Software Design Specifications Document

- **Project Title**

By

Student Name 1 Student ID

Student Name 2 Student ID

Supervisor:

Supervisor Name

Co-Supervisor (if any):

Co-Supervisor Name

Bachelor of Science in Computer Science (20xx-20xx)

Department of Computer Science
NUST Balochistan Campus
National University of Sciences & Technology

• Table of Contents

- **List of Tables**

- **List of Sequence Diagrams**

- **List of Activity Diagrams**

[1].

[1] **Note:** Refer to Appendix C, Section 1 about how to add separate caption labels

Revision History

Name	Date	Reason for changes	Version

Application Evaluation History

Comments (by committee) ***include the ones given at scope time both in doc and presentation**	**Action Taken**

Supervised by

Supervisor's Name

Signature_____

1. Introduction

Briefly explain scope of the project covered till now including modules.

2. Design Methodology and Software Process Model

Explain and justify the choice of design methodology being followed. (OOP or Procedural). Also explain which process model you are following and why.

3. System Overview

Give a general description of the functionality, context, and design of your project.
Provide any background information if necessary.

3.1 Architectural Design

Develop a modular program structure and explain the relationships between the modules to achieve the complete functionality of the system. This is a high-level overview of how the system's modules collaborate with each other in order to achieve the desired functionality.

Don't go into too much detail about the individual subsystems. The main purpose is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together.

Provide a diagram showing the major subsystems and their connections.

- In initial design stage create Box and Line Diagram for simpler representation of the systems
- After finalizing architecture style/pattern diagram (MVC, Client-Server, Layered, Multi-tiered) create a detailed mapping modules/components to each part of the architecture

To view example of box and line diagram and architecture styles, see Appendix A.

4. Design Models

Create design models as are applicable to your system. Provide detailed descriptions with each of the models that you add. Also ensure visibility of all diagrams.
Note: Follow version 2.5 for UML diagrams

Design Models for Object Oriented Development Approach

The applicable models for the project using object-oriented development approach may include:

- Class Diagram
- Sequence Diagram
- State Transition Diagram (for the projects which include event handling and backend processes)
- Activity Diagram
- Data flow diagrams

Design Models for Procedural Approach

The applicable models for the project using procedural approach may include:

- Activity Diagram
- Data Flow Diagram (data flow diagram should be extended to 2-3 levels. It should clearly list all processes, their sources/sinks and data stores.)
- State Transition Diagram (for the projects which include event handling and backend processes)

To view examples of all above models, see Appendix B

5. Data Design

Explain how the information domain of your system is transformed into data structures. Describe how the major data or system entities are stored, processed, and organized.

List any databases or data storage items in the form of data representation diagram (ERD, XML schema, SON schema etc.) with description.

5.1 Data Dictionary

Alphabetically list the system entities or major data along with their types and descriptions. If you provided a functional description, list all the functions and function parameters. If you provided an OO description, list the objects and its attributes, methods and method parameters.

6. User Interface Design

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

6.1 Screen Images

Display screenshots showing the interface from the user's perspective. These can be hand-drawn, or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

6.2 Screen Objects and Actions

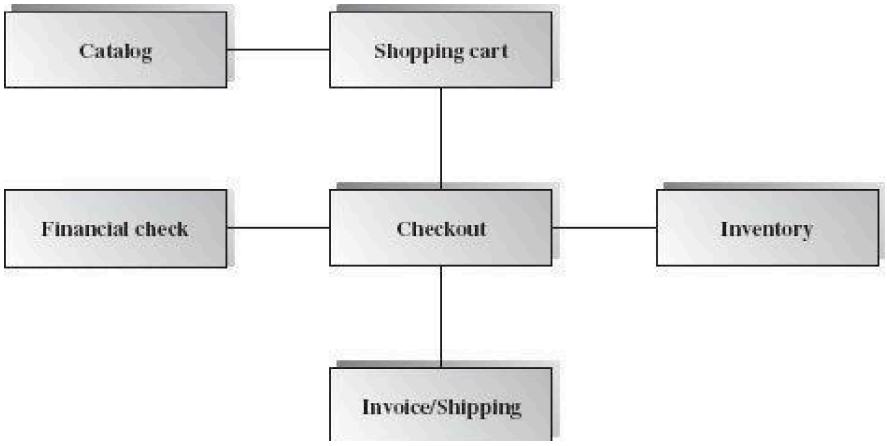
A discussion of screen objects and actions associated with those objects

• Appendix A

Box-and-line diagram

Box-and-line diagrams are often used to describe the business concepts and processes during the analysis phase of the software development lifecycle. These diagrams come with descriptions of components and connectors, as well as other descriptions that provide common inherent interpretations.

