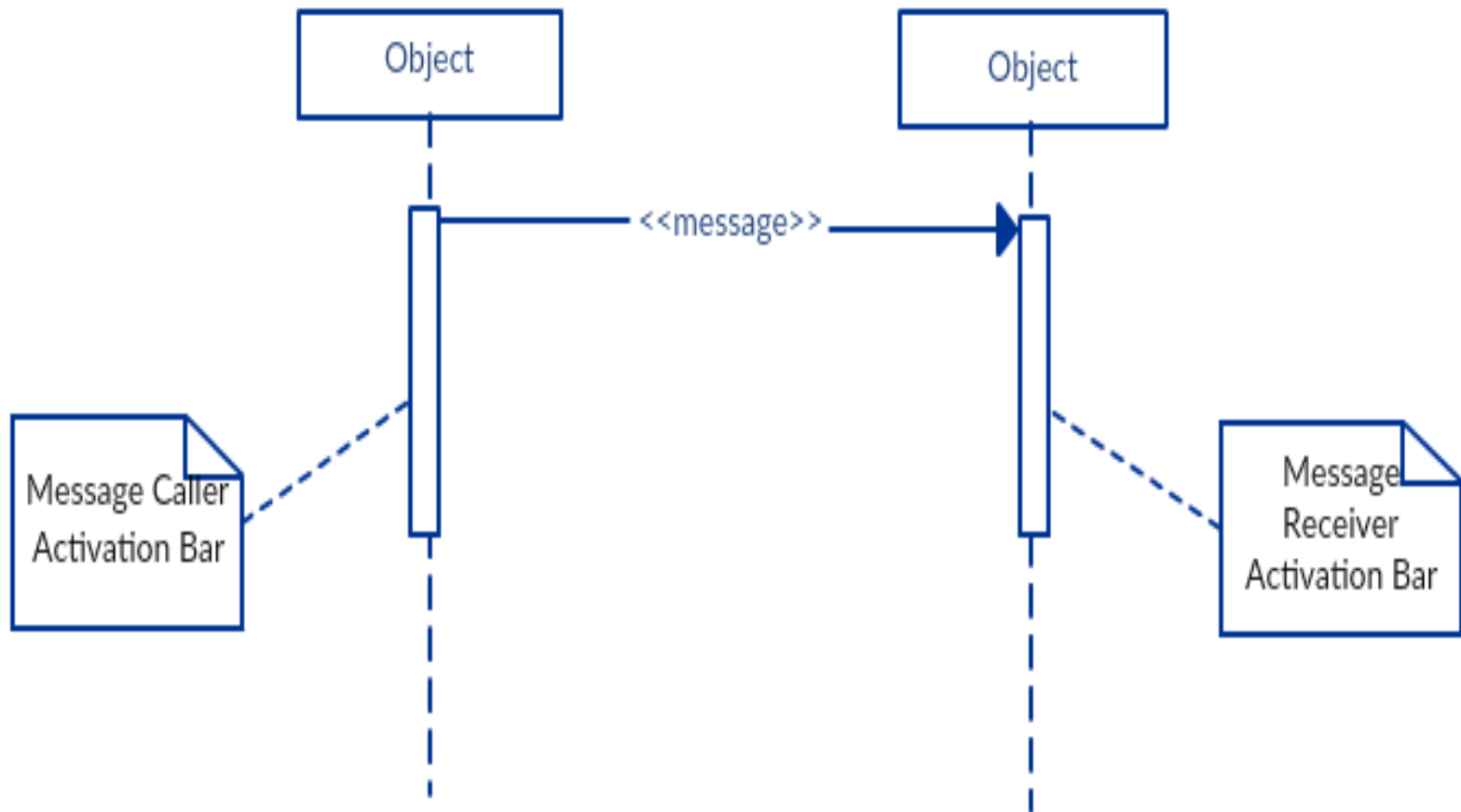
Name syntax: <objectname>:<classname>



# Activation Bars

- Activation bars are **boxes placed on the lifeline**. They are used to indicate that an object is active (or instantiated) during an interaction between two objects.
- The **length of the rectangle** indicates the **duration of the objects staying active**.
- Interaction between two objects occurs when one object sends a **message** to another.

# Activation Bars...

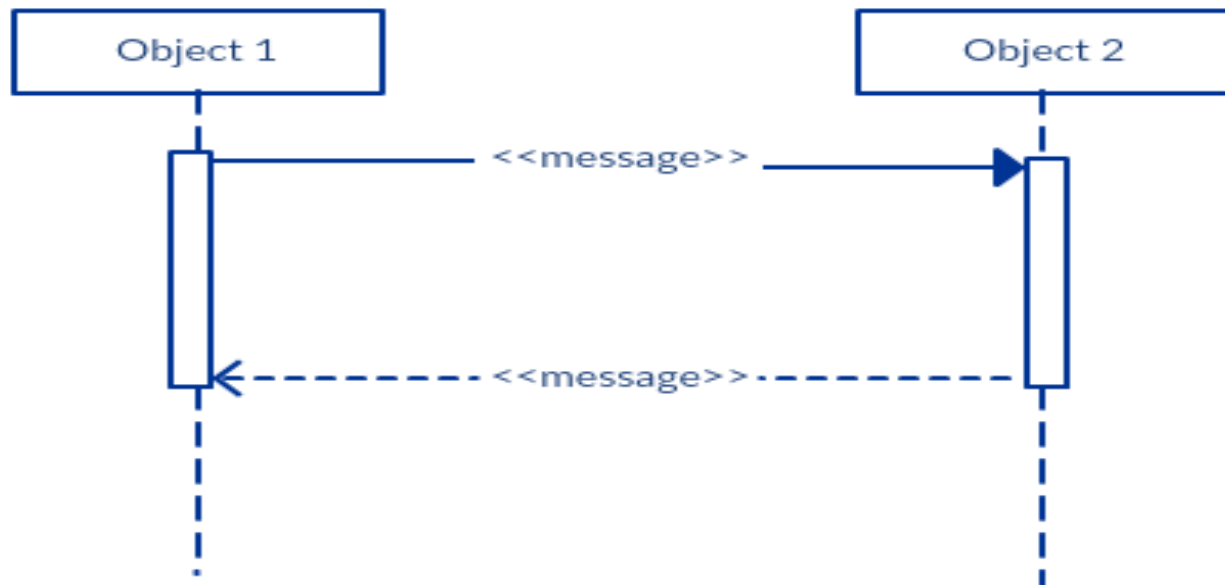


# Message Arrows

- An arrow from the **Message Caller** to the **Message Receiver** specifies a message in a sequence diagram.
- **A message can flow in any direction**; from left to right, right to left or back to the Message Caller itself.
- **Describe the message** being sent from one object to the other **on the arrow**, with different arrowheads you can indicate the type of message being sent or received.
- **Message format:**  
*message\_name (arguments): return\_type*

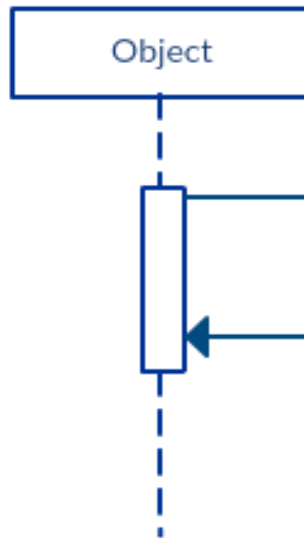
# ***Return message***

- A return message (optional) is used to indicate that the message receiver is done processing the message and is returning control over to the message caller.



# *Reflexive message*

- Is when an object sends a message to itself. It is indicated with a message arrow that starts and ends at the same lifeline.

