L D College of Engineering, Ahmedabad

Department of Information Technology

AY:2023

GTU B.E. 6th Semester

3161605- Software Engineering

Lab Manual

L D College of Engineering, Ahmedabad

Department of Information Technology

GTU B.E. 6th Semester **3161605- Software Engineering**

Practical List

Sr. No.	Aim	Hrs
1	**SDLC Models** Study and Prepare Summarize various Software Models. Application, Advantages, Disadvantages	2
2	**Software Engineering Project** Select a MIS System and Prepare Problem Description, Solution, User Roles and Responsibility, Inputs and Deliverable Output Products for system ● Prepare List of Requirements with Classification of Requirement(Feasibility, Functional/Non Functional, Type: User/System, Priority, Delivery Mode: InPhase/Immediate) ● Prepare **SRS Document** for Selected Project	2
3	**Structured Software Engineering** Prepare following software engineering documents ● Data flow Diagrams (0 level to Higher levels) ● Data Dictionary ● ER Diagram Database Queries	4
4	**Object Oriented Software Engineering: Structural UML Diagrams** Prepare following diagrams for selected project: ● Class diagram, ● Object diagram ● Package diagram ● Component diagram ● Composite structure diagram ● Deployment diagram	4
5	**Object Oriented Software Engineering: Behavioral UML Diagrams** Prepare following diagrams for selected project: ● Use case diagram ● Activity diagram ● Sequence diagram ● State diagram ● Communication diagram ● Interaction overview diagram ● Timing diagram	4
6	**User Interface Design** Draw User Interface using CUI/GUI Methods-Menu Driven, Card Driven for selected project. Prepare **SoftwareDesign Document** for the Project.	2
7	**Software Project Management :Activity Scheduling** Prepare the Time Line and Activity Scheduling using Turbo Project, Microsoft Project Tool Prepare Software **Project Plan Document/Gantt Chart** for the Project	2
8	**Software Project Cost Estimation** Calculate the Various Costs using CoCoMo, Function Point ,Algorithmic Cost Modeling Methods	2
9	**Software Testing** Design Test Cases and Scenarios for Testing Software as a parts and as a whole Prepare Software **Project Test Plan Document** for the Project	2
10	**Case study** on CASE Tools for Software Processes	2

11	**Review** of Website/ Desktop Application from Software Engineering Perspectives.	2

L D College of Engineering, Ahmedabad

Department of Information Technology

GTU B.E. 6th Semester **3161605- Software Engineering**

Laboratory Plan

Sr. No.	Aim	Planned Week
1	**SDLC Models** Study and Prepare Summarize various Software Models. Application, Advantages, Disadvantages	1
2	**Software Engineering Project** Select a MIS System and Prepare Problem Description, Solution, User Roles and Responsibility, Inputs and Deliverable Output Products for system ● Prepare List of Requirements with Classification of Requirement(Feasibility, Functional/Non Functional, Type: User/System, Priority, Delivery Mode: InPhase/Immediate) ● Prepare **SRS Document** for Selected Project	2
3	**Structured Software Engineering** Prepare following software engineering documents ● Data flow Diagrams (0 level to Higher levels) ● Data Dictionary ● ER Diagram Database Queries	3
4	**Object Oriented Software Engineering: Structural UML Diagrams** Prepare following diagrams for selected project: ● Class diagram, ● Object diagram ● Package diagram ● Component diagram ● Composite structure diagram ● Deployment diagram	5
5	**Object Oriented Software Engineering: Behavioral UML Diagrams** Prepare following diagrams for selected project: ● Use case diagram ● Activity diagram ● Sequence diagram ● State diagram ● Communication diagram ● Interaction overview diagram ● Timing diagram	7
6	**User Interface Design** Draw User Interface using CUI/GUI Methods-Menu Driven, Card Driven for selected project. Prepare **SoftwareDesign Document** for the Project.	8
7	**Software Project Management :Activity Scheduling** Prepare the Time Line and Activity Scheduling using Turbo Project, Microsoft Project Tool Prepare Software **Project Plan Document/Gantt Chart** for the Project	9
8	**Software Project Cost Estimation** Calculate the Various Costs using CoCoMo, Function Point ,Algorithmic Cost Modeling Methods	10
9	**Software Testing** Design Test Cases and Scenarios for Testing Software as a parts and as a whole Prepare Software **Project Test Plan Document** for the Project	11

10	**Case study** on CASE Tools for Software Processes	12
11	**Review** of Website/ Desktop Application from Software Engineering Perspectives.	12

INDEX

GTU B.E. 6th Semester **Subject: 3161605- Software Engineering**

Student Name : _____ **Enrollment No:** _____

Student Name : _____ **Enrollment No:** _____

Student Name : _____ **Enrollment No:** _____

Sr. No.	Aim	Date	Mark	Sign
1	**SDLC Models**			
2	**Software Engineering Project :SRS Document**			
3	**Structured Software Engineering**			
4	**Object Oriented Software Engineering: Structural UML Diagrams**			
5	**Object Oriented Software Engineering: Behavioral UML Diagrams**			
6	**User Interface Design SoftwareDesign Document**			
7	**Software Project Management :Activity Scheduling** **SPMP-Software Project Management Plan /Gantt Chart**			
8	**Software Project Cost Estimation**			
9	**Software Testing:Project Test Plan Document**			
10	**Case study on CASE Tools for Software Processes**			
11	**Software Engineering Case Study Review**.			

Faculty Sign: _____

Practical 1: SDLC Models

AIM: Study and Prepare Summarize various Software Models.Application, Advantages, Disadvantages

Models: Waterfall Model, Iterative Waterfall model, Spiral Model, Prototyping model,V-Shape SDLC, Agile SDLC

Parameters:

A. **Description**
B. **Diagram**
C. **Stages/Phases**
D. **Type of Software where it is applicable**
E. **Characteristics of Software Project**
F. **Characteristics of Software Project Team with Size**
G. **Risk Associated with Project**
H. **Characteristics of User/Customer**
I. **Scope and Cost of Change Request Management**

Practical 2: Software Engineering Project :SRS Document

AIM: Prepare Problem Description, Solution, User Roles and Responsibility, Inputs and Deliverable Output Products for Selected Project.

1. Prepare List of Requirements with Classification of Requirement (Feasibility, Functional /Non Functional, Type: User/System, Priority, Delivery Mode: InPhase/Immediate)
2. Prepare SRS Document for Selected Project

Project Title:

Stakeholders:

Major Requirements :

Expected Delivery Time(Months):

User Roles:

Responsibility of Roles:

Major Modules:

1.

2.

Requirement No (ModuleNo.ReqNo)	**Description**	**Type** System /User	**Nature** Functional /NonFunctional
M1.R1	**Steps :** R1.1: R1.2: R1.3: 1. User (Responsible) 2. Priority(High/Medium/Low,Immediate/NextPhase) 3. Input **4.** Output		
M1.R2			
....			
M2.R1			

Practical 3: Structured Software Engineering

AIM: Prepare following software engineering documents

1. Data flow Diagrams (0 level to Higher levels)
2. Data Dictionary
3. ER Diagram
4. Database Queries

---

1. **Data flow Diagrams (0 level to Higher levels)**
    a. **Context Level Diagram**
    b. **1st Level Diagram (All Modules)**
    c. **2nd Level Diagram (Any Two Modules)**
2. **Data Dictionary : (For atleast 5 Tables)**

Table 1: Table Name **Purpose:**			
FieldName	**Data Type**	**Constraints** **IsPK,IsFK,Unique, NOTNULL**	**Description**

3. **ER Diagram**

4. **Database Queries (For atleast 10 Pseudo Database Queries)**

Practical 4: Object Oriented Software Engineering: Structural UML Diagrams

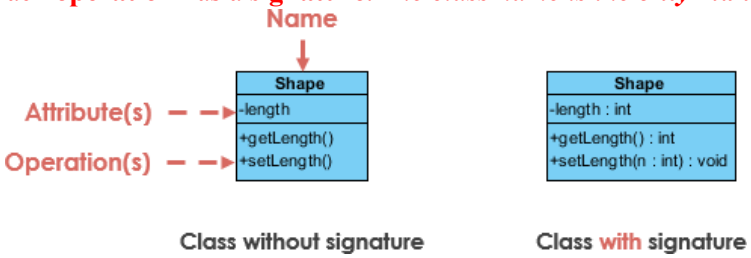
AIM: Prepare following diagrams for selected project:

1. Class diagram,
2. Object diagram
3. Package diagram
4. Component diagram
5. Composite structure diagram
6. Deployment diagram

---

1. Class diagram(All Classes)

A class represent a concept which encapsulates state (**attributes**) and behavior (**operations**). Each attribute has a type. Each **operation** has a **signature**. *The class name is the **only mandatory information***.



Class Name:

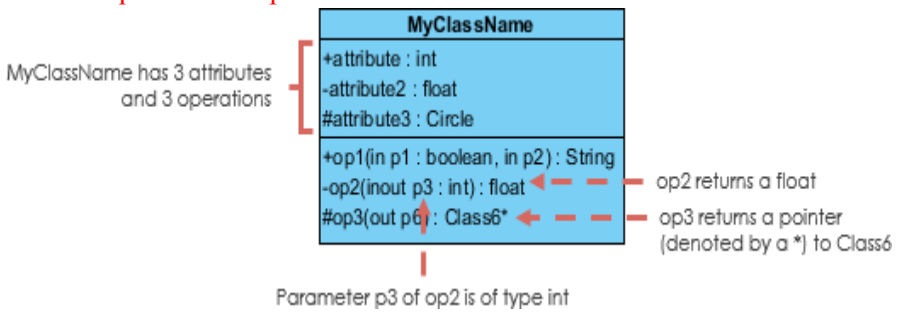
- The name of the class appears in the first partition.

Class Attributes:

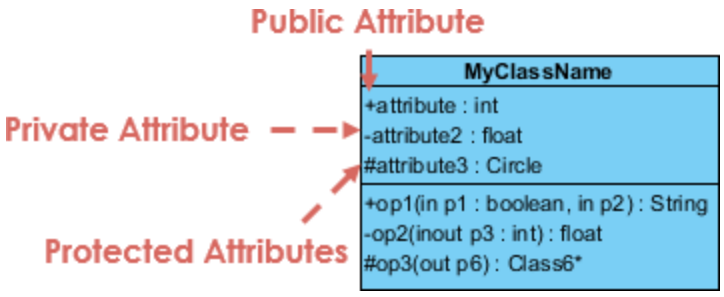
- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- Attributes map onto member variables (data members) in code.