Example:



Lines in the box-and-line diagrams indicate the relationship among components

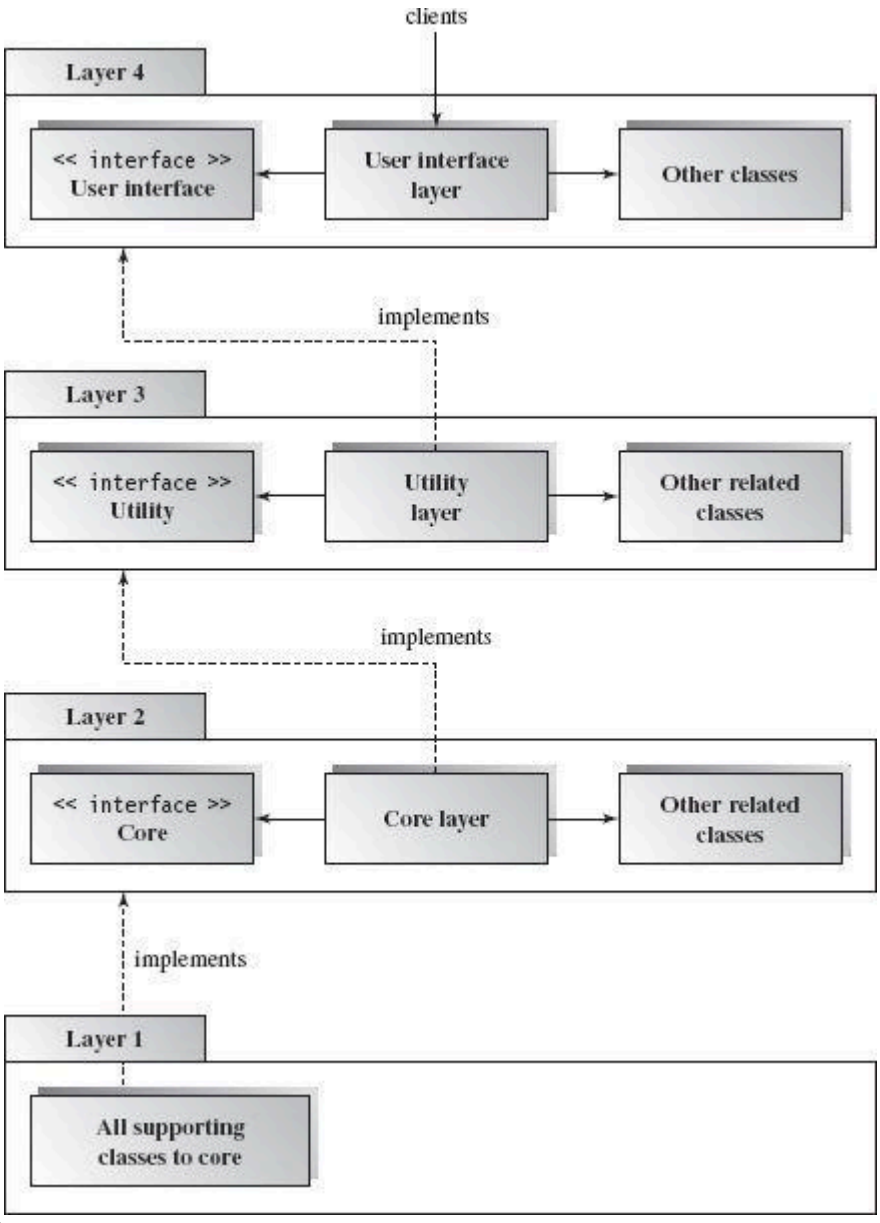
- The semantic of lines may refer dependency, control flow, data flow, etc
- Lines may be associated with arrows to indicate the process direction and sequence.
- A box-and-line diagram can be used as a business concept diagram describing its application domain and process concepts

Example of Architecture Pattern:

The **figure A-2** shows an example of the logical package organization of the layered architecture. The top level deals with user interface, the next level is for utilities, and the one below utility provides core services. Each layer gets support from its lower adjacent layer by an interface implementation and from the related classes in the same layer.

A simple software system may consist of two layers: an interaction layer and a processing layer:

- The interaction layer provides user interfaces to clients, takes requests, validates and forwards requests to the processing layer for processing, and responds to clients.
- The processing layer receives the forwarded requests and performs the business logic process, accesses the database, returns the results to its upper layer, and lets the upper layer respond to clients since the upper layer has the GUI interface responsibility.



Note: The Architecture pattern shall be selected according to the targeted system's requirements and quality attributes. Above example is provided to demonstrate that how the system architecture is required to be presented.