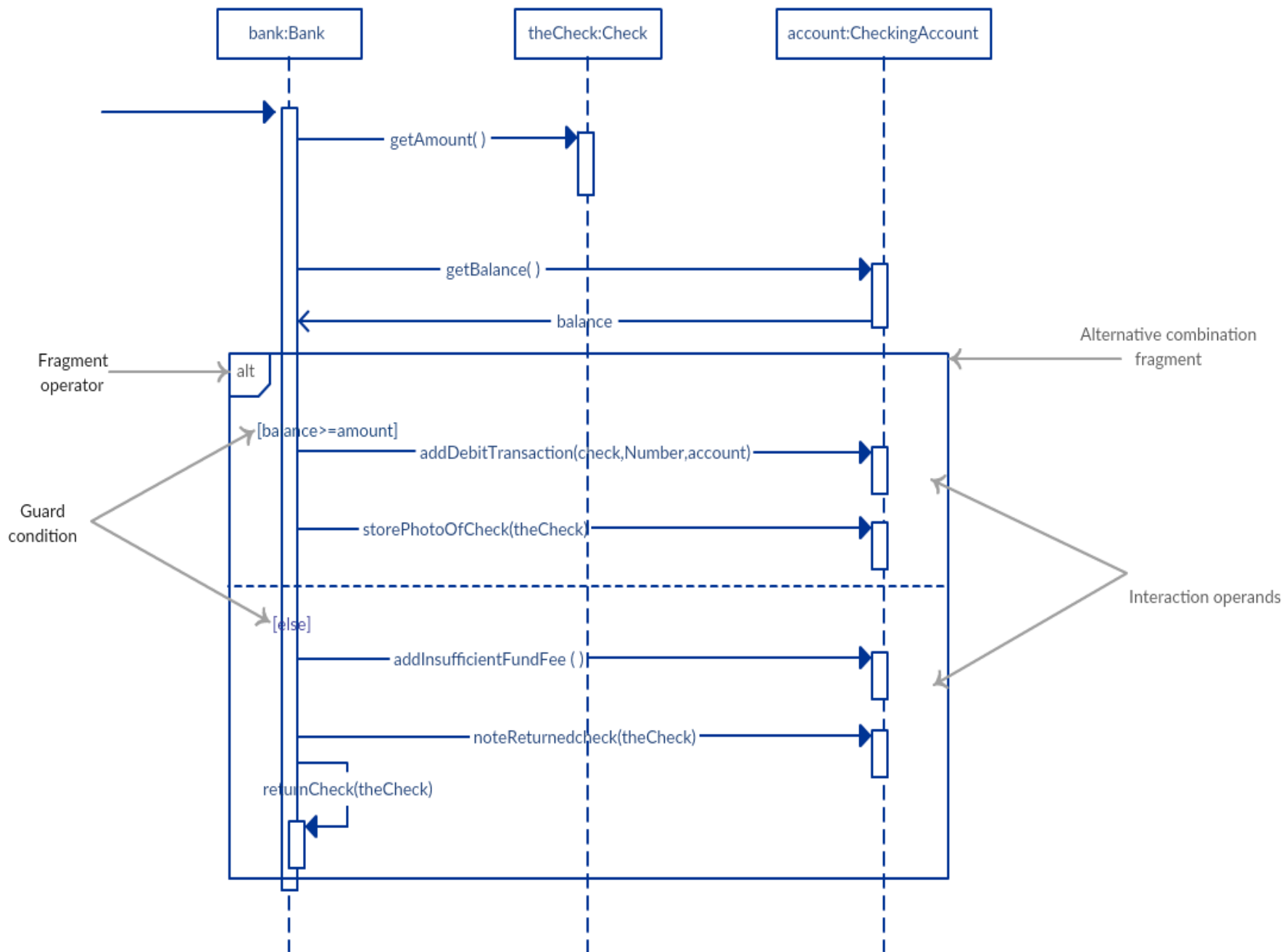


# Managing interactions with sequence fragments

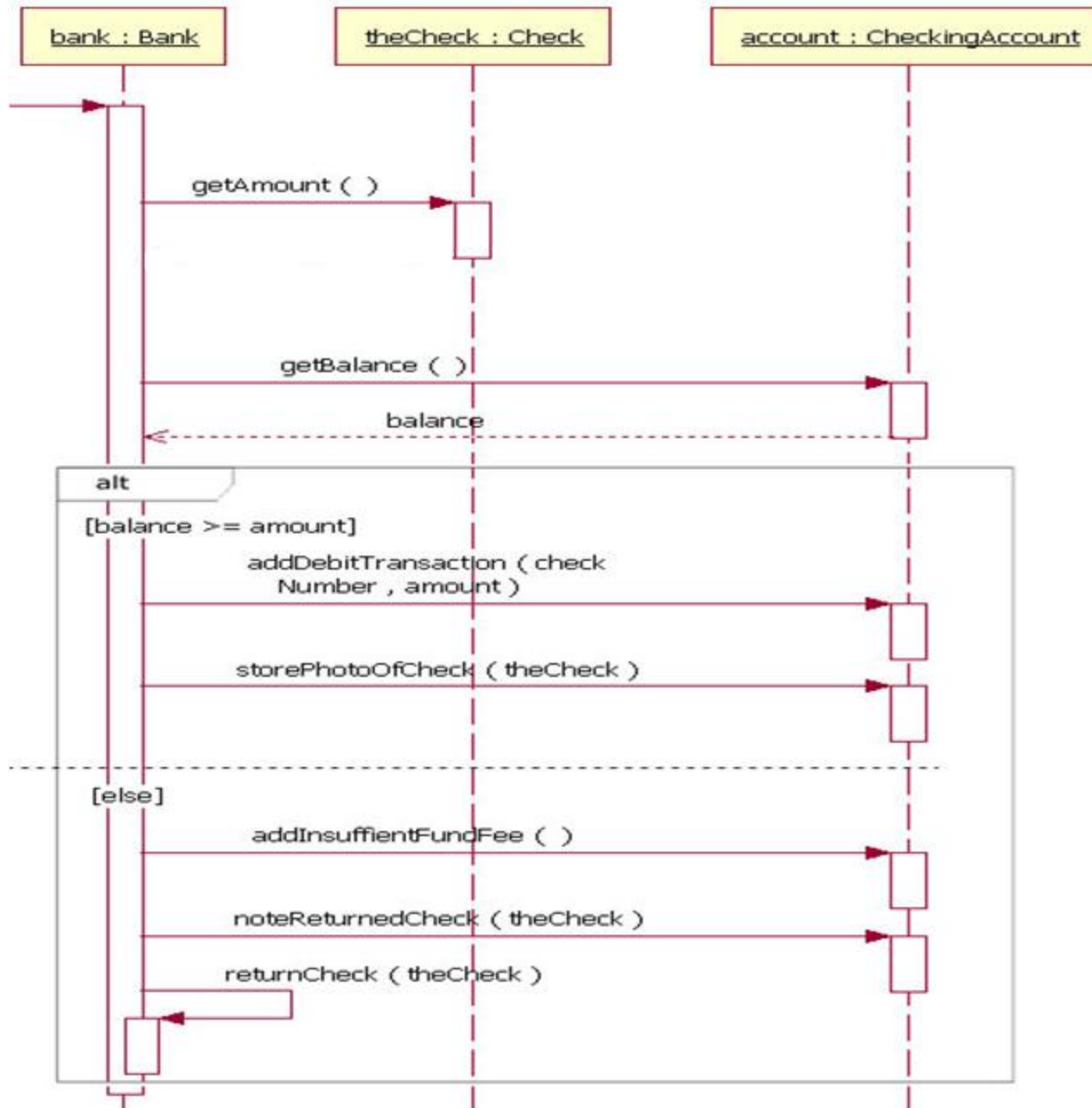
- A sequence **fragment** is presented as a **box** that frames a section of interactions between objects
- It is used to show interactions such as **alternative flows** and **loops** in a more structured way.

# *Alternatives*

- The **alternative combination fragment** is used when a choice needs to be made between two or more message sequences.
- It models the “**if then else**” logic.
- The **alternative fragment** is represented by a **frame**; it is specified by mentioning ‘**alt**’ inside the frame’s name box