Class Operations (Methods):

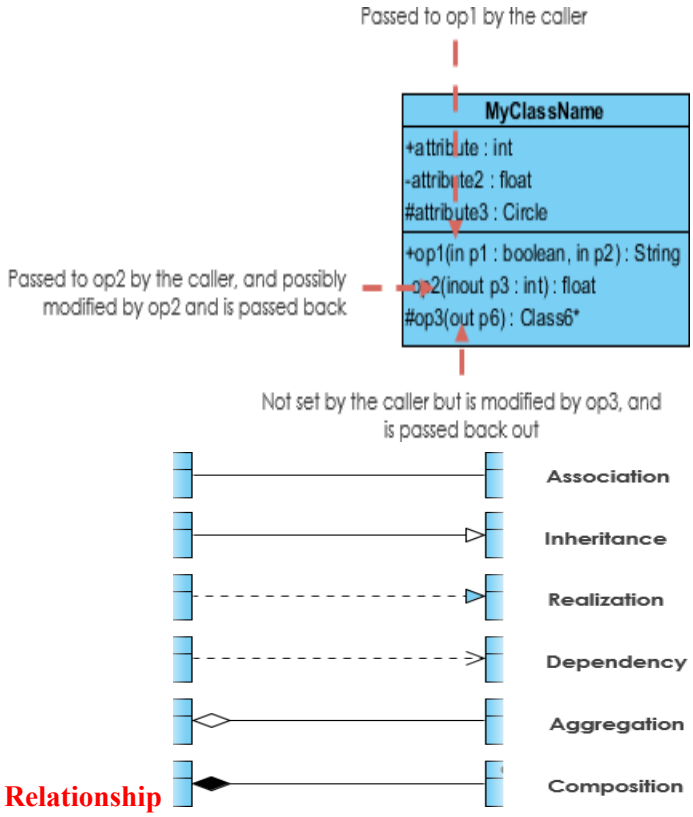
- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code



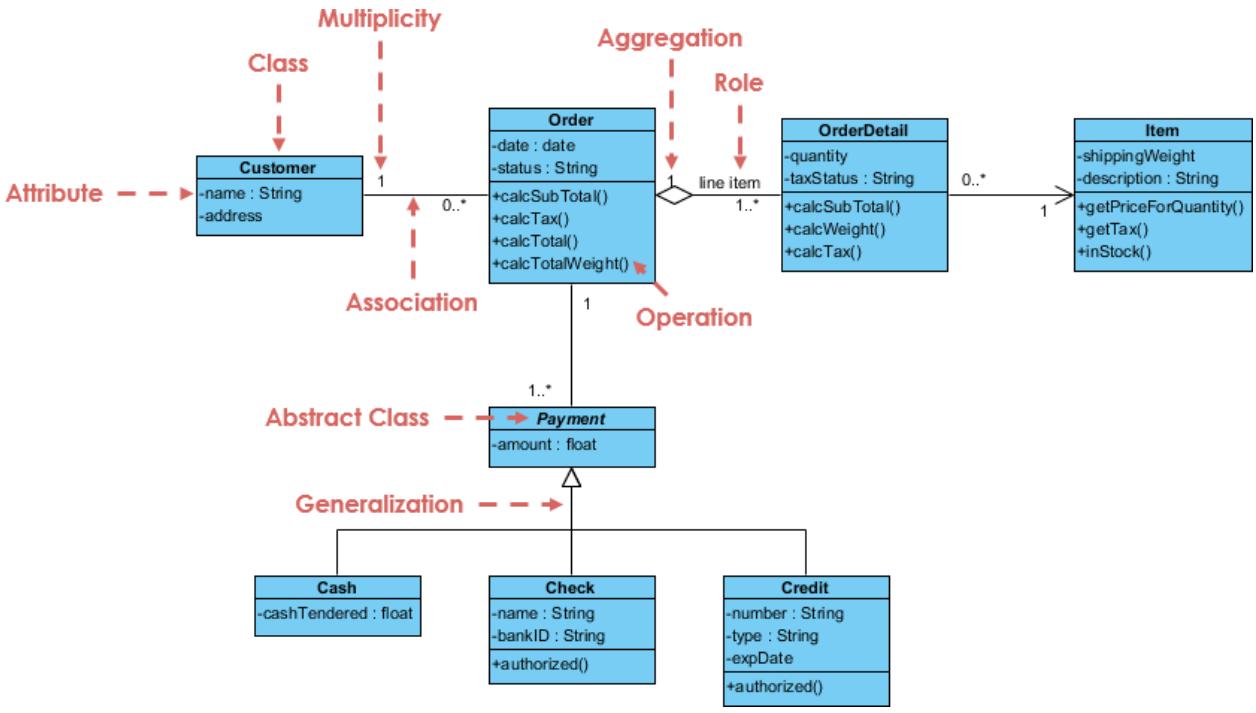
- Class Visibility: **The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.**
- \+ denotes public attributes or operations
- \- denotes private attributes or operations,
- # denotes protected attributes or operations



Parameter Directionality: Each parameter in an operation (method) may be denoted as in, **out** or **inout** which specifies its direction with respect to the caller. This directionality is shown before the parameter name.



Example: Order Processing



2. Object diagram (All Objects)

The use of object diagrams is fairly limited, mainly to show examples of data structures.

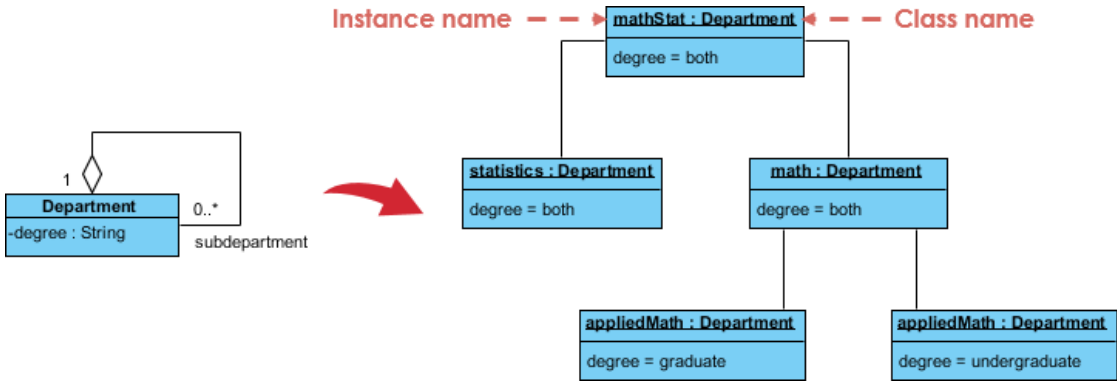
A. During the analysis phase of a project, you might create a class diagram to describe the structure of a system and then create a set of object diagrams as test cases to verify the accuracy and completeness of the class diagram.

B. Before you create a class diagram, you might create an object diagram to discover facts about specific model elements and their links, or to illustrate specific examples of the classifiers that are required.

C. Object Names: Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.

D. **Object Attributes:** Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.

E. **Links:** Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

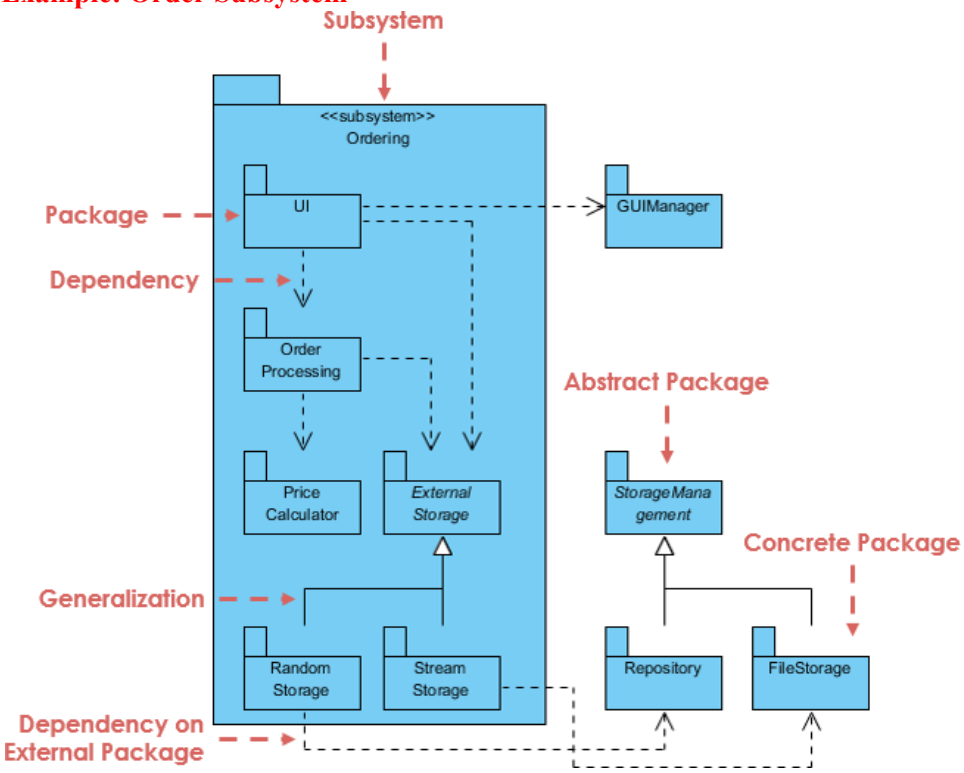


3. Package diagram

Package diagrams are used to structure high level system elements. Packages are used for organizing large system which contains diagrams, documents and other key deliverables.

A. Package Diagram can be used to simplify complex class diagrams, it can group classes into packages.
B. A package is a collection of logically related UML elements.
C. Packages are depicted as file folders and can be used on any of the UML diagrams.