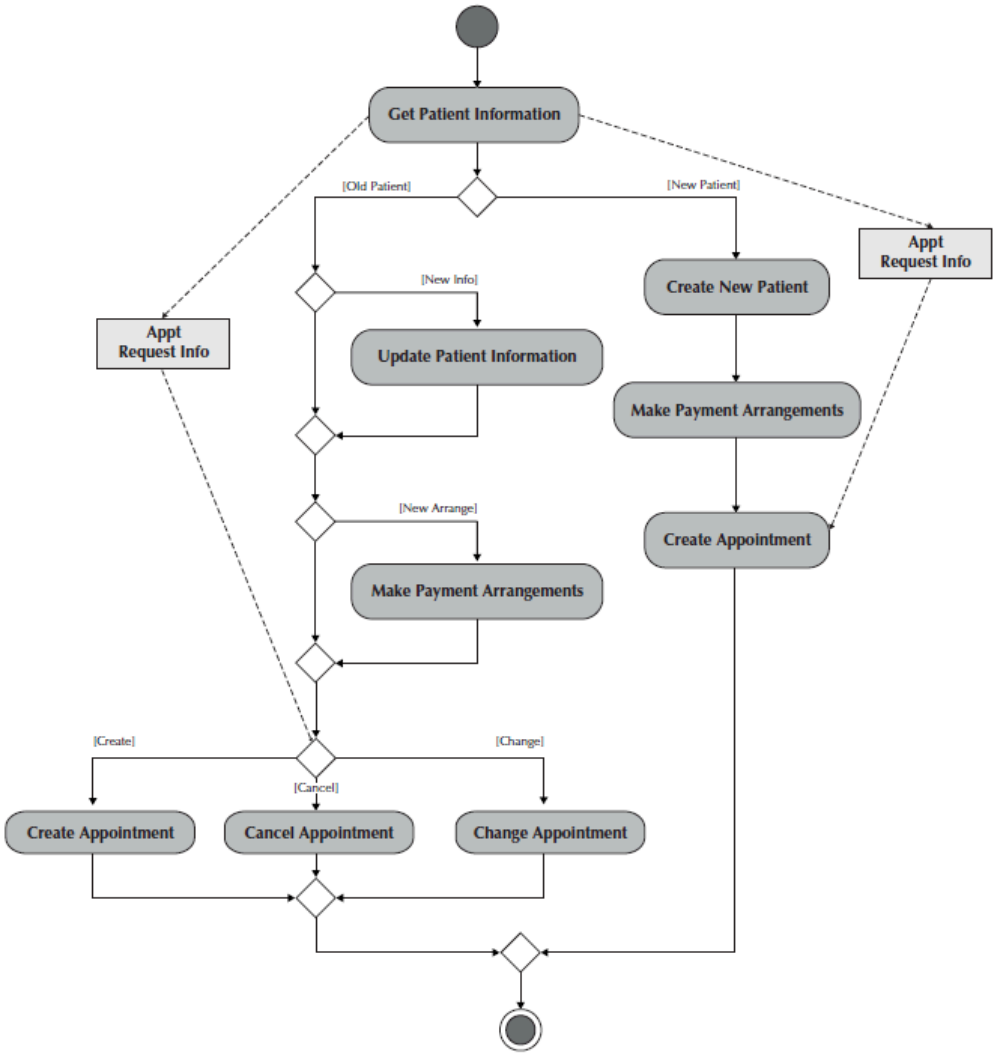
• Appendix B

Design Models

Activity Diagram

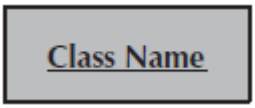
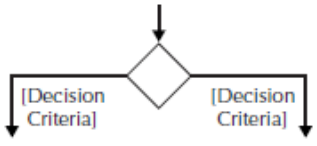
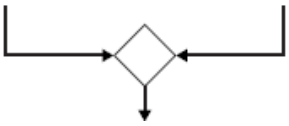
The following activity diagram is for an appointment system presenting a "**make an appointment"** process in which all diagram's elements are presented. Furthermore, **Table B-1** presents the details of the syntax of an activity diagram.