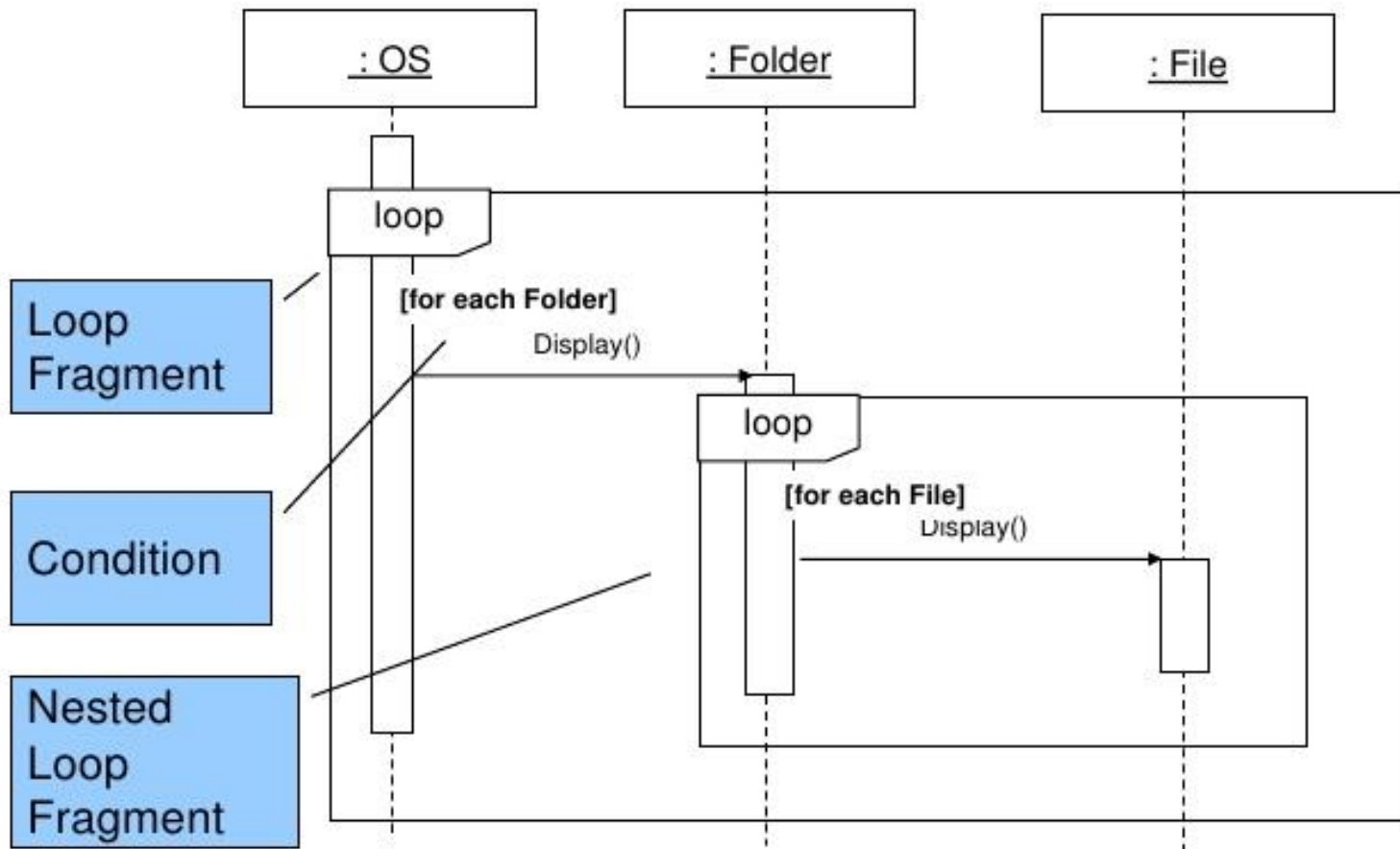




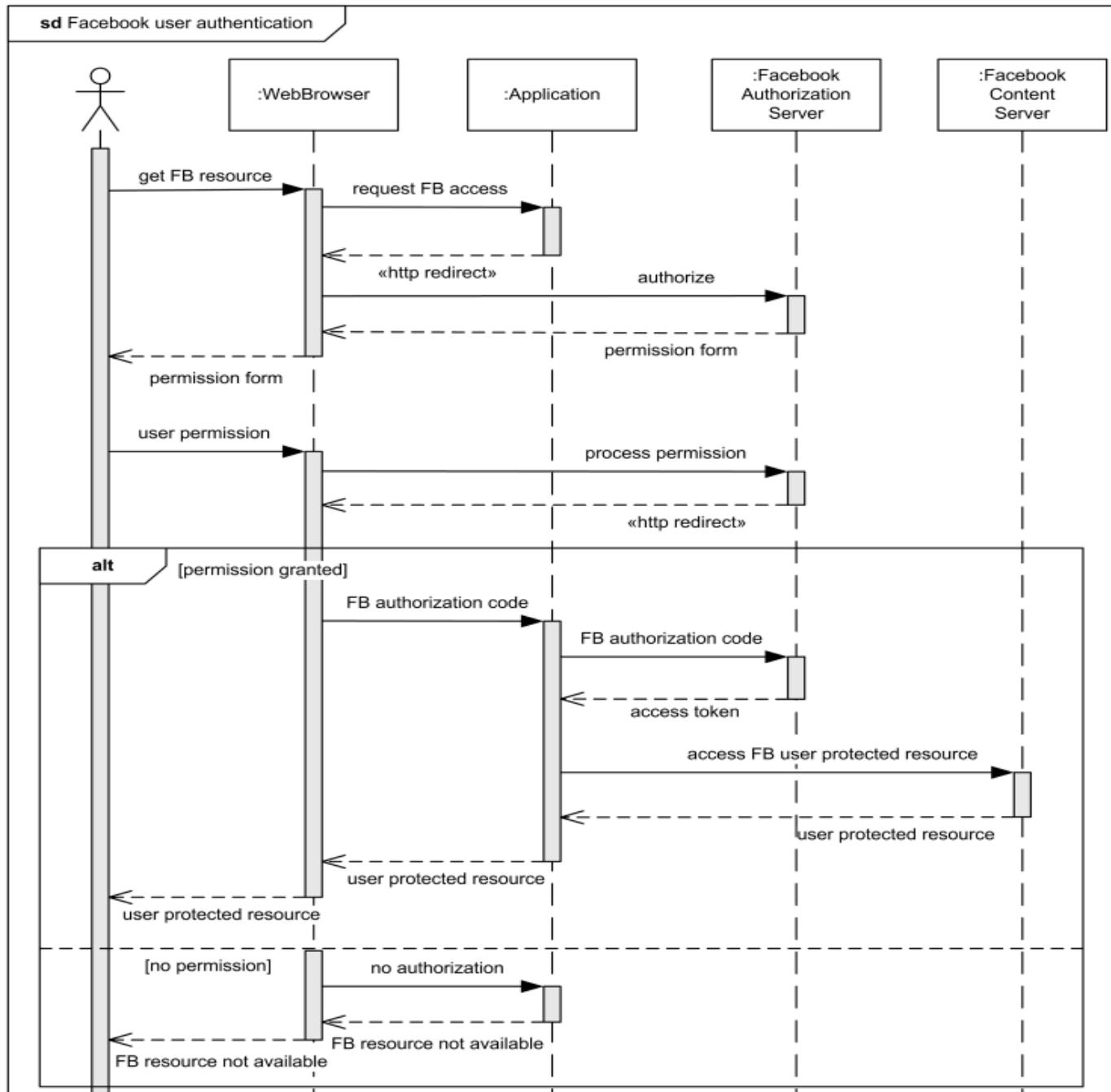
# *Loops*

- Loop fragment is used to represent a repetitive sequence.
- Place the words 'loop' in the name box and the guard condition near the top left corner of the frame.

# Loops

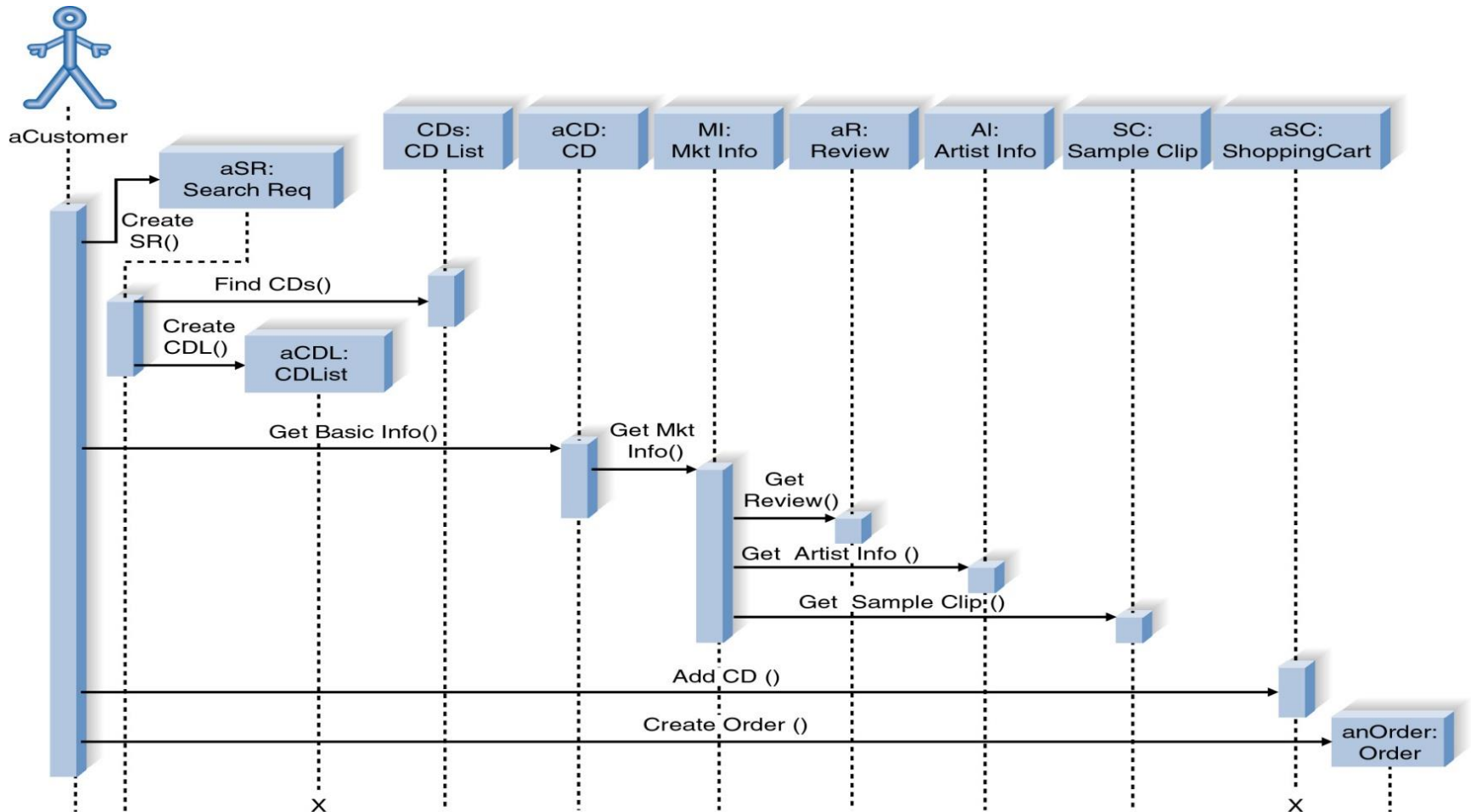


# Sequence Diagrams Examples



- **Summary:** Facebook uses **OAuth 2.0** protocol framework which enables web application (called "client") to request access to resources controlled by the FB user and hosted by the Facebook server. Instead of using the Facebook user credentials to access protected resources, the web application obtains an access token.