Example: Order Subsystem

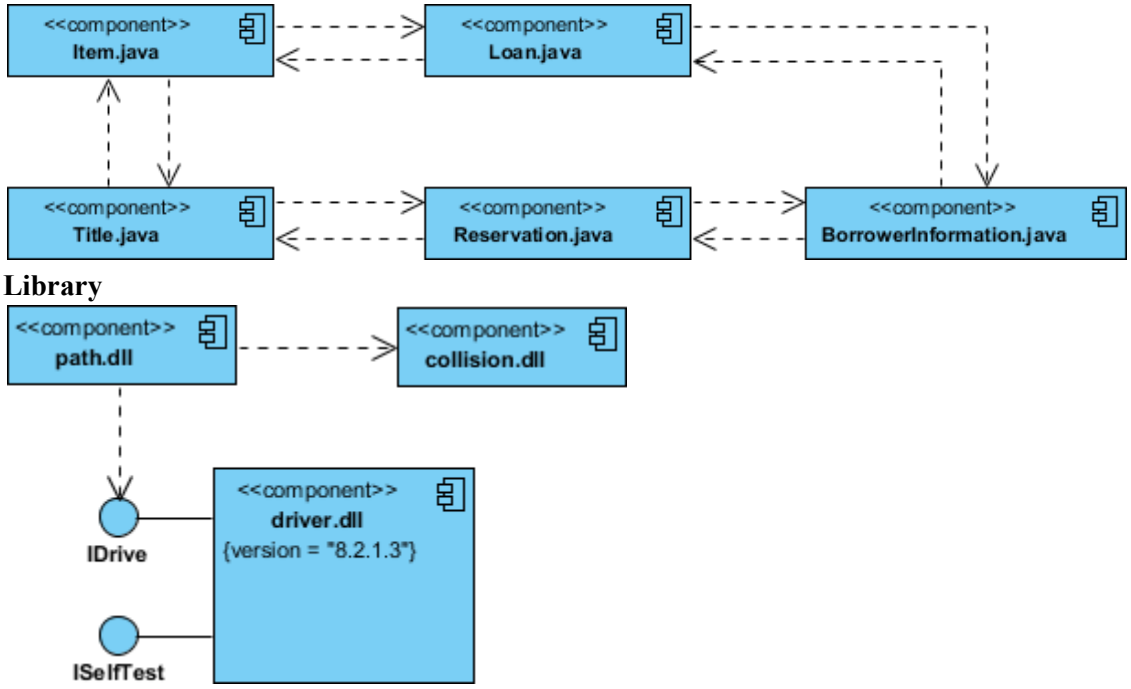


4. Component diagram

Component diagram is a collection of vertices and arcs and commonly contain components, interfaces and dependency, aggregation, constraint, generalization, association, and realization relationships. It may also contain notes and constraints.

A. Association
B. Composition
C. Aggregation
D. Constraint
E. Dependency
F. Links

Example : Java Source Code



5. Composite structure diagram:

A. Composite Structure Diagrams allow the users to "Peek Inside" an object to see exactly what it is composed of.

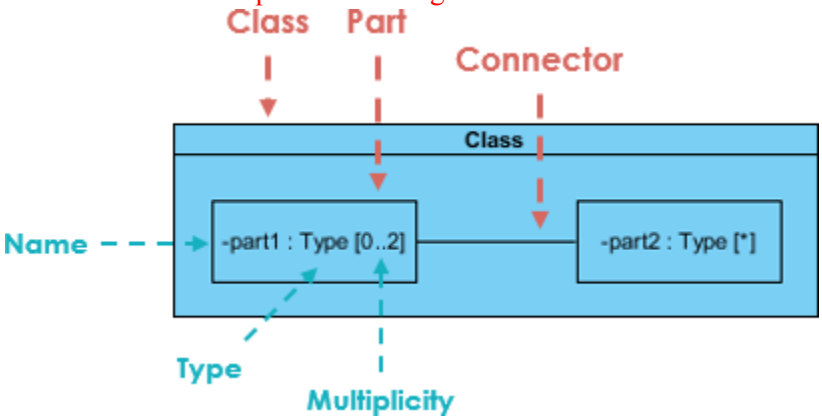
B. The internal actions of a class, including the relationships of nested classes, can be detailed.

C. Objects are shown to be defined as a composition of other classified objects.

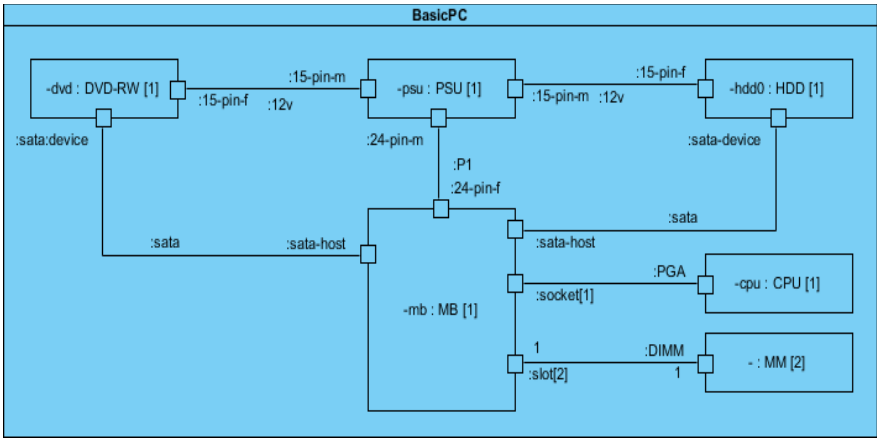
D. Composite Structure Diagrams show the internal parts of a class.

E. Parts are named: partName:partType[multiplicity]

F. Aggregated classes are parts of a class but parts are not necessarily classes, a part is any element that is used to make up the containing class.



Example : Computer System
Class, Part, Port, Multiplicity Connectors,Interface



6. Deployment diagram

A. They show the structure of the run-time system

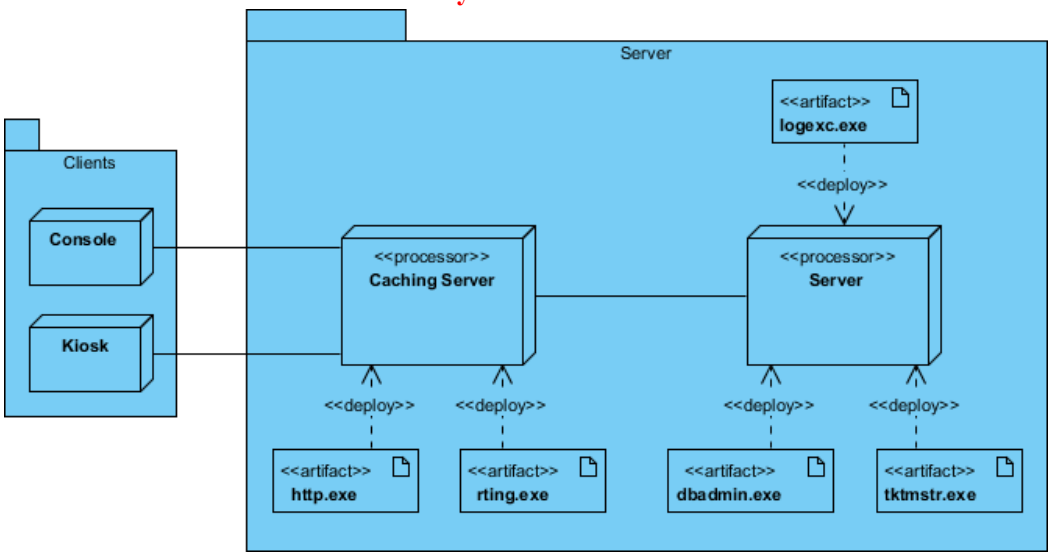
B. They capture the hardware that will be used to implement the system and the links between different items of hardware.

C. They model physical hardware elements and the communication paths between them

D. They can be used to plan the architecture of a system.

E. They are also useful for Document the deployment of software components or nodes

Place and Connections of Software SubSystems



Practical 5: Object Oriented Software Engineering: Behavioral UML Diagrams

AIM: Prepare following diagrams for selected project:

1. Use case diagram
2. Activity diagram
3. Sequence diagram
4. State diagram
5. Communication diagram
6. Interaction overview diagram
7. Timing diagram

---

1. Use case diagram

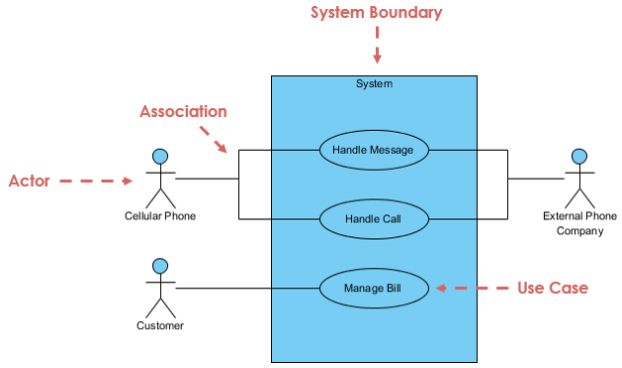
Purpose of Use Case Diagram

Use case diagrams are typically developed in the early stage of development and people often apply use case modeling for the following purposes:

- Specify the context of a system
- Capture the requirements of a system
- Validate a systems architecture
- Drive implementation and generate test cases
- Developed by analysts together with domain experts

A standard form of use case diagram is defined in the Unified Modeling Language as shown in the Use Case Diagram example below:

- Actor
- Use Case
- Communication Link
- Boundary of system

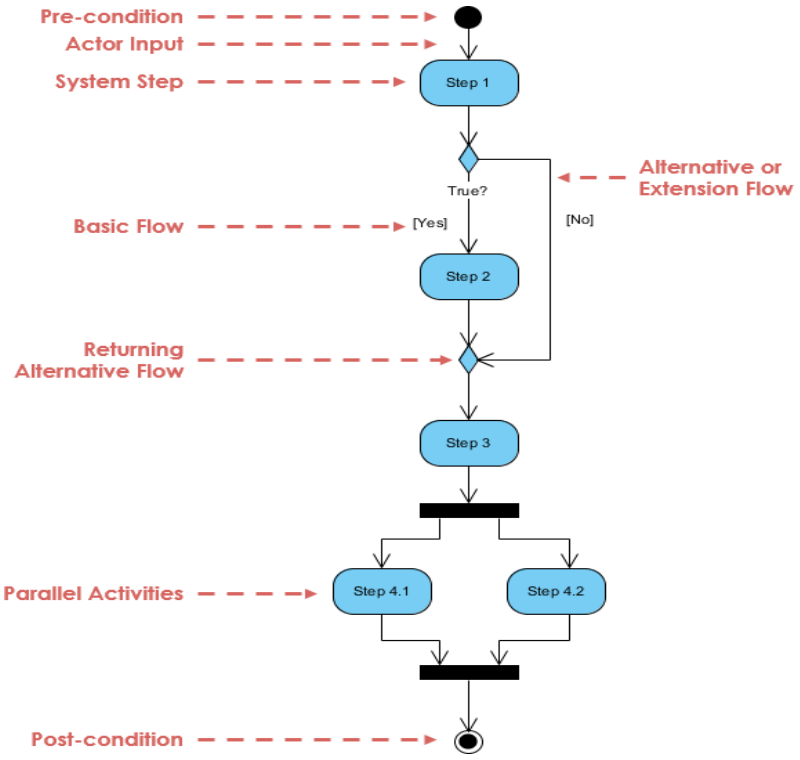


USE Case Relationship:

- Extends
- Include
- Generalization

2. Activity diagram

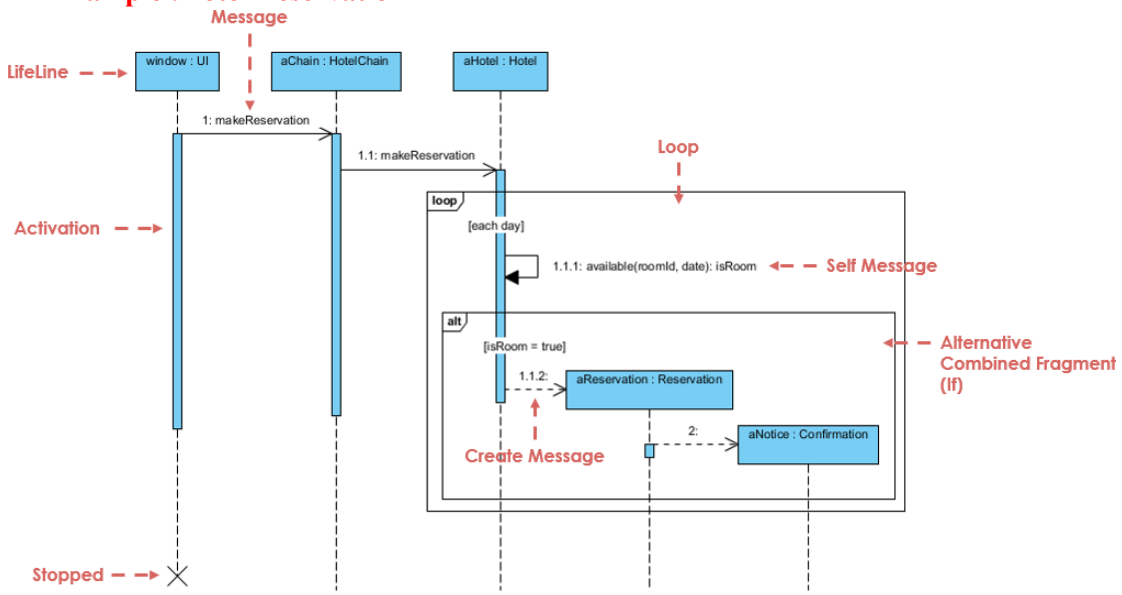
- Identify candidate use cases, through the examination of business workflows
- Identify pre- and post-conditions (the context) for use cases
- Model workflows between/within use cases
- Model complex workflows in operations on objects
- Model in detail complex activities in a high level activity Diagram



3. Sequence diagram

- Model high-level interaction between active objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

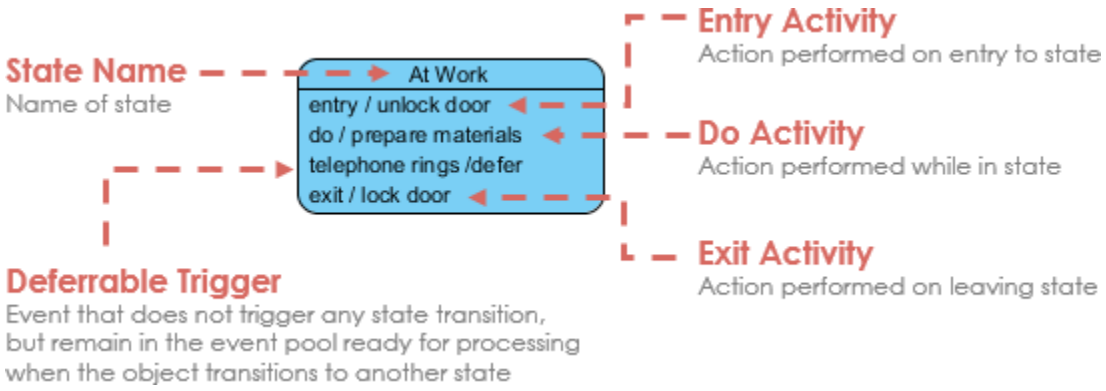
Example :Hotel Reservation



4. State diagram

State machine diagram typically are used to describe state-dependent behavior for an object. **An object responds differently to the same event depending on what state it is in**. State machine diagrams are usually applied to objects but can be applied to any element that has behavior to other

entities such as: actors, use cases, methods, subsystems systems and etc. and they are typically used in conjunction with interaction diagrams (usually sequence diagrams)



States:

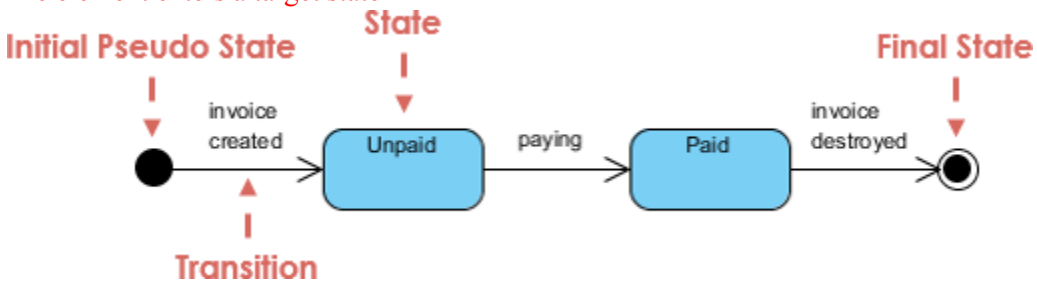
1. Initial State-Entry State
2. Final State -Exit State

Events :

1. Signal event - corresponding to the arrival of an asynchronous message or signal
2. Call event - corresponding to the arrival of a procedural call to an operation
3. Time event - a time event occurs after a specified time has elapsed
4. Change event - a change event occurs whenever a specified condition is met

Transition

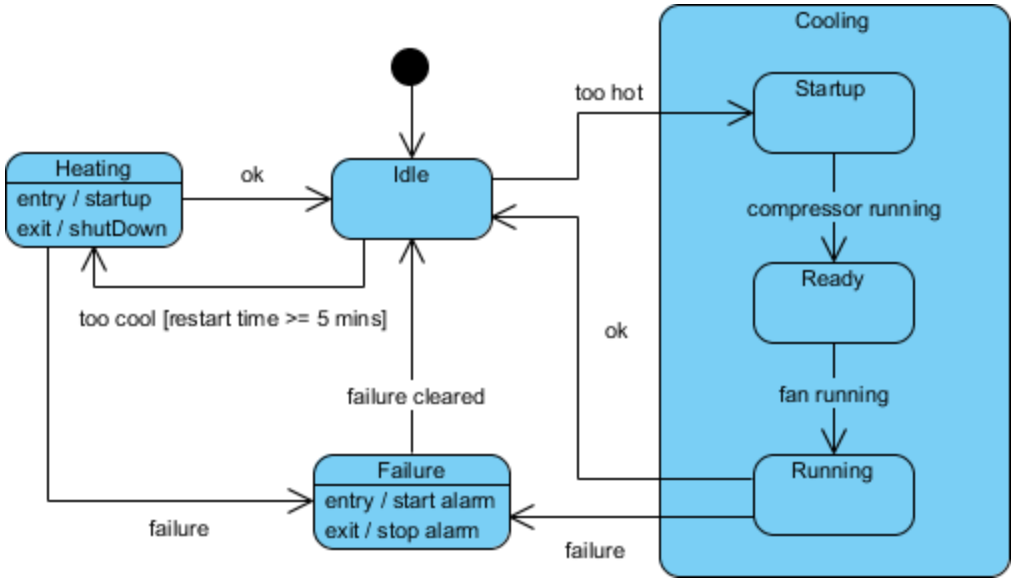
1. An element is in a source state
2. An event occurs
3. An action is performed
4. The element enters a target state



Substates:

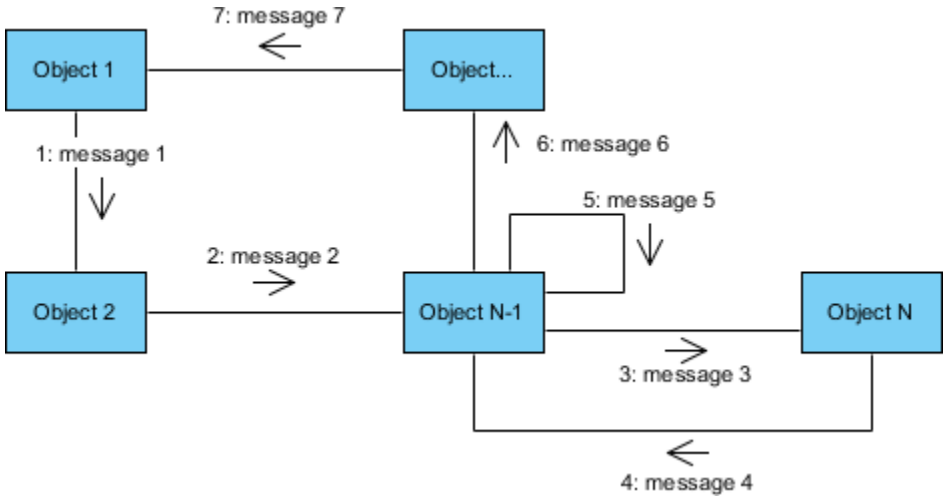
A simple state is one which has no substructure. A state which has substates (nested states) is called a composite state. Substates may be nested to any level. A nested state machine may have at most one initial state and one final state