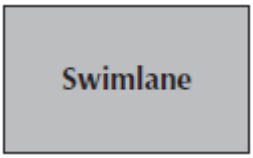
Example



Activity Diagram Syntax

Table B- 1Activity Diagram Syntax

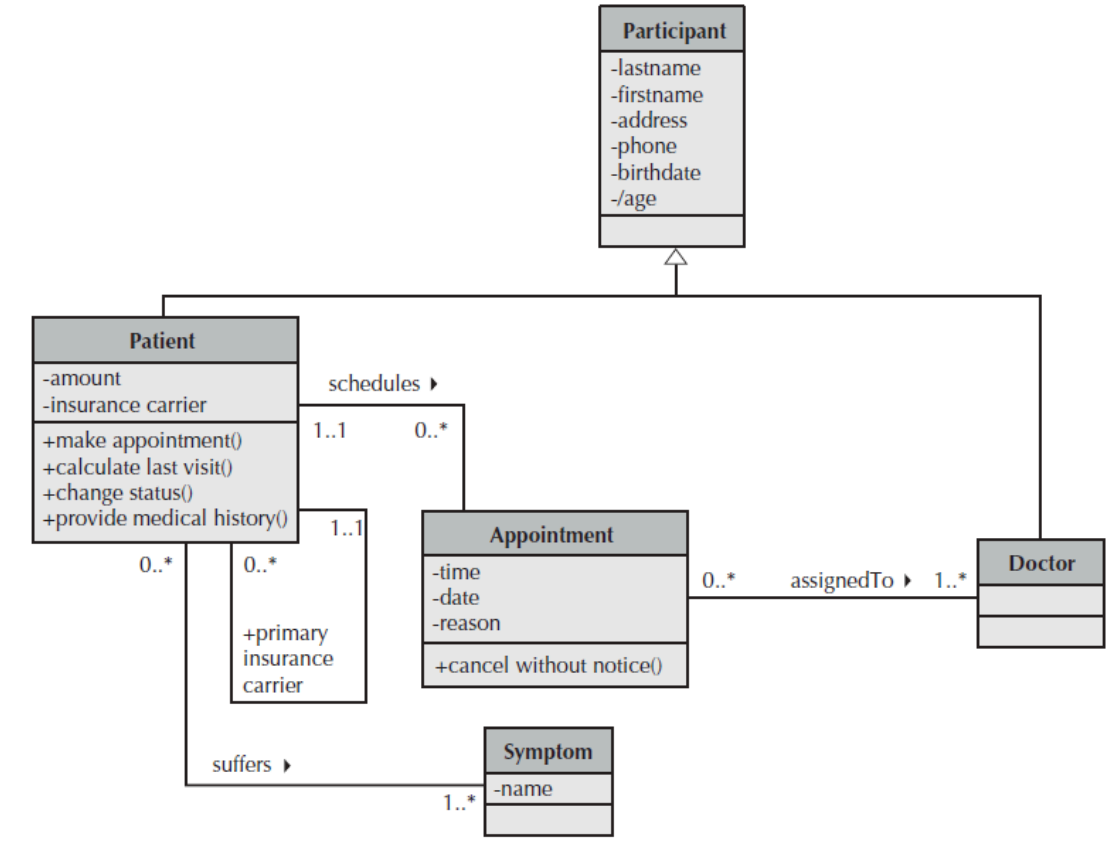
Term and definition	Symbol
An action: ▪ Is a simple, non-decomposable piece of behavior. ▪ Is labeled by its name.	
An activity: ▪ Is used to represent a set of actions. ▪ Is labeled by its name.	
An object node: ▪ Is used to represent an object that is connected to a set of object flows. ▪ Is labeled by its class name.	
A control flow: ▪ Shows the sequence of execution.	
An object flow: ▪ Shows the flow of an object from one activity (or action) to another activity (or action).	
An initial node: ▪ Portrays the beginning of a set of actions or activities.	
A final-activity node: ▪ Is used to stop all control flows and object flows in an activity (or action).	
A final-flow node: ▪ Is used to stop a specific control flow or object flow.	
A decision node: ▪ Is used to represent a test condition to ensure that the control flow or object flow only goes down one path. ▪ Is labeled with the decision criteria to continue down the specific path.	
A merge node: ▪ Is used to bring back together different decision paths that were created using a decision node.	
A fork node:	

▪ Is used to split behavior into a set of parallel or concurrent flows of activities (or action)	
A join node: ▪ Is used to bring back together a set of parallel or concurrent flows of activities (or action)	
A swimlane: ▪ Is used to break up an activity diagram into rows and columns to assign the individual activities (or actions) to the individuals or objects that are responsible for executing the activity (or action) ▪ Is labeled with the name of the individual or object responsible	

Class Diagram

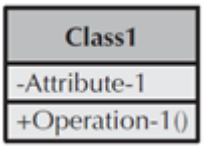
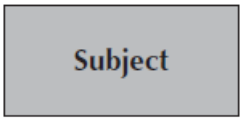
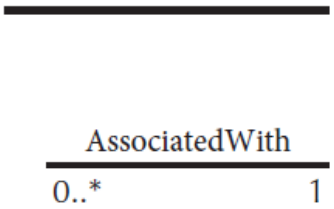
Following class diagram is of an appointment system in which all class diagram elements are presented. **Table B-2** presents the syntax of the class diagram as a guideline.