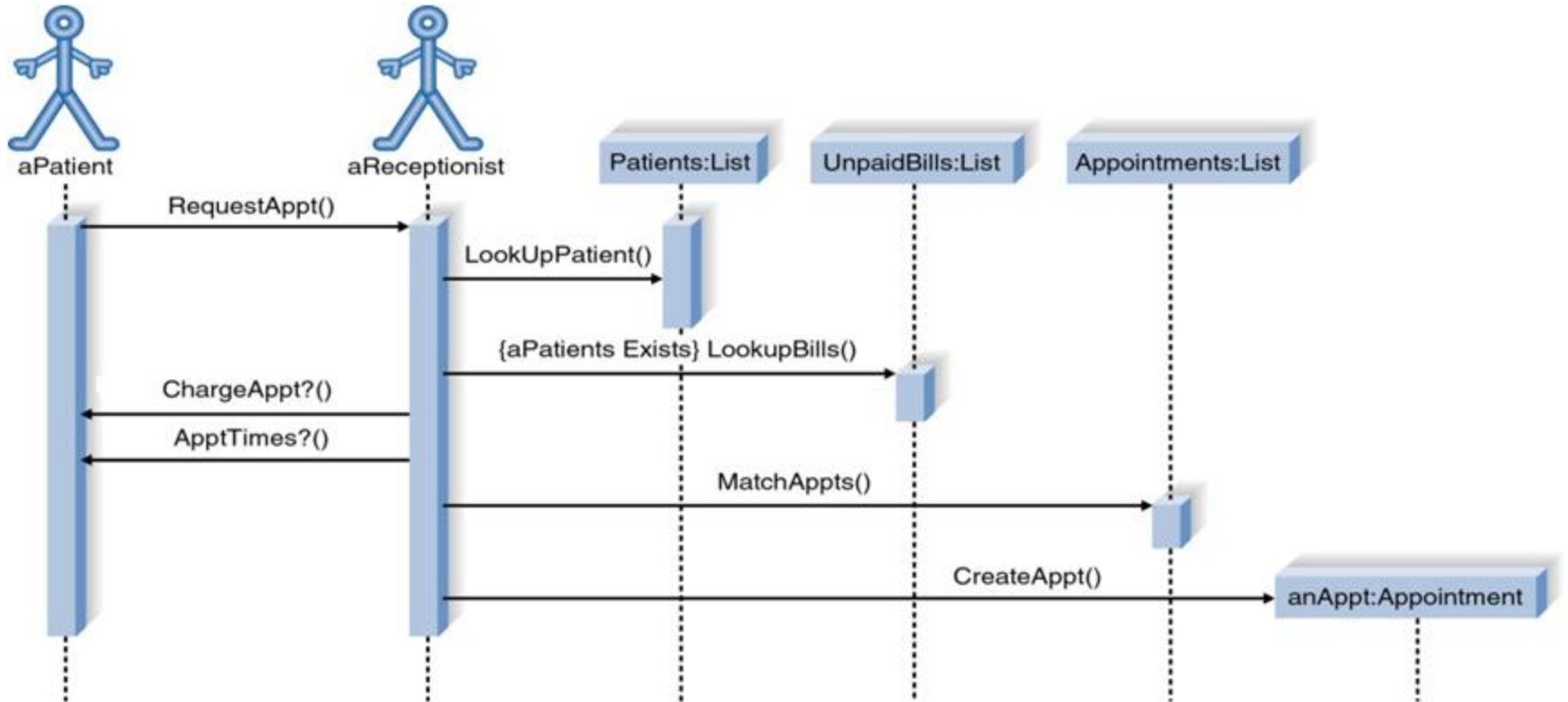
# Sequence Diagram-CD Selections



### CD Selection-Flow of Events:

1. **Customer** submits a search request to the system.
2. The system provides the **customer** a list of recommended CDs.
3. The **customer** chooses one of the CDs to find additional information.
4. The system provides the **customer** with basic information & CD Reviews
5. The **customer** calls the shopping cart **use case**.
6. The **customer** executes the place order **checkout use case**.
7. The **customer** leaves the website.

# Sequence Diagram Make Appointment

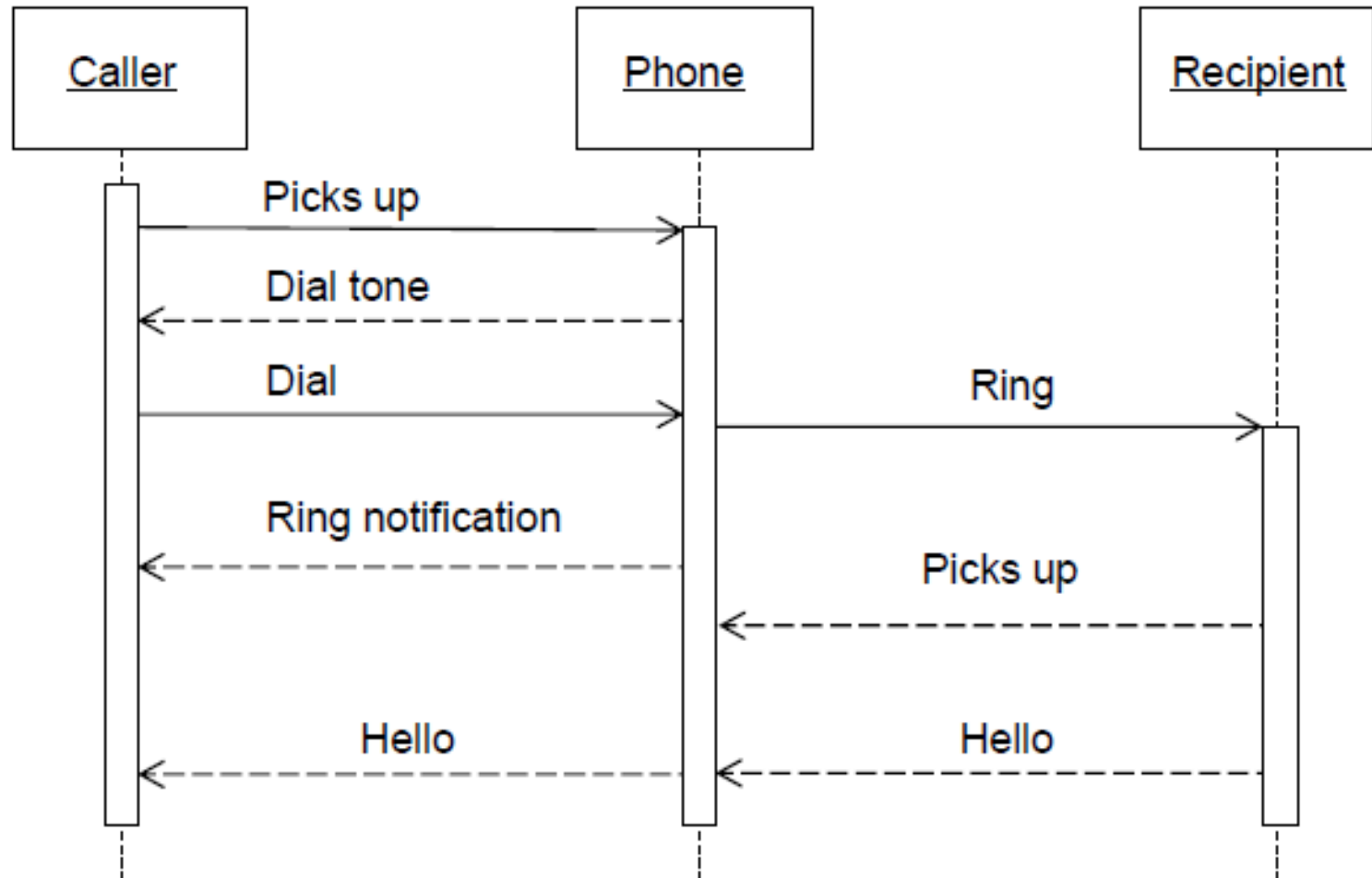


Dennis: SAD

Fig: 8-1 W-30 100% of size  
Fine Line Illustrations (516) 501-0400



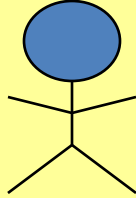



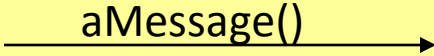

# Sequence Diagram (make a phone call)



# Building a Sequence Diagram

1. Determine the **context** of the sequence diagram
2. Identify the participating **objects**
3. Set the **lifeline for each object**
4. Add **messages**
5. Place the **execution occurrence (focus of control)** on each object's lifeline
6. **Validate** the sequence diagram

# Sequence Diagram Syntax

ACTOR	
OBJECT	
LIFELINE	
EXECUTION OCCURRENCE ( FOCUS OF CONTROL)	
MESSAGE	
OBJECT DESTRUCTION	



# Behavioral State Machines (State Chart Diagrams)

- The behavioral state machine is a **dynamic** model that shows different **states** of the object and what **events** cause the object to change from one state to another, along with its responses and actions.
- **The State Diagram.** It can be used either for **analysis** (if our goal is to record how behavior actually emerge in a system we are modeling) or for **design** (if our goal is to describe a behavior we want to produce.)

# Behavioral State Machines (State Chart Diagrams)...

## Note:

**UML 1** referred to these as **state chart diagrams**;

**UML 2** calls them **State Machine Diagrams**

- sometimes shortened to **State Diagrams**.
- UML state charts show:
  1. **States: simple** and **composite**
  2. **Transitions:** including **events** and **actions**.

# State Machines...

- Objects have behaviors and states.
- State chart Diagrams shows the possible states of the object and the transitions that cause a change in state.