Example:Heater

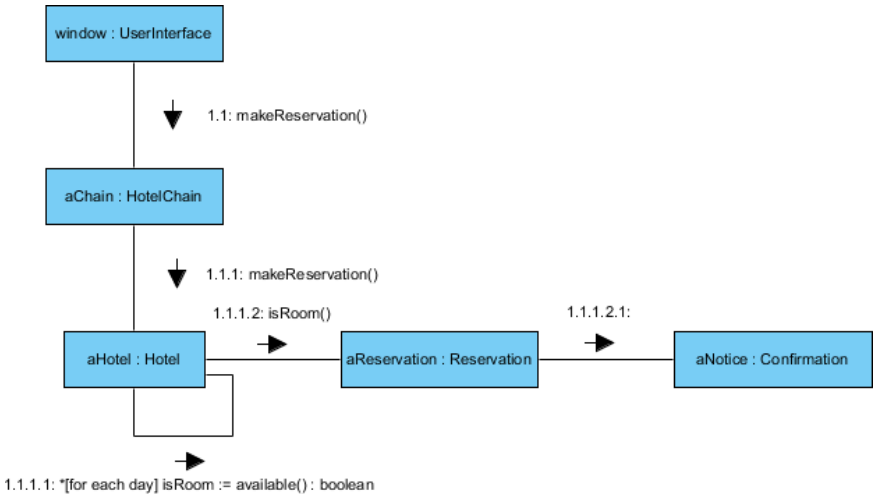


5. Communication diagram

- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Capture interactions that show the passed messages between objects and roles within the collaboration scenario
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use cases
- Each message in a communication diagram has a sequence number.
- The top-level message is numbered 1.
- Messages sent during the same call have the same decimal prefix, but suffixes of 1, 2, etc. according to when they occur.

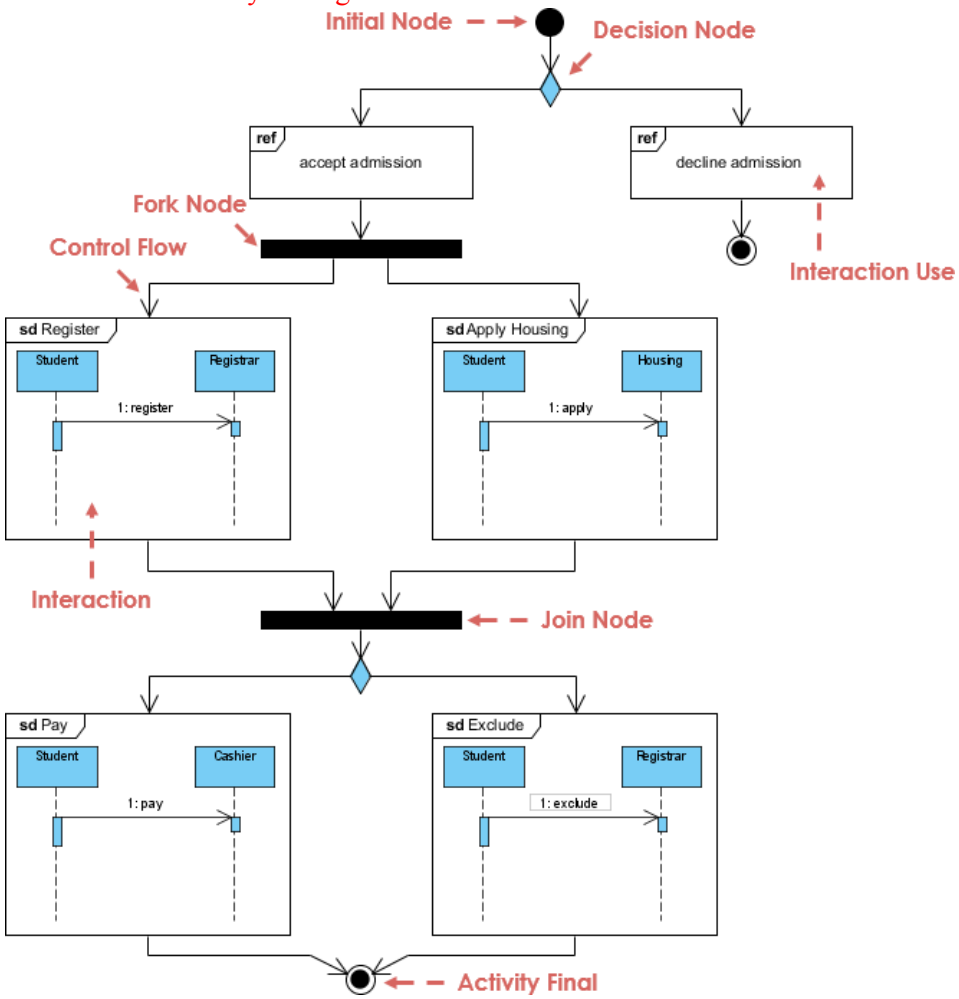


Example : Hotel Reservation Diagram



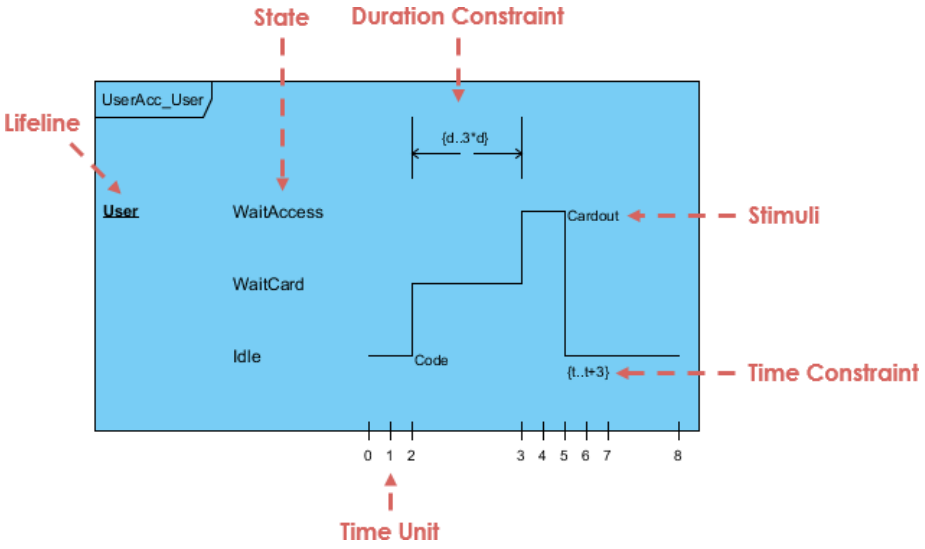
6. Interaction overview diagram

a student who has been accepted into a university. First the student must be accept or decline admission. After accepting, the student must both register for classes and apply for housing. After both of those are complete, the student must pay the registrar. If payment is not received in time the student is excluded by the registrar.

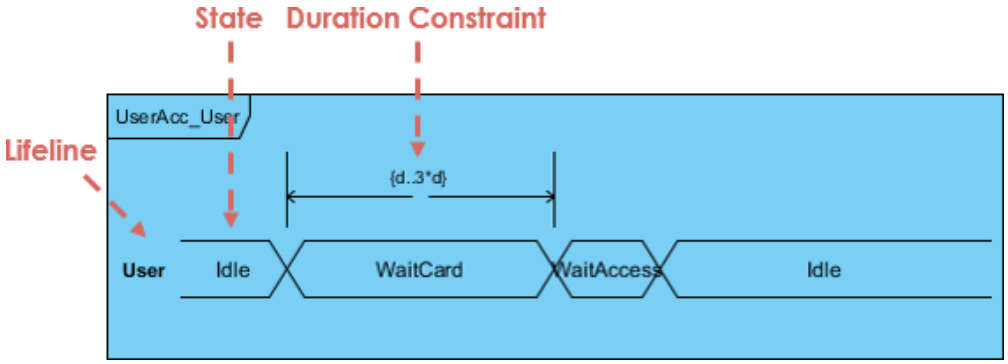


7. Timing diagram

Changes from one **state** to another are represented by **a change in the level of the lifeline**. For the period of time when the object is a given state, the timeline runs parallel to that state. A change in state appears as a vertical change from one level to another. The cause of the change, as is the case in a state or sequence diagram, is the receipt of a message, an event that causes a change, a condition within the system, or even just the passage of time.



Value Lifeline



Practical 6: User Interface Design Software Design Document

AIM: Draw User Interface using CUI/GUI Methods-Menu Driven, Card Driven for selected project.

Prepare **SoftwareDesign Document** for the Project.

GUI Design Layout(Desktop/Web/Mobile)(7-10 Screen Layout)

Design Principle for GUI

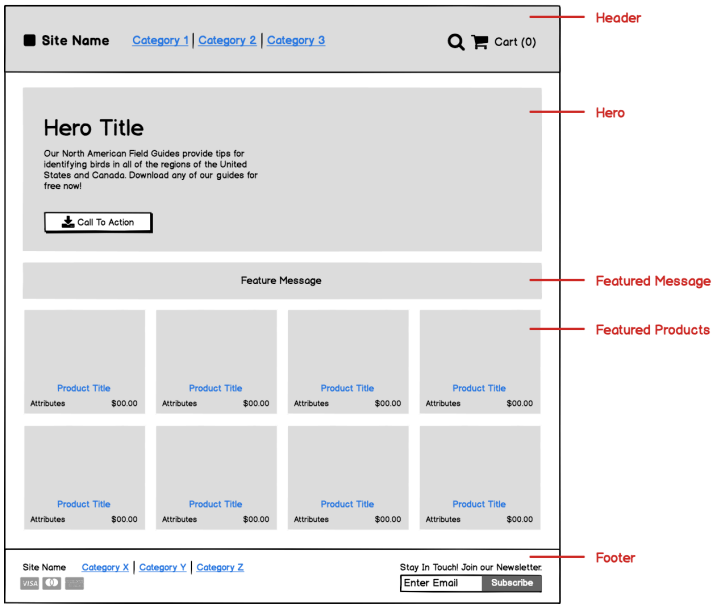
1. Contrast
2. Hierarchy
3. Proximity
4. Alignment