Example


Participant
-lastname
-firstname
-address
-phone
-birthdate
-/age
Patient
-amount
-insurance carrier
+make appointment()
+calculate last visit()
+change status()
+provide medical history()
schedules
1..1
0..*
1..1
0..*
0..*
+primary insurance carrier
Appointment
-time
-date
-reason
+cancel without notice()
0..*
assignedTo
1..*
Doctor
suffers
1..*
Symptom
-name

Class Diagram Syntax

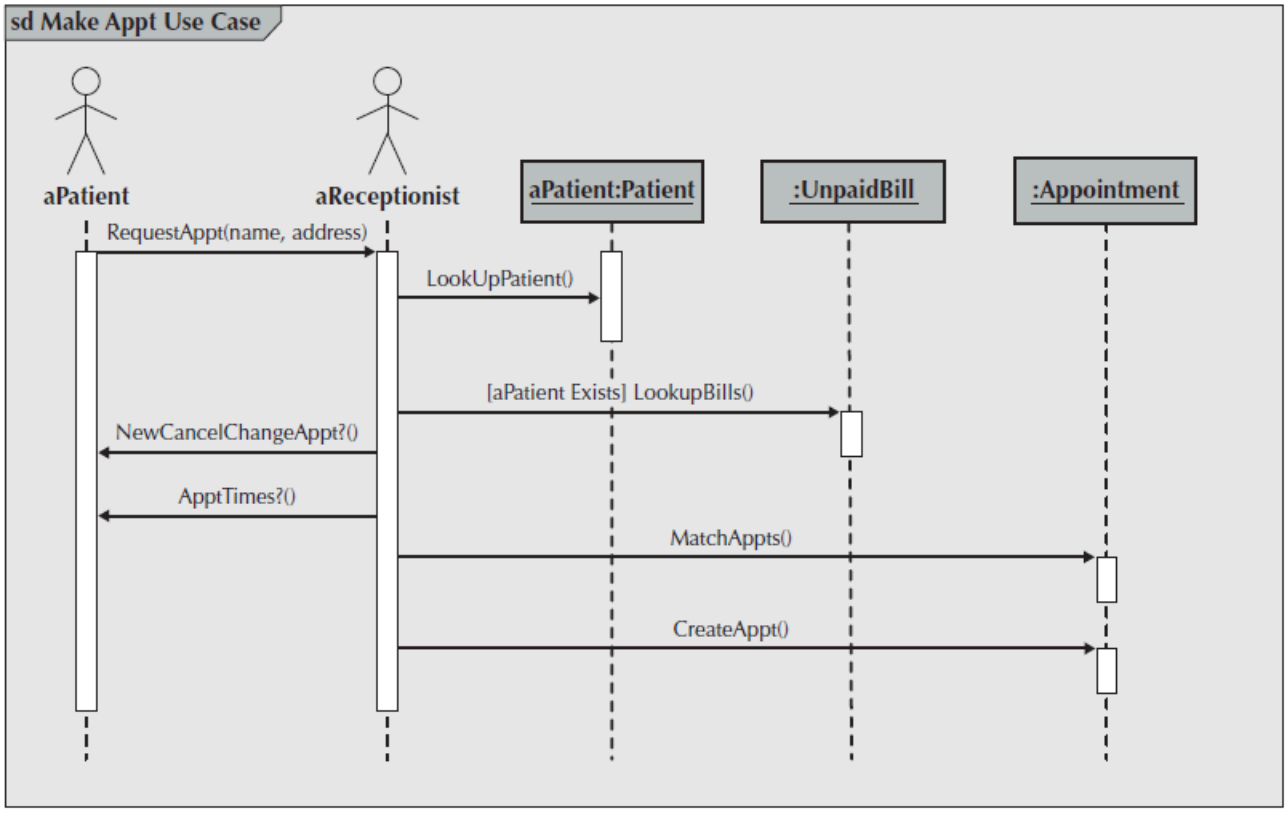
Table B- 2 Class Diagram Syntax

Term and definition	Symbol
A class: ▪ Has a name typed in bold and centered in its top compartment. ▪ Has a list of attributes in its middle compartment. ▪ Represents a kind of person, place, or thing about which the system will need to capture and store information. ▪ Has a list of operations in its bottom compartment. ▪ Does not explicitly show operations that are available to all classes.	
An attribute: ▪ Represents properties that describe the state of an object. ▪ Can be derived from other attributes, shown by placing a slash before the attribute's name.	attribute name /derived attribute name
An operation: ▪ Represents the actions or functions that a class can perform. ▪ Can be classified as a constructor, query, or update operation. ▪ Includes parentheses that may contain parameters or information needed to perform the operation.	 operation name ()
An association: ▪ Represents a relationship between multiple classes or a class and itself. ▪ Is labeled using a verb phrase or a role name, whichever better represents the relationship. ▪ Can exist between one or more classes. ▪ Contains multiplicity symbols, which represent the minimum and maximum times a class instance can be associated with the related class instance.	
A generalization: ▪ Represents a-kind-of relationship between multiple classes.	
An aggregation:	

▪ Represents a logical a-part-of relationship between multiple classes or a class and itself. ▪ Is a special form of an association.	
A composition: ▪ Represents a physical a-part-of relationship between multiple classes or a class and itself ▪ Is a special form of an association.	1..* IsPartOf 1

Sequence Diagram

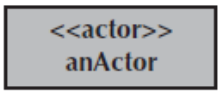
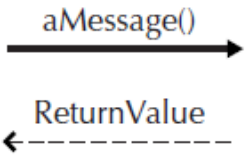
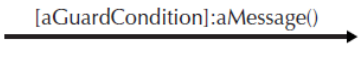
Following example shows an instance sequence diagram that depicts the objects and messages for the Make Old Patient Appt use case, which describes the process by which an existing patient creates a new appointment or cancels or reschedules an appointment. In further in **Table B-3** to the detail of class diagram syntax is provided.

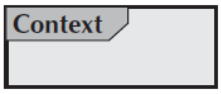


Example

Sequence diagram Syntax

Table B-3 Sequence Diagram Syntax

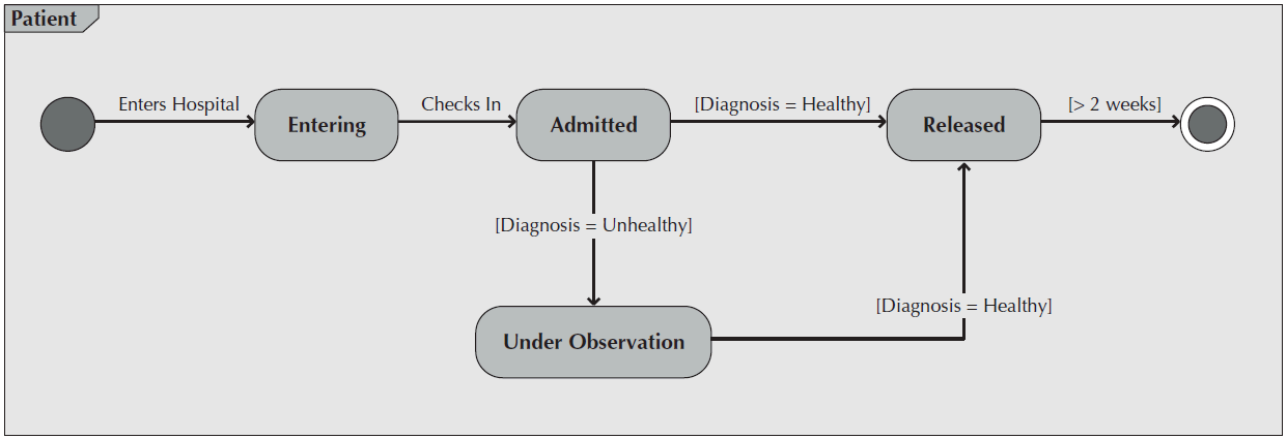
Term and definition	Symbol
An actor: ▪ Is a person or system that derives benefit from and is external to the system. ▪ Participates in a sequence by sending and/or receiving messages. ▪ Is placed across the top of the diagram. ▪ Is depicted either as a stick figure (default) or, if a nonhuman actor is involved, as a rectangle with <<actor>> in it (alternative).	 anActor 
An object: ▪ Participates in a sequence by sending and/or receiving messages. ▪ Is placed across the top of the diagram.	
A lifeline: ▪ Denotes the life of an object during a sequence. ▪ Contains an X at the point at which the class no longer interacts.	
An execution occurrence: ▪ Is a long narrow rectangle placed atop a lifeline. ▪ Denotes when an object is sending or receiving messages.	
A message: ▪ Conveys information from one object to another one. ▪ An operation call is labeled with the message being sent and a solid arrow, whereas a return is labeled with the value being returned and shown as a dashed arrow.	
A guard condition: ▪ Represents a test that must be met for the message to be sent.	
For object destruction: ▪ An X is placed at the end of an object's lifeline to show that it is going out of existence.	X

A frame: ▪ Indicates the context of the sequence diagram.	

Behavioral State Machine Diagram

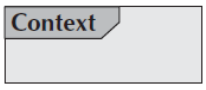
Following example of a behavioral state machine representing the patient class in the context of a hospital environment. From this diagram, we can tell that a patient enters a hospital and is admitted after checking in. If a doctor finds the patient to be healthy, he or she is released and is no longer considered a patient after two weeks elapse. If a patient is found to be unhealthy, he or she remains under observation until the diagnosis changes.

Example



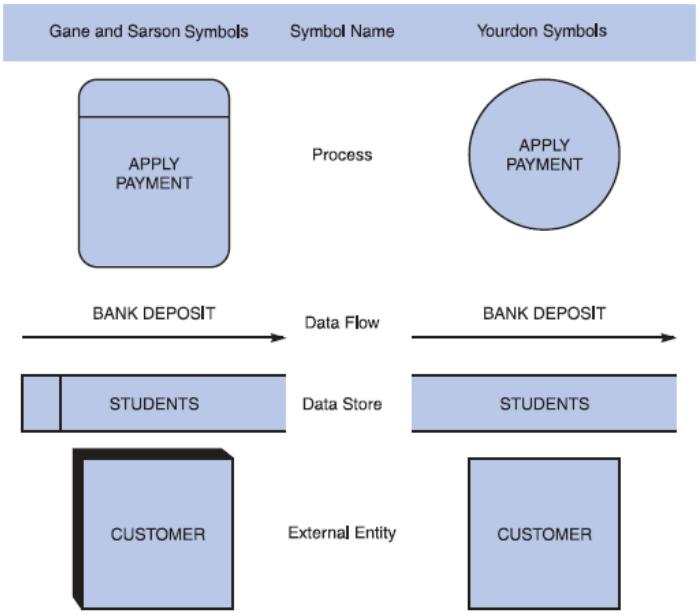
Behavioral State Machine Diagram Syntax

Table B-4 Behavioral State Machine Diagram Syntax

Term and definition	Symbol
A state: ▪ Is shown as a rectangle with rounded corners. ▪ Has a name that represents the state of an object.	
An initial state: ▪ Is shown as a small, filled-in circle. ▪ Represents the point at which an object begins to exist.	
A final state: ▪ Is shown as a circle surrounding a small, filled-in circle (bull's-eye). ▪ Represents the completion of activity.	
An event: ▪ Is a noteworthy occurrence that triggers a change in state. ▪ Can be a designated condition becoming true, the receipt of an explicit signal from one object to another, or the passage of a designated period of time. ▪ Is used to label a transition.	anEvent
A transition: ▪ Indicates that an object in the first state will enter the second state. ▪ Is triggered by the occurrence of the event labeling the transition. ▪ Is shown as a solid arrow from one state to another, labeled by the event name.	
A frame: ▪ Indicates the context of the behavioral state machine.	

Data Flow Diagram

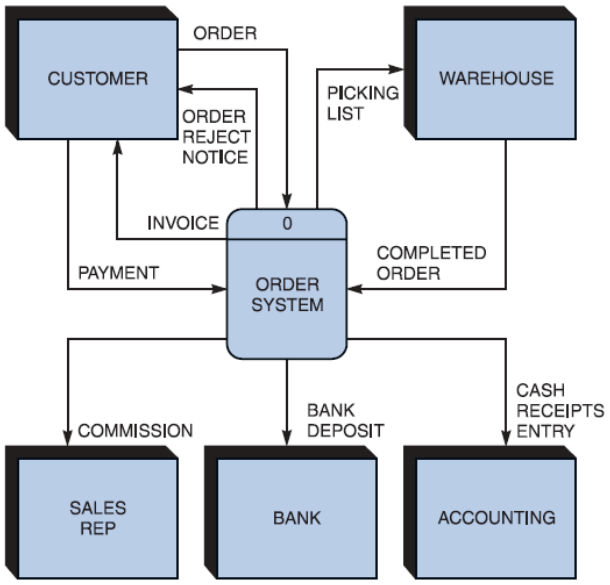
Data flow diagram symbols, symbol names, and examples



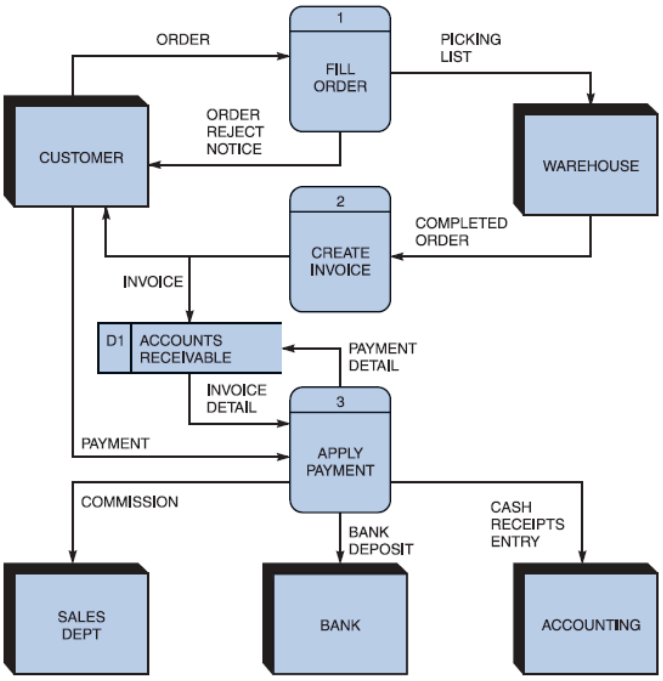
Guidelines for Drawing DFDs

Step 1: Draw a Context Diagram: The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

Example



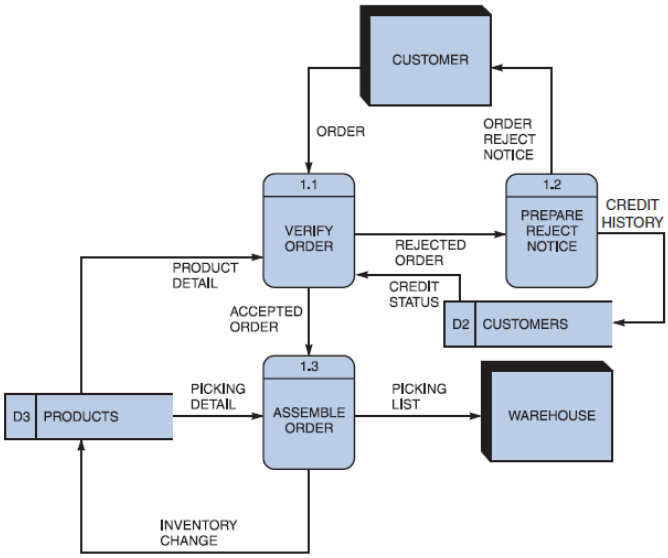
Step 2: Draw a Diagram 0 DFD: To show the detail inside the black box, you create DFD diagram 0. **Diagram 0** zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0. \



Example

Step 3: Draw the Lower-Level Diagrams:

To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.

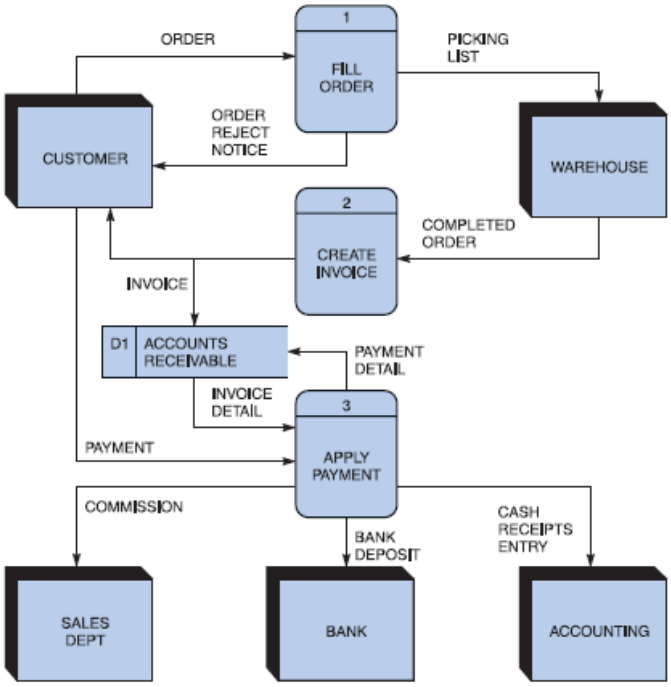


Leveling Example

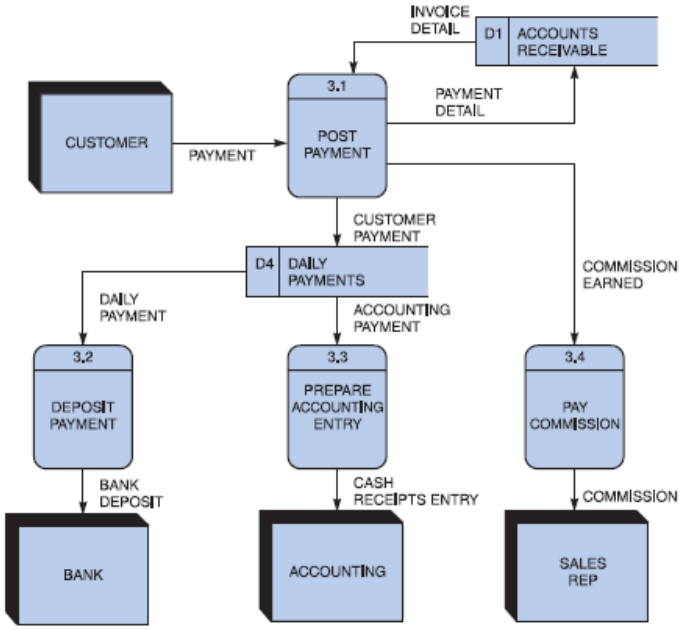
Balancing maintains consistency among a set of DFDs by ensuring that input and output data flows align properly.

Balancing Example

Order System Diagram 0 DFD



Order System Diagram 3 DFD



The order system diagram 0 is shown at the top of the figure and exploded diagram 3 DFD (for the APPLY PAYMENT process) is shown at the bottom. The two DFDs are balanced, because the child diagram at the bottom has the same input and output flows as the parent process 3 shown at the top.

Appendix C

Section 1 - Adding Captions and generating Lists of Figures Etc

- Add, format, or delete captions in Word