# State Machines...



A state machine diagram for invoices



# Purpose of State chart Diagrams

- The most important purpose of State chart diagram is **to model lifetime of an object from creation to termination.**

# Events

- Internal or External Events trigger some activity that changes the state of the system
- Software design involves examining events in a State Machine and considering how those events will be supported by system objects

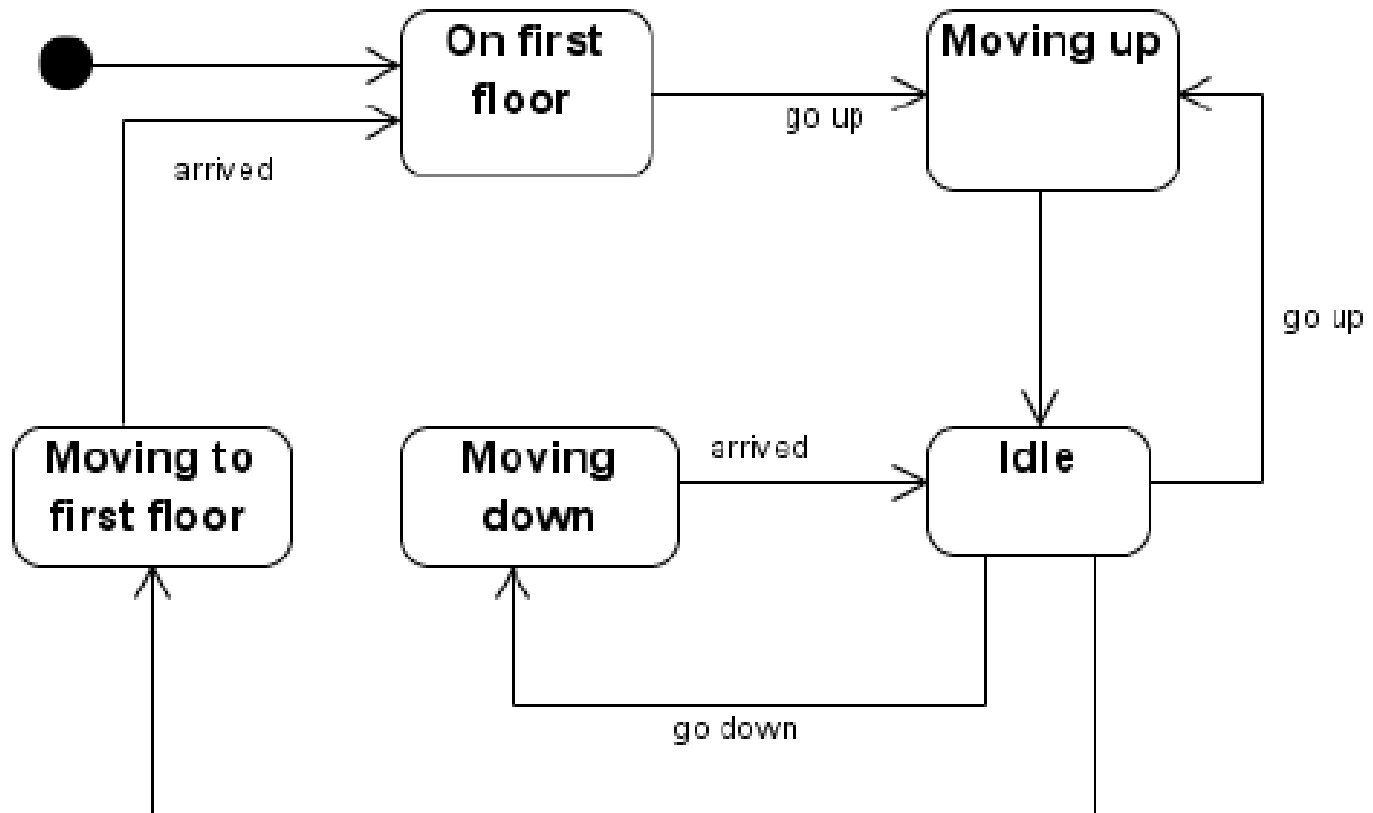
# States

- States are defined as a condition in which an object exists and it changes when an event is triggered.
- Objects (or Systems) can be viewed as moving from state to state
- A point in the lifecycle of a model element that satisfies some condition, where some particular action is being performed or where some event is waited.

# States...

- Stable state represents a condition in which an object may exist for some identifiable period of time.
- When an event occurs, the object may move from state to state (a transition).
- In reaction to an event or a state change, the object may respond by dispatching an action.

# States



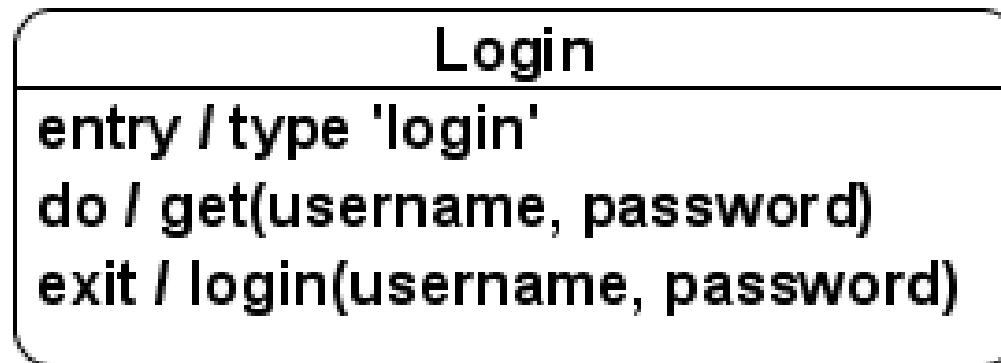
A state machine diagram for a lift

# Actions

- States can trigger actions
- States can have a second compartment that contains actions or activities performed while an entity is in a given state
- Action is an atomic execution and therefore completes without interruption
- Five triggers for actions: On Entry, Do, On Event, On Exit, and Include
- An activity captures complex behavior that may run for a long duration.

# Actions...

- e.g. A state called Login
- Actions are performed on entry, on exit and while in the state.









# How to Draw a State chart Diagram?

- Identify the important objects to be analyzed.
- Identify the states.
- Identify the events.