Components

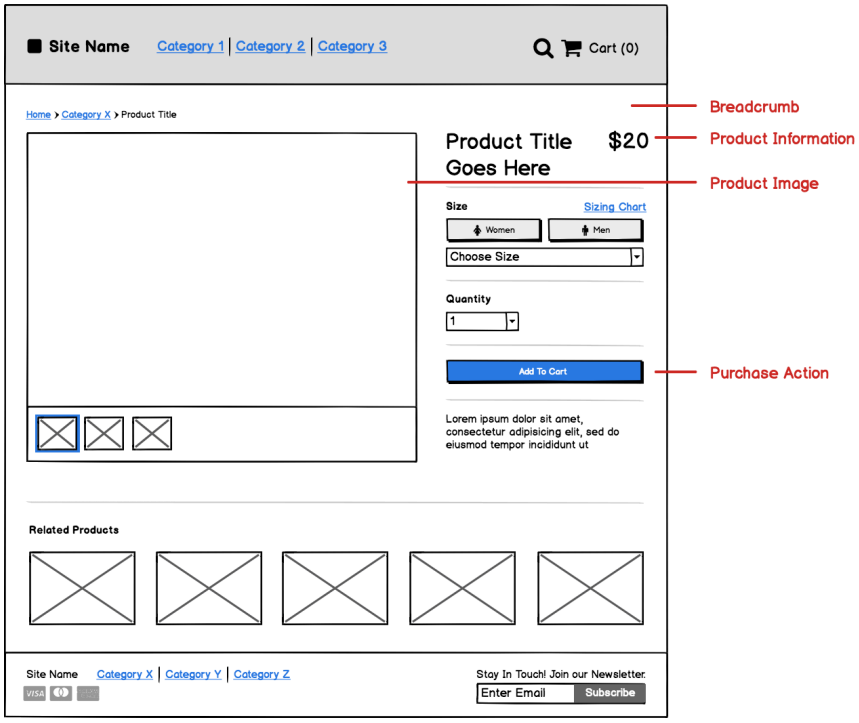
1. **Buttons**
1. **2 Text Input**
2. **Dropdown Menu (Combo Box)**
3. **Radio Buttons and Checkboxes**
4. **Links**
5. **Tabs**
6. **Breadcrumbs**
7. **Vertical Navigation**
8. **Menu Bars**
9. **Accordions**
10. **Validation**
11. **Tooltips**
12. **Alerts**
13. **Data Tables**
14. **Icons**

Shopping System

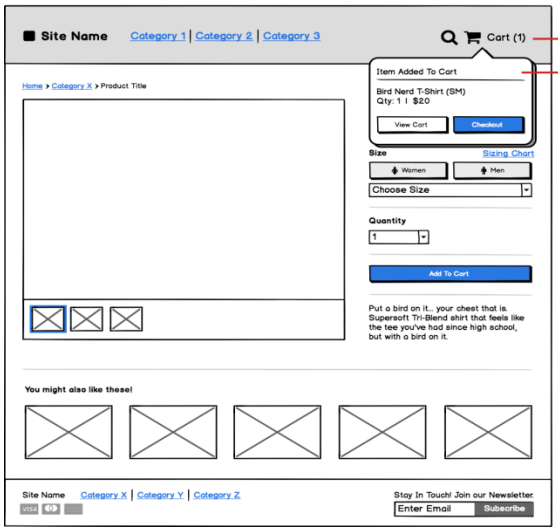
Home Page



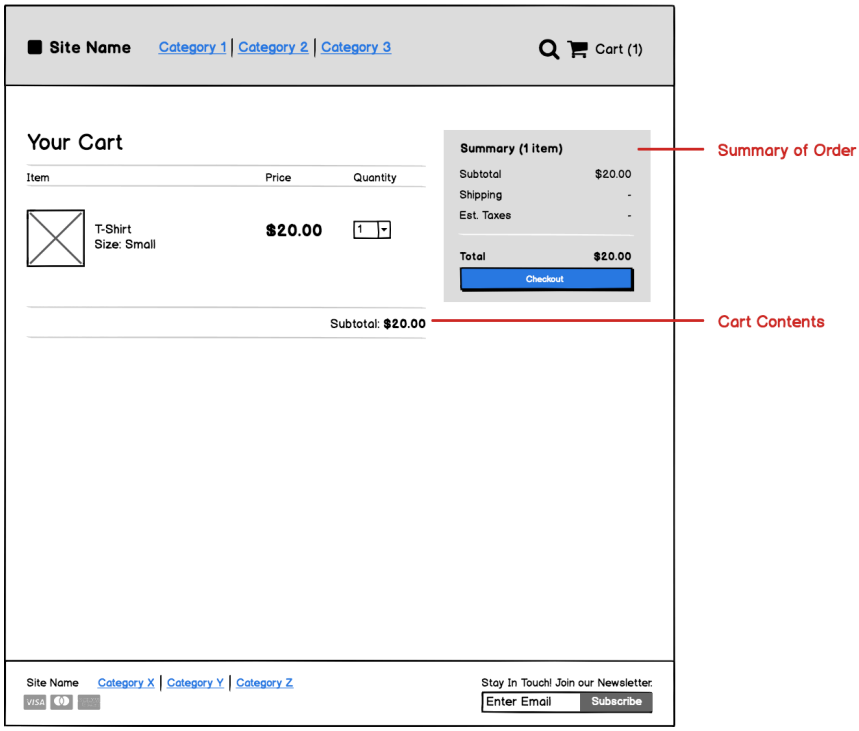
Product Page



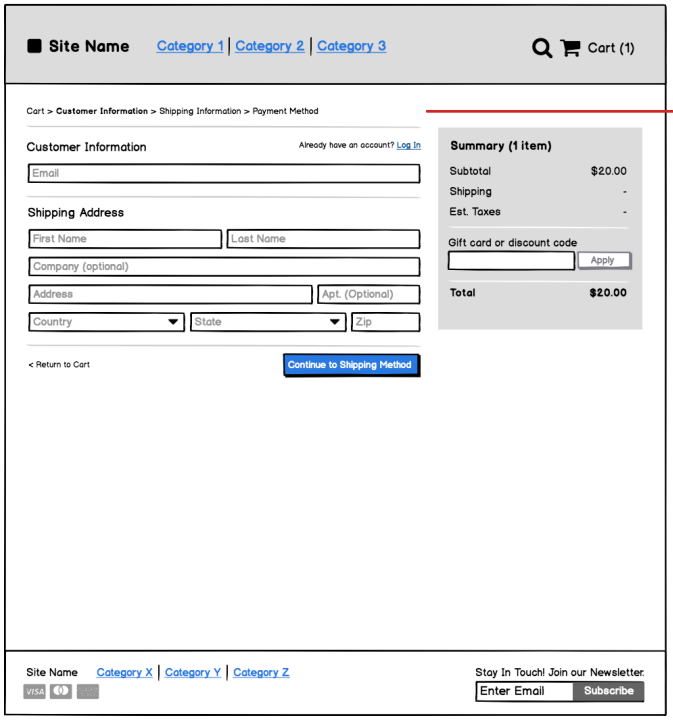
Product Page - Add To Cart



Cart



Checkout - Customer Info



Practical 7: Software Project Management :Activity Scheduling

SPMP- Software Project Management Plan /Gantt Chart

AIM: Prepare the Time Line and Activity Scheduling using Turbo Project, Microsoft Project Tool

Prepare Software **Project Plan Document/Gantt Chart** for the Project

---

Roles and Responsibilities

Software Engineer: will plan, analyze and design the software project during preparing the documentations of project.
Programmer:will write the code of the system.
Test Team Member:will test the system to augment the quality.
Training Coordinator: will prepare the stuff-training plan to ensure that necessary skill levels in sufficient numbers will be available to successfully conduct the software project.
Meeting Tracer: will organize and coordinate the meetings of team members and also meetings with customers.
DB Administrator: will be responsible from database configuration and management.
Software System Engineer: will help the project team to identify, control, and track requirements and changes to requirements at any time as the project proceeds and make the architectural design of the project accordingly.
Project Manager: will plan (schedule, cost and budget), motivate, organize and control the software team.
User interface designer: will design the web-based user interfaces.
Configuration Manager: will manage different versions of the work products, control the changes that are imposed and audit and report on the changes that are made. And also will update the project's web page regularly.

Name	E-mail	Roles and Responsibilities
		Project Manager, Software Engineer, Configuration Manager, Programmer, DB Administrator, Meeting Tracer, Test Team Member, Training Coordinator.
		Software Engineer, Programmer, DB Administrator, Test Team Member, User Interface Designer.
		Software Engineer, Programmer, Test Team Member, Configuration Manager, Training Coordinator.
		Software System Engineer, Software Engineer, Programmer, DB Administrator, Test Team Member, Programmer, Configuration Manager

Work Plan

Work Activities

Work activities of project are given in table 10. The milestone tasks are identified at the third column by *Milestone mark.

WBS No:	Task Name:	Milestone
1.	**Project**	
1.1	Problem Analysis	
1.1.1	Meeting with the Customer	
1.1.2	Determining the Problems	
1.2	**Initial Plan**	
1.2.1	Studying about IEEE std 1058	
1.2.2	Determining Cases for Initial Plan	
1.2.3	Documenting Initial Plan	
1.2.4	Discussion about issues of Initial Plan	
1.2.5	Delivery of the first version of Initial Plan	*Milestone
1.2.6	Feedback of Initial Plan	
1.2.7	Delivery of updated Initial Plan	*Milestone
1.3	**Software Requirement Specification**	
1.3.1	Studying about IEEE std 830	
1.3.2	Determining actors and use cases	
1.3.3	Detailing use cases	
1.3.4	Documenting SRS	
1.3.5	Discussion about issues of SRS	
1.3.6	Delivery of the first version of SRS	*Milestone
1.3.7	Feedback of SRS	
1.3.8	Delivery of updated SRS	*Milestone
1.4	**Software Project Management Plan**	
1.4.1	Studying about IEEE std 1058	
1.4.2	Determining Cases for SPMP	
1.4.3	Documenting SPMP	
1.4.4	Discussion about issues of SPMP	
1.4.5	Delivery of the first version of SPMP	*Milestone
1.4.6	Feedback of SPMP	
1.4.7	Delivery of updated SPMP	*Milestone
1.5	**Software Design Description**	
1.5.1	Studying about IEEE std 1016	
1.5.2	Determining Design Entities	
1.5.3	Determining Design Entity Attributes	
1.5.4	Drawing Design Views	
1.5.5	Documenting SDD	
1.5.6	Discussion about issues of SDD	
1.5.7	Delivery of the first version of SDD	*Milestone
1.5.8	Feedback of SDD	
1.5.9	Delivery of updated SDD	*Milestone
1.6	**Coding**	
1.6.1	Determining the Coding Standard	
1.6.2	Coding of General Interfaces	
1.6.3	Coding of Infrastructure of Project	
1.6.4	Code Review of General Interfaces	
1.6.5	Testing of Code side of Product	
1.6.6	Meeting with Customer about the Product	
1.6.7	Update and Delivery of Product	*Milestone

1.7	**User Manual**	
1.7.1	Determining Cases for UM	
1.7.2	Documenting UM	
1.7.3	Discussion about issues of UM	
1.7.4	Delivery of the first version of UM	*Milestone

Work packages for each activity are given below:

Activity Number	1.1.1
Activity Code	M-Cstm.
Activity Name	Meeting with the Customer
Estimated Duration/Effort	0.12 day/ 1.30 man-hour
Resources Needed	**Personnel:** Team Members **Skill:** Communication **Tool:** Notebook **Travel**: N/A
Outputs	Borders of the Project
Baselines	No
Predecessors	
Successors	1.1.2
Completion Criteria	Confirmation of the Customer
Implementation Personnel Assigned Starting Date Completion Date Cost (Budgeted and Actual) Legacy Comments	 Team Members 08 October 06 08 October 06 NA None

Resource Allocation

Group members have the general information about design techniques of .Net language. PCs will be used for documentations of the reports that are IP, SRS, SPMP and SDD and coding. Table below shows software tools to be used by project phase.

WBS No	WBS Item	SOFTWARE RESOURCES
1.2	Initial Plan	MS Word, MS Project, MS Visio, Photoshop
1.3	Software Requirement Specification	MS Word, MS Visio
1.4	Software Project Management Plan	MS Word, MS Project, MS Visio
1.5	Software Design Description	MS Word, MS Visio, Photoshop
1.6	Coding	MS Word, Visual Studio 2005 Professional Edition, ASP .NET
1.7	User Manual	MS Word

Risk Management

Risks	Category	Probability (%)	Impact
Uncertainty in the format of PS file	Process definition	90	Moderate

Development tool unfamiliar	Staff experience	70	High
Not enough time for SW integration	Business impact	87	Moderate
Staff inexperienced	Staff	72	High
Absentees of personnel could result in loss of effort	Staff	75	High
Design and coding deficiencies.	Process definitions	70	High
Expected technical changes	Development environment	40	Moderate
Documentation	Process definition	30	Moderate
May have to train the trainer	Staff	55	Low
Incorrect and missing Work package definition	Process definition	45	High
Optimistic schedule for HW/SW integration	Business impact	25	Low
Less reuse than planned	Product size	25	Moderate
Customer may change requirement	Product size	20	Critical
Misunderstood and/or undetermined requirements	Product size	20	Critical
Process demands not adequately planned in estimates	Product size	15	High
Inappropriate metrics	Product size	18	Moderate

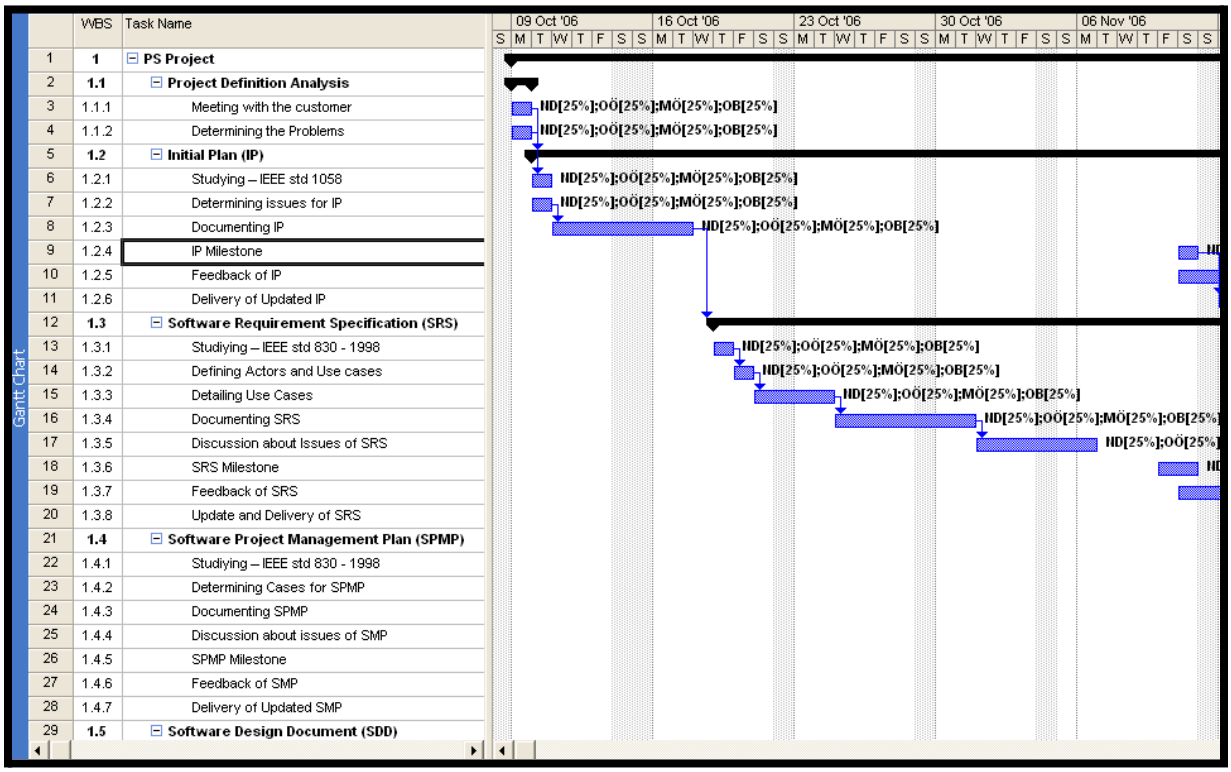
Risk Mitigation

PRIORITY	Risk Scenarios	Risk Alternatives / Resolutions	Monitoring
1	Environment: not suitable for team communication.	**Alternatives:** Choose a work environment that allows team communication. **Resolution:** Optimizing the environment situation.	During the group meetings.
1	Group deficiency: One of the members is sick.	**Resolution:** Other group members are over work.	Dead line is very close.
4	Communication skill is not enough: In the code phase, the new requirements will be come.	**Alternatives:** Group willmake a meeting with customers.	Middle of the coding phase.
2	The development tool is hard to use.	**Alternatives:**If project member is not enough knowledge about development tool, they will take a related info from project training coordinator. **Resolution:** Before implementation team will make mini projects with development tool.	During the implementation phase.

Project Development and Document/Milestone Schedule

WP NO	Who Will Prepare	Start Date	Due Date	Days	Dependency on Task	Milestone Document Name
W1						Initial Plan
W2					W1	Software Requirement Specification
						Software Project Management Plan
						Software Design Description
						Coding
						User Manual
						Final Product

Project Schedule



Practical 8: Software Project Cost Estimation

AIM: Calculate the Various Costs using CoCoMo, Function Point ,Algorithmic Cost Modeling Methods

---

Estimation Plan

The cost and schedule for conducting the project (PS) as well as the methods, tools, techniques, used to estimate project cost, schedule, resource requirements, external input and output data associated confidence levels are presented below.

Function Point method is used for size estimation of the project. The functions of the project are identified according to the user requirements defined during the initial meetings. These functions are:

External inputs:

- New Patient Information Add
- New Patient Information Delete
- New Patient Information Update
- Patient Disease Identification and Therapy Information Add
- Patient Disease Identification and Therapy Information Delete
- Patient Disease Identification and Therapy Information Update
- Drug Information Add
- Drug Information Delete
- Drug Information Update
- Drug Stock Input-Output Information Add
- Drug Stock Input-Output Information Delete
- Drug Stock Input-Output Information Update
- Disease Information ICP-10 Add
- Disease Information ICP-10 Delete
- Disease Information ICP-10 Update

External Output:

- Patient Registration
- Drug Registration
- Stock Registration
- Warning Message

External Enquiry:

- Access to Patient Records
- Access drugs which are given to patient before
- Access to Drugs Records
- Access to stock records
- Access number of decreasing drugs records
- Access to drug names and codes records

External Interface Files:

- Wireless network interface

Internal Logical File:

- Doctor Table

- Patient Table
- Drug Table
- Disease Names Table
- Disease Groups Table

		Function Levels			
Components	**Count**	**Low**	**Average**	**High**	**Total**
External Inputs	**16**	**9** x 3	6 x 4	1 x 6	54
External Outputs	**4**	**2** x 4	2 x 5	0 x 7	18
External Inquiries	6	4 x 3	2 x 4	1 x 6	26
Internal Logical Files	5	5x 7	0 x 10	0 x 15	35
External Interface Files	1	0 x 5	0 x 7	1 x 10	10
				Total Unadjusted FP (UFP):	143

Table 1 Unadjusted Function Point Table

In Table 1, the Degree of Influences (DI) is shown:

Characteristic	**Degree of Influence (DI)**	**Description**
1. Data Communication	2	Output Data Entry is required
2. Distributed data processing	4	Online and in both directions
3. Performance	3	Response time is critical
4. Heavily used configuration	2	Security and timing issues
5. Transaction rate	4	Service level agreements are high level
6. Online data entry	5	More than %90 is online data entry
7. End user efficiency	3	Six of more of the elements
8. Online update	5	Data lost is essential
9. Complex processing	3	Three of the elements
10. Reusability	4	Modulated coding
11. Installation ease	1	Only server-side installation
12. Operational ease	1	Minimizes the need of paper handling
13. Multiple sites	1	Identical hardware and software
14. Facilitate change	3	Three of the elements
Total Degree of Influence (TDI):	41	

Table 2 Degree of Influences

The scale of criteria weights for the table 1 (TDI):

0	**1**	**2**	**3**	**4**	**5**
No influence	**Incidental**	**Moderate**	**Average**	**Significant**	**Essential**

The table shown above lists the replies of the system to the specific questions depending on the IFPUG documents.