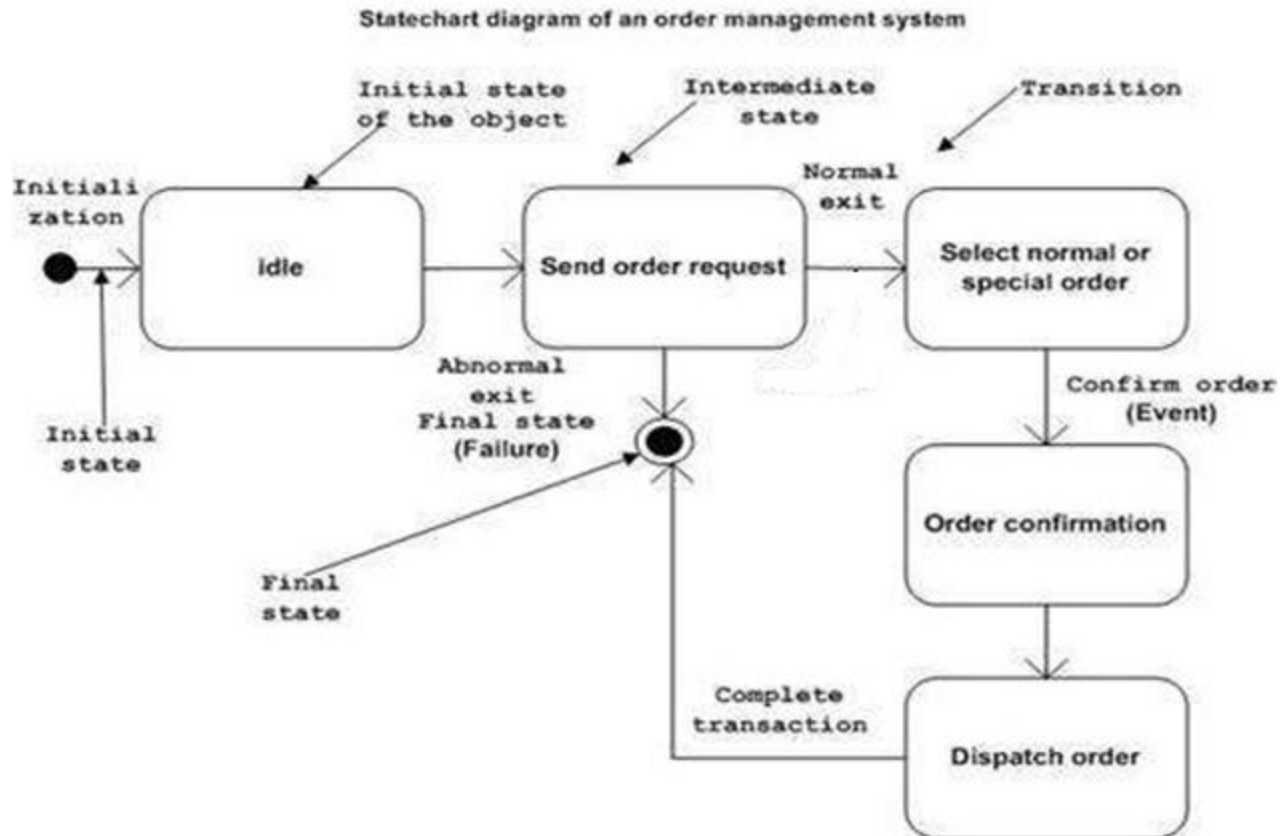


# Behavioral State Machine Diagram Syntax

A STATE	
AN INITIAL STATE	
A FINAL STATE	
AN EVENT	
A TRANSITION	
A Frame	

# Example

- State chart diagram where the **state of Order object** is analyzed



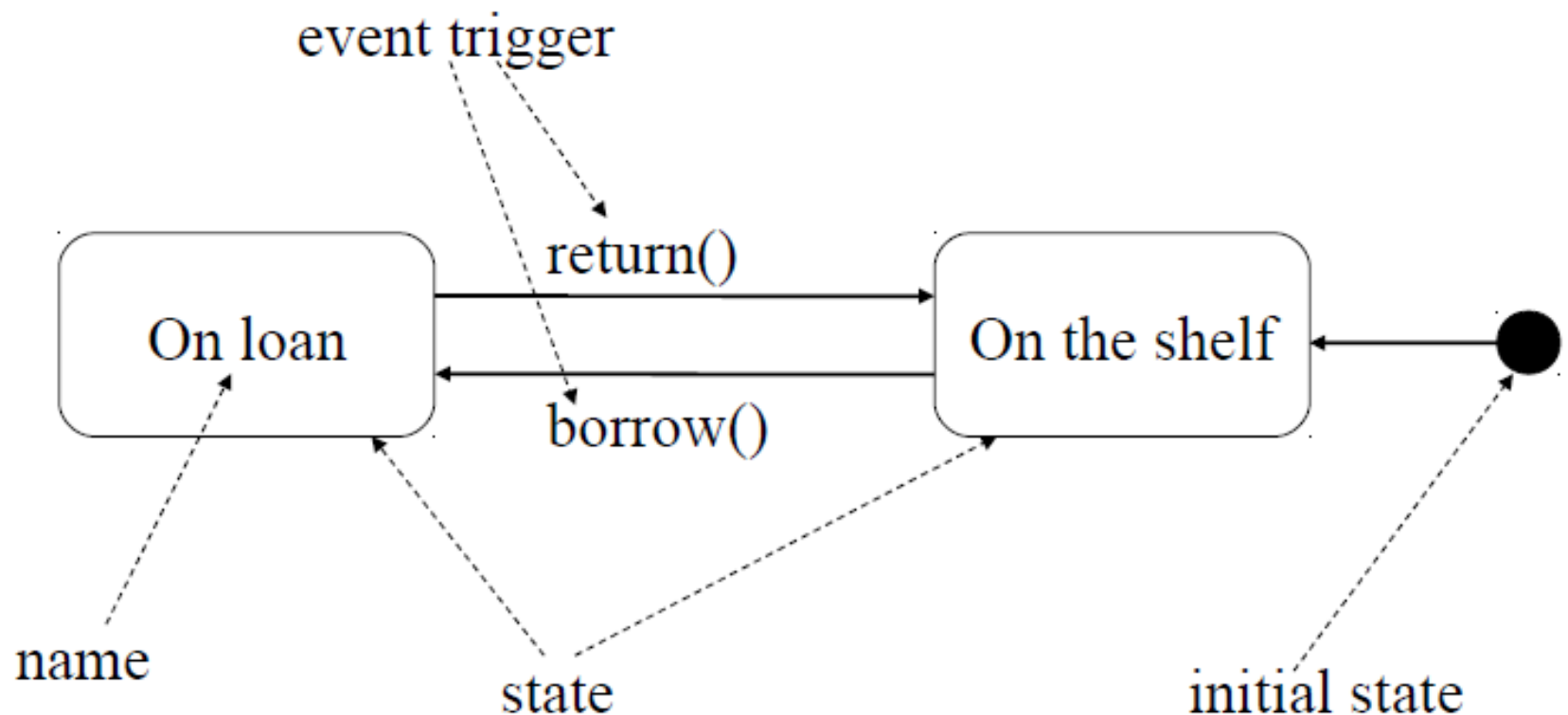
# When to use State charts?

- When you REALLY need to know the states of an object – because it is vital to the success of the application you are working on.

# Simple and Composite States

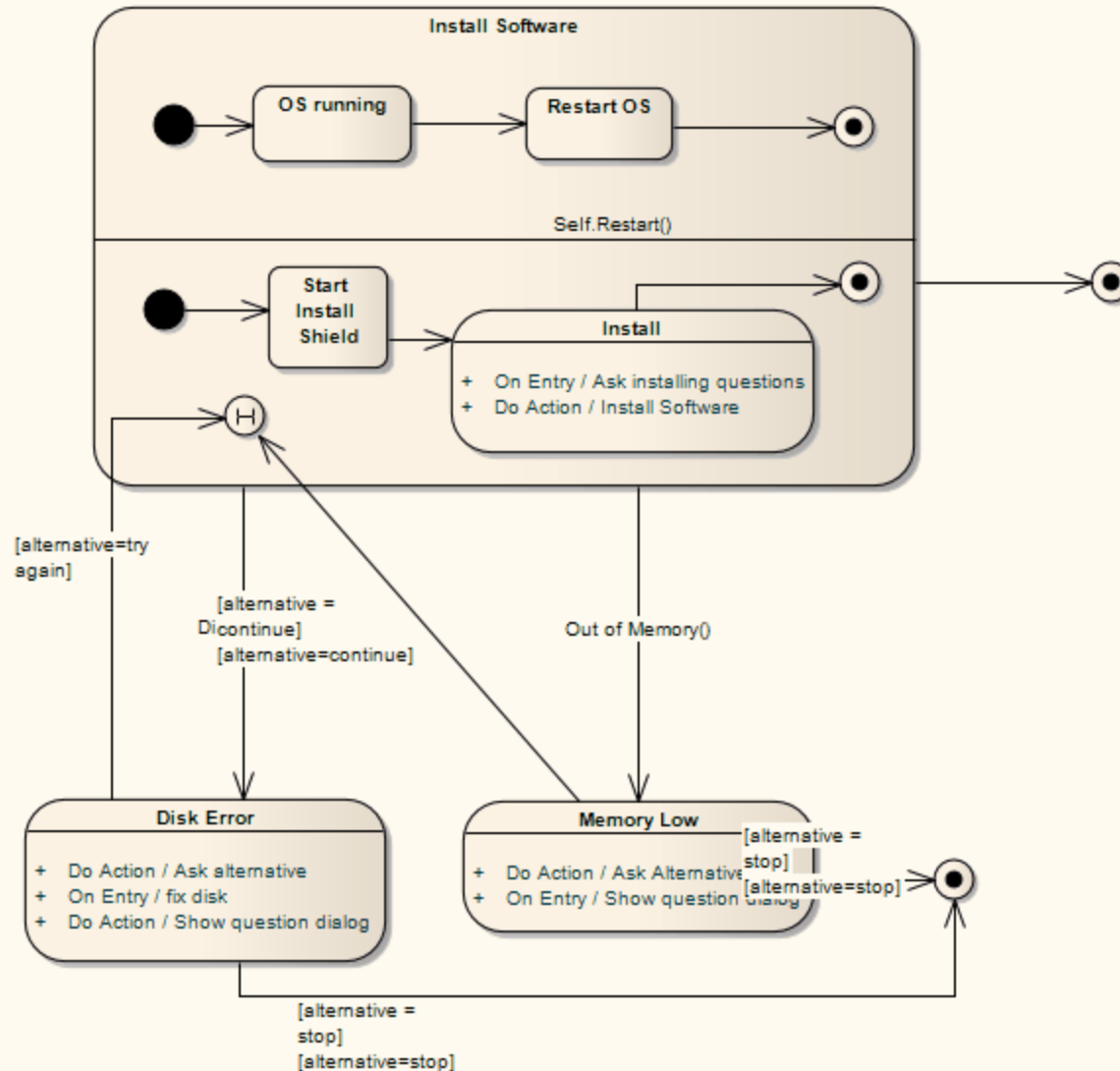
- **Simple States** - simplest of all states, they have no sub states
- **Composite States** - have one or more regions for sub states
- **Submachine States** - semantically equivalent to composite states, submachine states have sub states that are contained within a sub state machine
- **History State** - indicated by a circle with an H inside it. Allows re-entering of a composite state at the point which it was last left

# Simple State chart example

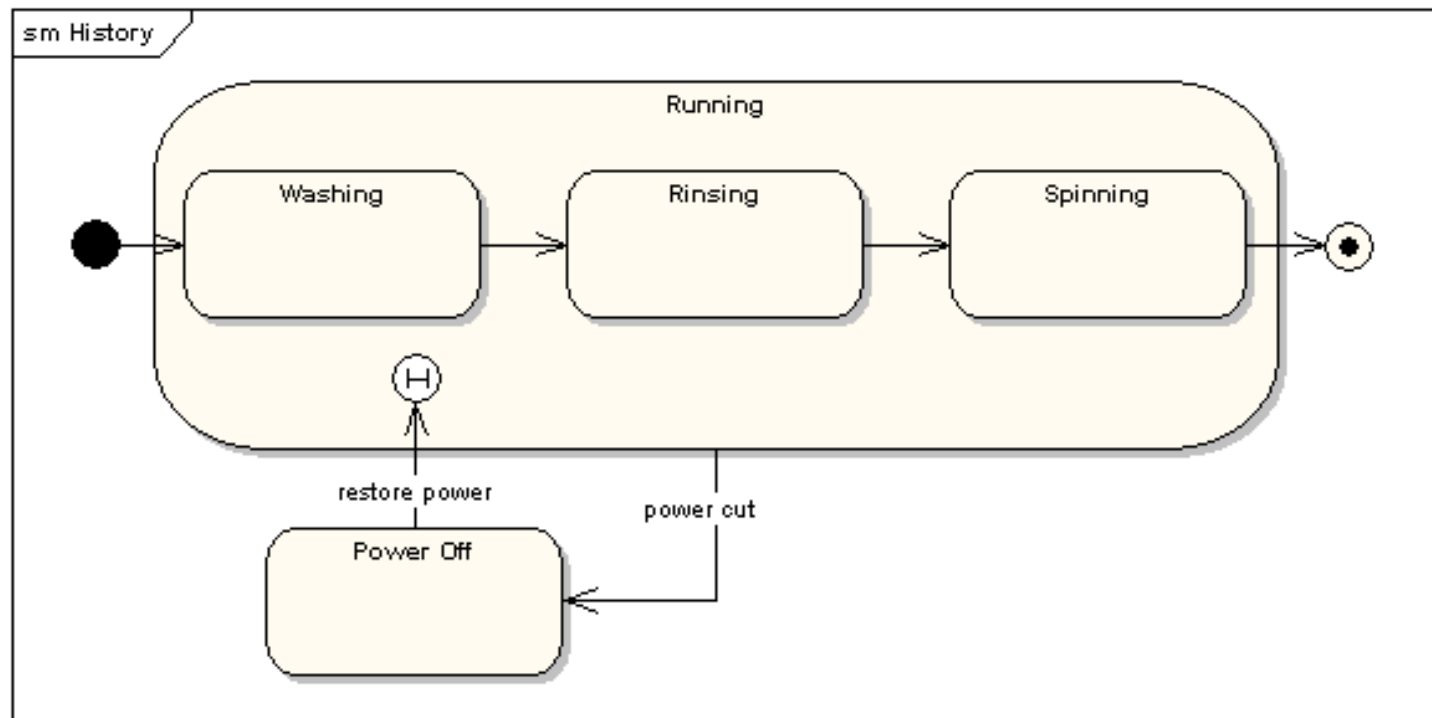


Describing a Copy of a Book object

# Composite State



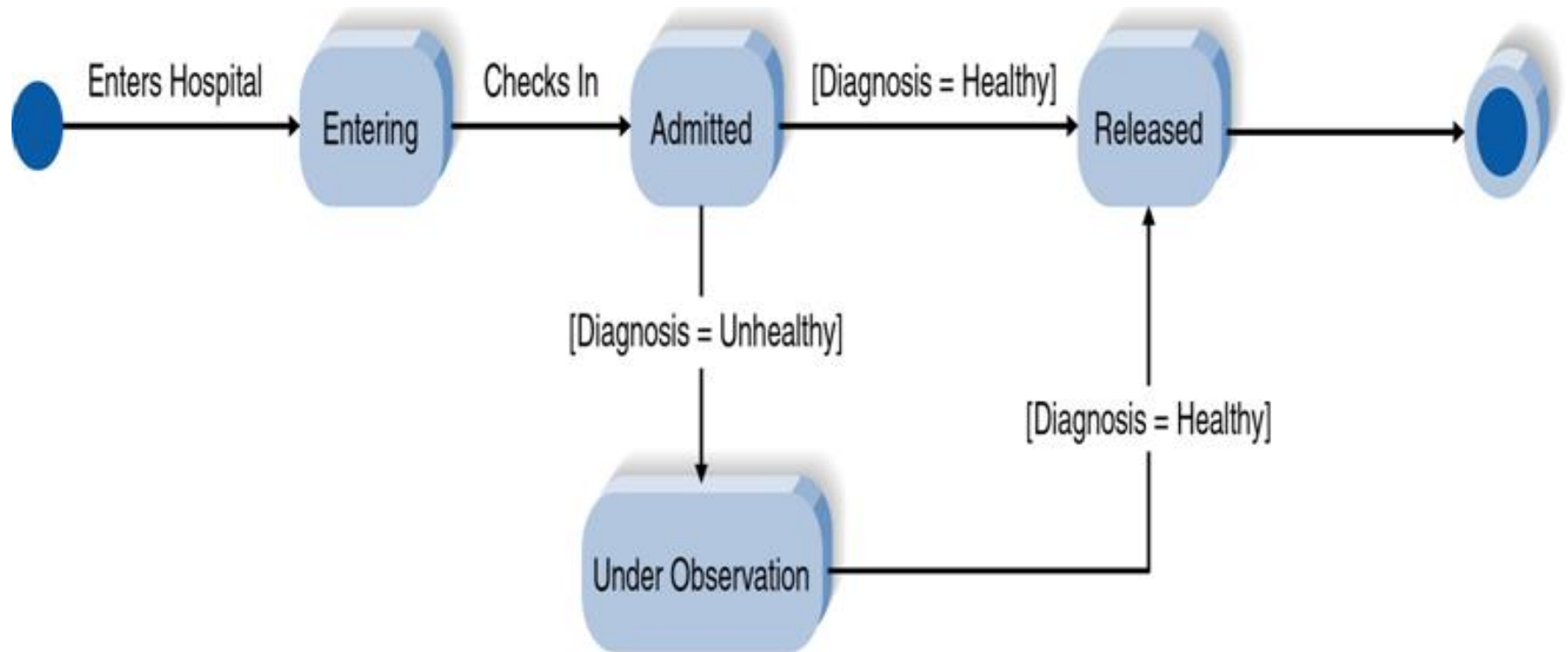
- In this state machine, when a washing machine is running, it will progress from "Washing" through "Rinsing" to "Spinning".
- If there is a power cut, the washing machine will stop running and will go to the "Power Off" state.
- Then when the power is restored, the Running state is entered at the **"History State"** symbol meaning that it **should resume where it last left-off**.



extra



# Example Behavioral State Machine Diagram



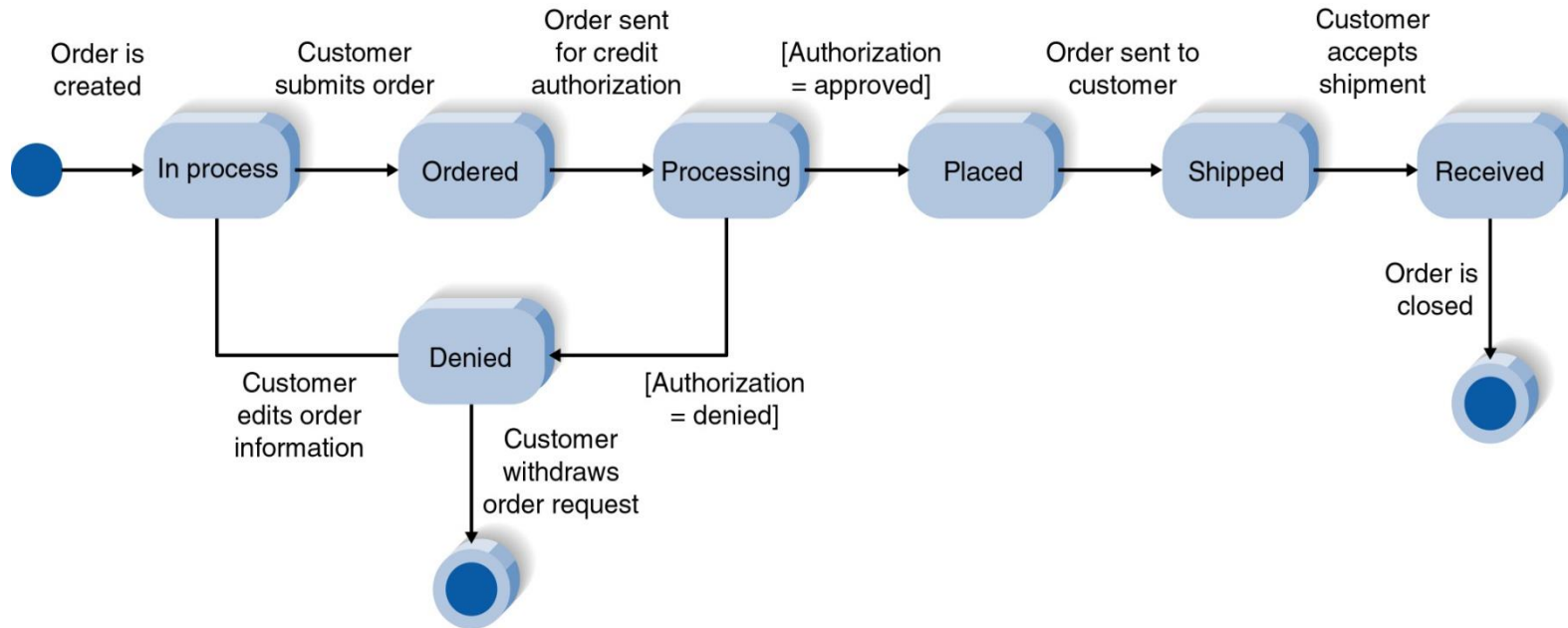
# Note

- Note the difference between an interaction diagram and a state diagram
- Interaction diagram deals with how several objects work together to accomplish one use case;

While...

- State diagram deals with internal state of one object, which may be affected by multiple use cases.

# CD Selections



Dennis: SAD

Fig: 8-15 W-38a 100% of size  
Fine Line Illustrations (516) 501-0400

# Summary

- *Sequence diagrams* illustrate the classes/objects that participate in a use case and the messages that pass between them.
- *Behavioral State Machine diagrams* show the different states that a single class/object passes through in response to events.

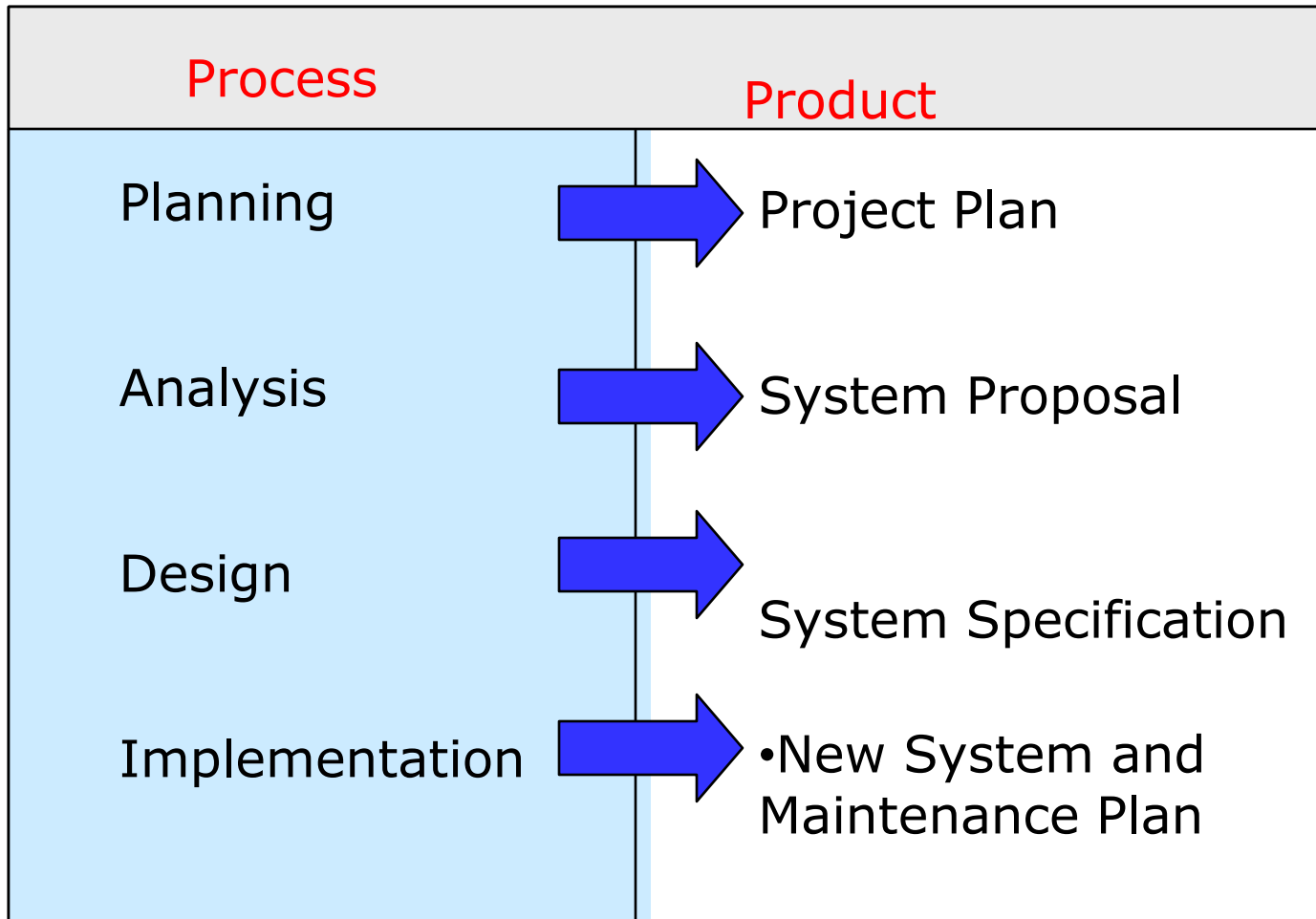
# SYSTEM IMPLEMENTATION

## Module 7

# UML Diagrams & SDLC

- Different types of **UML diagrams**, includes **class, activity, sequence, use case, statechart e.t.c**
- The diagrams are used in the analysis and design phases of the SDLC

# Processes and Deliverables (SDLC)



---

# System Implementation

- There are six major activities during system implementation which are **coding, testing, installation, documentation, training** and **support**.
- The purpose of these activities is to convert physical design into working software and hardware.



# System Implementation

Coding

Coding and testing can proceed in parallel

Testing

Installation



# Coding...

- **Once the design is complete**, most of the major decisions about the system have been made.
- The goal of the coding phase is to **translate the design of the system into code** in a given programming language.
- For a given design, the aim of this phase is to implement the design in the best possible manner.
- The **coding phase affects both testing and maintenance** profoundly.
- A well written code reduces the testing and maintenance effort.

# Coding...

- **Structured programming** is an important concept that helps the understandability of programs.
- **Structured programming** arranges the control flow in the program. i.e. program text is organized as a sequence of statements, and during execution, the statements are executed in the sequence in the program.
- **Structured programming** uses few single-entry-single-exit constructs. i.e. selection (if-then-else), and iteration (while - do, repeat - until etc).

With **constructs** it is possible to create a program as sequence of single - entry - single - exit constructs. This helps for verifying the code in Some methods such as **static system testing**.

# Coding...

- In coding phase, the entire system is not tested together.
- Different modules are tested separately. This testing of modules is called "unit testing".
- Coding phase is often referred to as "coding and unit testing".
- **The output** of this phase is the verified and unit tested code.

# Testing

- **Testing** is the major quality control measure employed during software development. Its basic function is to **detect errors in the software.**
- During analysis and design phase, the output is a document that is usually textual and non-executable.
- **After the coding phase**, computer programs are available that can be executed for **testing phase.**

# Testing...

- **Testing** not only has to **uncover errors** introduced during coding, but also errors introduced during the previous phases.

# Types of System Testing

1. **Static testing** - code being tested is not executed.
2. **Dynamic testing** - code being tested involves execution.
3. **Automated testing** – conducted using a computer
4. **Manual testing** - people conducts the test.

# Types of Testing...

## **Inspections (static, manual)**

testing technique where participants examine program code for expected language-specific errors.

## **Walkthroughs (dynamic, manual)**

a peer group review of any product created during the systems development process, including code



# Types of Testing...

## **Desk Checking ( dynamic, manual)**

a testing technique in which the program code is sequentially executed manually by the reviewer

## **Syntax checking (static, automated)**

### **a. Unit Test ( dynamic, automated)**

Each module is tested alone in an attempt to discover any errors in its code

### **b. Integration Test ( dynamic, automated)**

Brings together all of the modules that a program comprises for testing purposes

# Types of Testing...

## **System Test ( dynamic, automated)**

Brings together all of the programs that a system comprises for testing purposes

# User Acceptance Test

- User acceptance testing (UAT) is **the final phase of the software testing process.**
- UAT is one of the critical software project procedures that must occur **before newly developed software is rolled out** to the market.

# User Acceptance Test...

There are two types of UAT:

## **1. Alpha testing:**

User test a completed information system using simulated data

## **2. Beta testing:**

User test a completed information system using real data in the real user environment

# Installation

- A process for organization change over from the **current information system to a new one.**

## **Four approaches of installation:**

- Direct Installation
  - Parallel Installation
  - Single-location Installation
  - Phased Installation

# Direct Installation

- The organization switches off the old system and switches on the new one.
- This is probably the most straightforward method but is also probably the uncertain.

# Parallel Installation

- The organisation runs both **old and new system** in parallel for a time.
- Once the organisation is certain that new **system is working properly and staff are ready to begin using it**, will make the decision to completely change over.
- **During a quiet period** (at night or weekend) data is fully transferred from the old system which is then shut down.

# Single-location Installation

- New system is installed and tested in a small number of departments or branches.
- They then use the system and report their feedback and any issues to the analyst.
- Once the organisation is confident that the system is working as expected, it will be rolled out across the whole organisation.



# Phased Installation

- The old system is still running but parts of the new system or modules are brought in.
- Once any problems are smoothed out with the new modules then extra modules will be introduced.

# Documentation

## **System Documentation:**

- Detailed information about a system's design specifications, its internal workings, and its functionality

## **User Documentation:**

- Written or other visual information about the system, how it works, and how to use it

# Documentation

## **Internal documentation:**

System documentation that is part of the program source code

## **External documentation:**

System documentation that includes the outcome of diagramming techniques such as UML, E – R e.t.c

# Training and Support

Type of training needed varies by system type and user skill.

- Possible topics to be trained :
  - Use of the system
  - Information System concepts
  - System management
  - System installation

# Types of Training

- Several training methods can be depending on the requirements:
  - Traditional instructor-led classroom training
  - E-learning/distance learning
  - Blended learning (instructor plus e-learning)

# Support

**Support** means providing ongoing educational and problem-solving assistance to information system users.

Support is extremely vital to users.

Providing support can be expensive and time-consuming.

# Automating Support

Automated support can cut the costs of providing support at user site.

## **Example of automated support:**

- Internet-based online support forums
- On-demand fax/phone
- Knowledge bases (e.g. FAQ)

# Help Desk Support

- This is a **centralized point of contact** for all user inquiries and problems about a particular information system or for all users in a particular department.



# PRESENTATION (10 Marks)

- Using the same groups used for previous group assignment, work on the following **mini project** activity and prepare a 10 minutes presentation.
- Discussion among members on how to conduct this activity is highly encouraged.
- Each member will be graded individually based on his/her participation to the project and presentation.
- Presentations will be conducted on 5<sup>th</sup> February (for Stream B) and on 7<sup>th</sup> February (for Evening)

# MINI PROJECT - MAJOR TASKS

1. Think of any unique system that is useful and can solve community problems.
2. Apply the knowledge gained from this course to model your system. Simply you should be able to describe its design and analyze it using:
  - i. Functional Modeling (Use Case and Use Case Diagram)
  - ii. Functional Modeling (Activity Diagram)
  - iii. Structural Modeling (Class Diagram or Object Diagram)
  - iv. Behavioral Modeling (Sequence Diagram)
  - v. Behavioral Modeling (Behavioral State Machines)