Three Formulas for Size Estimation of the FIS;

- ***Total Degree of Influence (TDI) = Sum of the Influence Factors***
- ***Value Adjustment Factor (VAF) =0.65 + (0.01 * TDI)***
- ***Function Points (FP) = VAF * unadjusted FP***

Value Adjustment Factor (**VAF**) = (TDI x 0.01) + 0.65 = (41 x 0.01) + 0.65 = **1.06**
Adjusted Function Point Count (**FP**) = UFP x VAF = 1.06 x 143 = **151.58**

The number of lines of code is then defined by:

Line of Code = (SLOC per FP for Java) x FP = 30 x 151.58 = 4547.4

Effort Estimation

An LOC-oriented estimation model is used for effort estimation.The number of lines of code is then defined by:

***LOC = FPx Language Factor*(Language Factor vary by Language of Programming)**

KLOC = 4.547 (4547.4 LOC)

Then the effort is:

Effort (Coding) = a x Size b

Development Time (T) =c*(Effort)^d

a and b are constants that depend on the project, and size is measured in thousands of lines of code (KLOC).

Software Project	a	b	c	D
Organic	3.2	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Because of the decision, that the software project has an "***organic"*** structure.

a = 3.2 is determined for Organic Project
b = 1.05 is determined for Organic Project
c = 2.5 is determined for Organic Project
d = 0.38 is determined for Organic Project

E = 3.2 x 4.547 $^{1.05}$= 3.2 x 4.90 = 15.68 (man month)

COCOMO model is used to calculate the schedule estimation for the PS. COCOMO model calculates effort in man months.

COCOMODevelopment Time (T) =c*(Effort Applied)^d = 2.5 * (15.68) ^ 0.38 ≈ 7.11 Months

"c " and 'd' depends on the modes of the difficulty so that d is decided as 0.38 as it is for the other organic software projects.COCOMO model calculates effort in staff months (19 days per month or 152 working hours per month).

Therefore such a software project needs:

Staff = Effort / Duration = 15.68 / 7.1 = **2.20** persons

Cost Estimation

The salary of each group member is 40000Rs./p.m @152 Hr p.m, therefore the estimated cost of the Project is:

15.68 man month = 15.68 x 152 (hours in a month) = 2383.36 man hours

Cost = Effort x $ (the hourly salary)

C = 2383.36 man-hours x (40000/152)man-hours=6,27,200Rs.

Practical 9: Software Testing: Project Test Plan Document

AIM: Design Test Cases and Scenarios for Testing Software as a parts and as a whole

Prepare Software **Project Test Plan Document** for the Project

---

1. Test Cases need to be simple and transparent:
Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them. Use assertive language like go to the home page, enter data, click on this and so on. This makes the understanding the test steps easy and tests execution faster.

2. Create Test Case with End User in Mind
The ultimate goal of any software project is to create test cases that meet customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective

3. Avoid test case repetition.
Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column

4. Do not Assume
Do not assume functionality and features of your software application while preparing test case. Stick to the Specification Documents.

5. Ensure 100% Coverage
Make sure you write test cases to check all software requirements mentioned in the specification document. Use Traceability Matrix to ensure no functions/conditions is left untested.

6. Test Cases must be identifiable.
Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.

7. Implement Testing Techniques
It's not possible to check every possible condition in your software application. Software Testing techniques help you select a few test cases with the maximum possibility of finding a defect.

- **Boundary Value Analysis (BVA):** As the name suggests it's the technique that defines the testing of boundaries for a specified range of values.
- **Equivalence Partition (EP):** This technique partitions the range into equal parts/groups that tend to have the same behavior.
- **State Transition Technique**: This method is used when software behavior changes from one state to another following particular action.
- **Error Guessing Technique:** This is guessing/anticipating the error that may arise while doing manual testing. This is not a formal method and takes advantages of a tester's experience with the application

8. **Self-cleaning**
The test case you create must return the Test Environment to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

9. **Repeatable and self-standing**
The test case should generate the same results every time no matter who tests it

10. Peer Review.
After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.

Popular Test Management tools are: Quality Center and JIRA

(Minimum 15 Critical Test Cases)

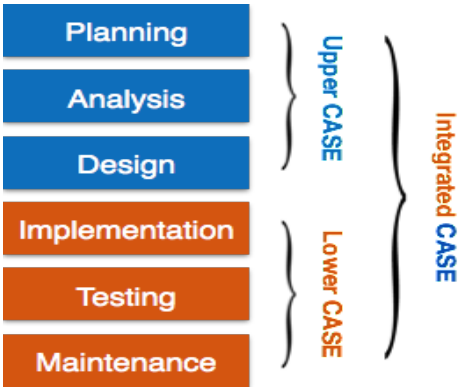
TestCaseID	Test Scenario: Check Customer Login with valid Data	Pass/Fail
T1	**Test Steps:** 1. Go to site Enter UserId 2. Enter Password 3. Click SubmiT	Pass
	Test Data: Userid = abc99 Password = pass123	
	Expected Results: User should Login into an application	
	Actual Results: As Expected	
T2	**Test Scenario**	
	Test Steps:	
	Test Data:	
	Expected Results:	
	Actual Results:	

Practical 10: Case study on CASE Tools for Software Processes

AIM: Case study on CASE Tools for Software Processes

Project group hase to prepare case study of one of the CASE Tool

Central Repository - CASE tools require a central repository, which can serve as a source of common, integrated and consistent information. Central repository is a central place of storage where product specifications, requirement documents, related reports and diagrams, other useful information regarding management is stored. Central repository also serves as data dictionary.



Upper Case Tools - Upper CASE tools are used in Planning, Analysis and Design stages of SDLC.
Lower Case Tools - Lower CASE tools are used in Implementation, Testing and Maintenance.
Integrated Case Tools - Integrated CASE tools are helpful in all the stages of SDLC, from Requirement Gathering to Testing and Documentation.
CASE tools can be grouped together if they have similar functionality, process activities and capability of getting integrated with other tools.
Scope of Case Tools: The scope of CASE tools goes throughout the SDLC.

Case Tools Types

A. Diagram tools

These tools are used to represent system components, data and control flow among various software components and system structure in a graphical form. For example, **Flow Chart Maker tool** for creating state-of-the-art flowcharts.

B. Process Modeling Tools

Process modeling is method to create software process model, which is used to develop the software. Process modeling tools help the managers to choose a process model or modify it as per the requirement of software product. For example, **EPF Composer**

C. Project Management Tools

These tools are used for project planning, cost and effort estimation, project scheduling and resource planning. Managers have to strictly comply project execution with every mentioned step in software project management. Project management tools help in storing and sharing project information in real-time throughout the organization. For example, **Creative Pro Office, Trac Project, Basecamp**.

D. Documentation Tools

Documentation in a software project starts prior to the software process, goes throughout all phases of SDLC and after the completion of the project.Documentation tools generate documents for technical users and end users. Technical users are mostly in-house professionals of the development team who refer to system manual, reference manual, training manual, installation manuals etc. The end user documents describe the functioning and how-to of the system such as user manual. For example, **Doxygen, DrExplain, Adobe RoboHelp** for documentation.

E. Analysis Tools

These tools help to gather requirements, automatically check for any inconsistency, inaccuracy in the diagrams, data redundancies or erroneous omissions. For example**, Accept 360, Accompa, CaseComplete for requirement analysis, Visible Analyst for total analysis**.

F. Design Tools

These tools help software designers to design the block structure of the software, which may further be broken down in smaller modules using refinement techniques. These tools provides detailing of each module and interconnections among modules. For example, **Animated Software Design**

G. Configuration Management Tools

An instance of software is released under one version. Configuration Management tools deal with **–Version and revision management, Baseline configuration management,Change control management.** CASE tools help in this by automatic tracking, version management and release management. For example, **Fossil, Git, Accu REV.**

H. Change Control Tools

These tools are considered as a part of configuration management tools. They deal with changes made to the software after its baseline is fixed or when the software is first released. CASE tools automate change tracking, file management, code management and more. It also helps in enforcing change policy of the organization.

I. Programming Tools

These tools consist of programming environments like IDE (Integrated Development Environment), in-built modules library and simulation tools. These tools provide comprehensive aid in building software product and include features for simulation and testing. For example, Cscope to search code in C, Eclipse.

J. Prototyping Tools

Software prototype is simulated version of the intended software product. Prototype provides initial look and feel of the product and simulates few aspect of actual product.Prototyping CASE tools essentially come with graphical libraries. They can create hardware independent user interfaces and design. These tools help us to build rapid prototypes based on existing information. In addition, they provide simulation of software prototype. For example, **Serena prototype composer, Mockup Builder.**

K. Web Development Tools

These tools assist in designing web pages with all allied elements like forms, text, script, graphic and so on. Web tools also provide live preview of what is being developed and how will it look after completion. For example, **Fontello, Adobe Edge Inspect, Foundation 3, Brackets.**

L. Quality Assurance Tools

Quality assurance in a software organization is monitoring the engineering process and methods adopted to develop the software product in order to ensure conformance of quality as per organization standards. QA tools consist of configuration and change control tools and software testing tools. For example, **SoapTest, AppsWatch, JMeter**.

M. Maintenance Tools

Software maintenance includes modifications in the software product after it is delivered. Automatic logging and error reporting techniques, automatic error ticket generation and root cause Analysis are few CASE tools, which help software organization in maintenance phase of SDLC. For example, **Bugzilla for defect tracking, HP Quality Center**.

Practical 11: Software Engineering Case Study Review.

AIM: Review of Website/ Desktop Application from Software Engineering Perspectives.

Name of Desktop/Web /Mobile Application :

Segment : (like Social Media/ Banking/ Gaming/ Trading/ Shopping/ etc)

Major Functionalities:

Ease of User Interface:

No of Users(Approx.) :

Targeted Users with Types :(Skilled Professionals/ Novice/ETC)

Input: Data/Image/Information

Output: Transaction/Reports/Statistics/etc

No of Screens/Pages/Cards(Approx.) :

MediaUpload/Download: Images/Audio/Video/Animations

Money Payment Interface(If Any):

Query/ Feedback/Complaint Resolution:

FAQs Sections:

User Manual (If Any):

User Guidance(if any):

Type : Online/ Offline/ Mixed Mode

Update Mode:

Critical Point of Failures: (When application is consider as failure)

Security Features:

Alternatives/Similar Application (if Any):

Advantages over Competing Application: