Workbook

Software Design & Architecture
(SE – 301)

Name	
Roll No	
Batch	
Year	
Department	

Workbook

Software Design & Architecture (SE – 301)

Prepared by

Mrs. Nazish Irfan
I.T Manager, CS&IT

Approved by

Chairman
Department of Computer Science & Information Technology

Table of Contents

Lab # 1

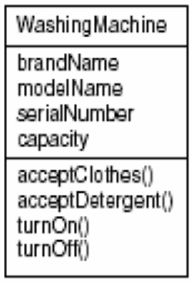
Object:
Introduction to Unified Modeling Language (UML).

Theory:
The Unified Modeling Language (UML) enables system builders to create blueprints that capture their visions in a standard, easy-to understand way, and provides a mechanism to effectively share and communicate these visions with others. It does this via a set of symbols and diagrams. Each diagram plays a different role within the development process.

Components of the UML
- The UML consists of a number of graphical elements that combine to form diagrams.
- UML is a language; it has rules for combining these elements.
- The purpose of the diagrams is to present multiple views of a system; this set of multiple views is called a **model.**
- A UML model tells what a system is supposed to do.

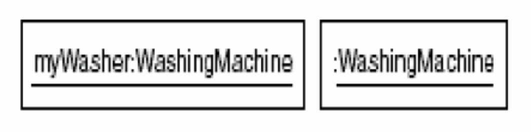
UML Diagrams
- Class Diagram
- Object Diagram
- Use Case Diagram
- State Diagram
- Sequence Diagram Activity Diagram
- Communication Diagram
- Component Diagram
- Deployment Diagram

Class Diagram:
A class is a category or group of things that have the same attributes and the same behaviors. A rectangle is the icon that represents the class. It's divided into three areas. The uppermost area contains the name, the middle area holds the attributes, and the lowest area holds the operations i.e. anything in the class of washing machines has attributes such as brand name, model, serial number, and capacity. Behaviors for things in this class include the operations "accept clothes," "accept detergent," "turn on," and "turn off."

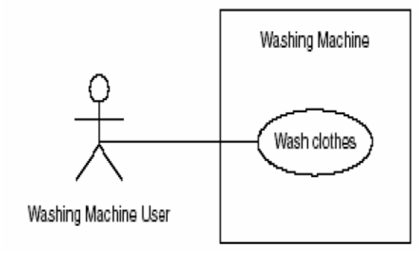


In UML, a multiword class name has initial capital letters for all the words and eliminates white space between each word (for example, WashingMachine). Attribute names and operation names follow the same convention, but the first letter of the first word isn't capitalized (for example, acceptClothes ()).

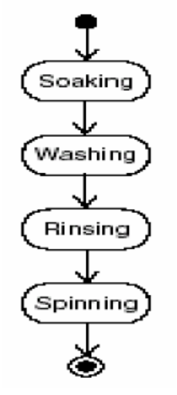
Object Diagram:
An object is an instance of a class—a specific thing that has specific values of the class's attributes. The icon is a rectangle, just like the class icon, but the name is underlined. In the icon on the left, the name of the specific instance is on the left side of a colon, and the name of the class is on the right side of the colon. The name of the instance begins with a lowercase letter. It's also possible to have an anonymous object, as the icon on the right of Figure shows. This just means that you don't supply a specific name for the object, although you do show the class it belongs to.



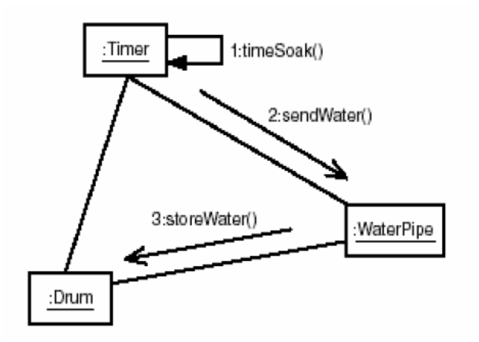
Use Case Diagram:
A use case is a description of a system's behavior from a user's standpoint. For system developers, the use case is a valuable tool: It's a tried-and-true technique for gathering system requirements from a user's point of view. Obtaining information from the user's point of view is important if the goal is to build a system that real people can use.



State Diagram:
At any given time, an object is in a particular state. This transition from one state to the next state is represented by state diagram i.e. a person can be a newborn, infant, child, adolescent, teenager, or adult. An elevator is either moving or stationary. A washing machine can be either in the soaking, washing, rinsing, spinning, or off state. The symbol at the top of the figure represents the start state and the symbol at the bottom represents the end state.

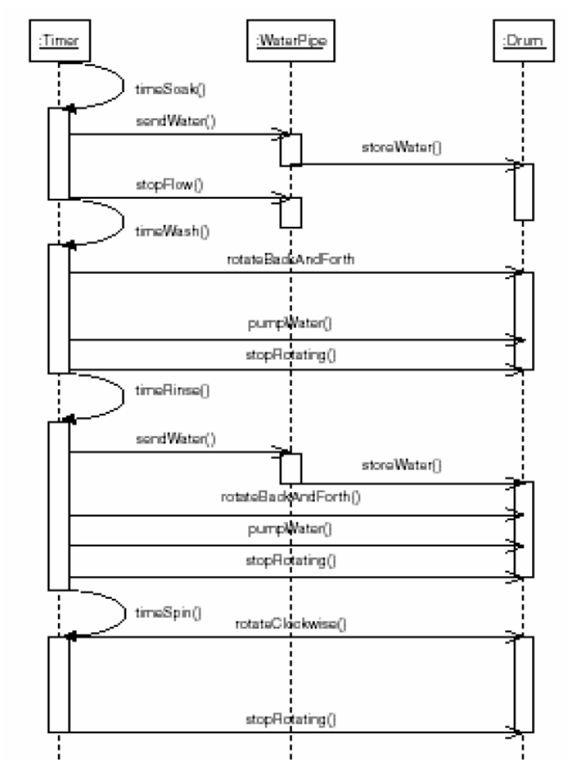


Communication Diagram:
Both the sequence diagram and the communication diagram show interactions among objects. For this reason, the UML refers to them collectively as interaction diagrams.



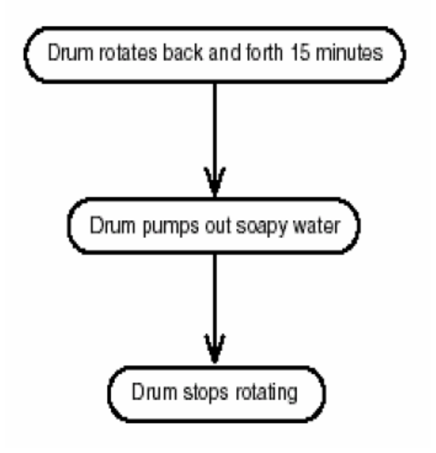
Rather than represent time in the vertical dimension, this diagram shows the order of messages by attaching a number to the message label.

Sequence Diagram:
Objects interact with one another, and these interactions occur over time. The UML sequence diagram shows the time-based dynamics of the interaction.

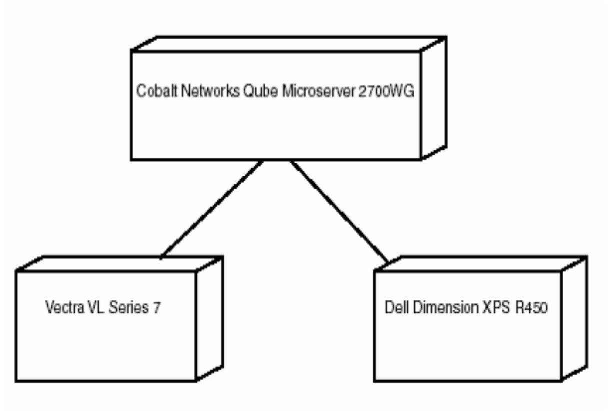


Each arrow represents a message that goes from one object to another. Time, in this diagram, proceeds from top to bottom. So the first message is timeSoak (), which the timer sends to itself. The second message is sendWater (), which the timer sends to the water pipe. The final message, stopRotating (), goes from the timer to the drum

Activity Diagram:
The activities that occur within a use case or within an object's behavior typically occur in a sequence is represented by the activity diagram.


Drum rotates back and forth 15 minutes
Drum pumps out soapy water
Drum stops rotating

Deployment Diagram:
The UML deployment diagram shows the physical architecture of a computer based system. It can depict the computers, show their connections with one another, and show the software that sits on each machine. Each computer is represented as a cube, with interconnections between computers drawn as lines connecting the cubes.



Exercise:

1. List the advantages of Unified Modeling Language.

Lab # 2

Object:
Introduction to Rational Rose.

Theory:
Once the Requirements have been gathered, it is necessary to convert them into an appropriate design. In order to bridge the requirements gathering and design phases, we use some modeling or specification tool, e.g. the Unified Modeling Language. A commonly available CASE tool for design through UML is the Rational Rose. Given below is an overview of the various aspects in which development could be done using Rational Rose.

The Browser.
The Browser contains a list of all the modeling elements in the current model. The Browser contains a tree view of all the elements in the current Rose model. Elements in the tree structure are either compressed or expanded.

The Documentation Window.
The Documentation Window may be used to create, review, and modify for any selected modeling element. If nothing is selected, the window displays the string “No selection”. The contents of the Documentation Window will change as different modeling elements are selected.

To create the documentation for a selected element, click to select the element and enter its documentation in the Documentation Window.

The Documentation Window may be docked or floating just like the Browser. Visibility of the window is controlled via the View: Documentation window command.

Diagram Windows.
Diagram windows allow you to create and modify graphical views of the current model. Each icon on a diagram represents a modeling element. Since diagrams are used to illustrate multiple views of a model, each model element can appear in none, one, or several of a model's diagrams.

Views in Rational Rose.
There are many views of a software project under development. Rational Rose is organized around the views of a software project (4 + 1 View):
- The Use Case View.
- The Logical View.
- The Component View.
- The Deployment View.

Each view presents a different aspect of the visual model. Each view contains a Main diagram by default. Additional elements and diagrams are added to each view throughout the analysis and design process.

The Use Case View.
The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

Packages in the Use Case View.
Packages in the use case view can contain actors, use cases, sequence diagrams, and/or collaboration diagrams. To create a package:

1. Right-click on the parent modeling element (use case view or another package) to make the shortcut menu visible.
2. Select the New: Package menu command. This will add a new package called NewPackage to the browser.
3. While the new package is still selected, enter its name.

Once a package is created, modeling elements may be moved to the package. To move a modeling element to a package:
1. Click to select the modeling element to be relocated.
2. Drag the modeling element to the package.

The Logical View.
The logical view of the system addresses the functional requirements of the system. This view looks at classes and their static relationships. It also addresses the dynamic nature of the classes and their interactions. The diagrams in this view are class diagrams and state transition diagrams.

Class Diagram.
The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by double-clicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.

State Transition Diagram.
State transition diagrams show the life history of a given class, the events that cause a transition from a state and the actions that result from a state change. They are created for classes whose objects exhibit significant dynamic behavior.

The Component View.
The component view of the system addresses the software organization of the system. This view contains information about the software, executable and library components for the system. This view contains component diagram.

Component Diagram.
A component diagram shows the organizations and dependencies among components. Most models contain many component diagrams. A component is represented as a rectangle with one small ellipse and two small rectangles protruding from its side.

Rose automatically creates one component diagram called Main. To create an additional component diagram:

1. Right-click on the owning package (either the component view itself or a user created package) to make the shortcut menu visible.
2. Select the New: Component Diagram menu command. This will place a new component diagram called NewDiagram in the browser.
3. While the new diagram is still selected, enter its name.

Rose will automatically add the new diagram to the browser.

To open a component diagram:
1. Double-click on the diagram in the browser.

To create a component:
1. Click to select the package specification icon from the toolbar.
2. Click on the component diagram to place the component.
3. While the component is still selected, enter its name.

Rose will automatically add the new component to the browser.

To create a dependency relationship:
1. Click to select the dependency icon from the toolbar.
2. Click on the package or component representing the client.
3. Drag the dependency arrow to the package or component representing the supplier.

The Deployment View.
The deployment view addresses the configuration of run-time processing nodes and the components, processes, and objects that live on them. This view contains only one diagram – the deployment diagram.

Deployment Diagram
The deployment diagram contains nodes and connections. Rose provides two icons that can be used to draw a node – the processor and the device. A processor is a node that has processing capacity.

To open the deployment diagram:
1. Double-click on the Deployment View in the browser.

To create a node:
1. Click to select the processor icon from the toolbar.
2. Click on the deployment diagram to place the node.
3. While the node is still selected, enter its name.

To create a connection:
1. Click to select the connection icon from the toolbar.
2. Click on the node representing the client.
3. Drag the connection line to the node representing the supplier.

Exercise:

1. What is 4 + 1 view? Briefly define each view.

Lab # 3

Object:

To understand Use Case View.

Theory:

The use case view of the system addresses the understandability and usability of the system. This view looks at actors and use cases along with their interactions. The diagrams in this view are use case diagrams, sequence diagrams and collaboration diagrams.

Actor

An actor is represented by a stickman. To create an actor:

1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Actor menu command. This will add an actor called NewClass to the browser.
3. While the new class is still selected, enter the name of the actor.

As actors are created, their documentation is added to the model. To add the documentation for an actor:

1. Click to select the actor in the browser.
2. Position the cursor in the Documentation Window.
3. Enter the documentation for the actor.

Each modeling element is associated with a Specification window. The Specification contains additional information about the element. To view the Specification for an actor:

1. Right-click to select the actor in the browser and make the shortcut menu visible.
2. Select the Specification menu command.

When you view the Specification for an actor, you will notice that the title is "Class Specification for <actor>". This is due to the fact that an actor is a class in Rose with a stereotype /*Extension of Metaclass*/ of Actor.

Use Case.

A use case is represented by an oval. To create a use case:

1. Right-click to select the Use Case View in the browser and make the shortcut menu visible.
2. Select the New: Use Case menu command. This will add an unnamed use case to the browser.
3. While the new use case is still selected, enter the name of the use case.

Use cases are documented in two ways – they have a brief description and a flow of events. To add the brief description for a use case:

1. Click to select the use case in the browser.
2. Position the cursor in the Documentation Window.
3. Enter the brief description for the use case.

The flow of events is captured in documents and/or web pages external to Rose. The documents and web pointers are linked to the use case. To link an external document or web pointer to the use case:

1. Right-click to select the use case in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Files tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert File menu command and enter the name of the document to be linked to the use case.
6. Or, select the Insert URL menu command and enter the www pointer to be linked to the use case.
7. Click the Open button.

Use Case Diagrams.

A use case diagram shows the relationships among actors and use cases within the system. The use case view is automatically created with one use case diagram called Main. A system may have other use case diagrams (e.g. a diagram to show all the use cases for one actor).

To open a use case diagram:

1. Click the + next to the Use Case View in the browser to expand the view.
2. Double-click on the diagram you wish to open.
3. Re-size the diagram as needed.

To add an actor or use case to the diagram:

1. Click to select the actor or use case in the browser.
2. Drag the actor or use case to the diagram.

The main type of relationship is a communication relationship. This relationship is shown as a uni-directional association. To add communication relationships to the diagram:

1. Click to select the uni-directional icon from the toolbar.
2. Click on the actor or use case initiating the communication.
3. Drag the relationship line to the participating use case or actor.

A use case diagram may have two other types of relationships – extends and uses relationships. Extends and uses relationships are shown as generalization relationships with stereotypes. To create an extends or uses relationship:

1. Click to select the generalization icon from the toolbar.
2. Click on the using or extension use case.
3. Drag the relationship line to the used or extended use case.
4. Double-click on the relationship line to make the Specification visible.
5. Enter “uses” or “extends” in the Stereotype field.
6. Click the OK button to close the Specification.
7. Right-click on the generalization line to make the shortcut menu visible.
8. Select the Show Stereotype menu command.

Actors and use cases may also be created on a use case diagram. To create an actor or use case on the diagram:

1. Click to select the actor icon or the use case icon from the toolbar.
2. Click on the diagram to place the actor or use case.
3. While the actor or use case is still selected, enter its name.

The actor or use case is automatically added to the browser.

Sequence Diagrams.
Use case diagrams present an outside view of the system. The functionality of a use case is captured in its flow of events. Scenarios are used to describe how use cases are realized as interactions among societies of objects. Scenarios are captured in sequence diagrams. Sequence diagrams are associated with use cases.

To create a new sequence diagram:
1. Right-click on the use case in the browser to make the shortcut menu visible.
2. Select the New: Sequence Diagram menu command. This will add a new sequence diagram called NewDiagram to the browser.
3. While the new diagram is still selected, enter the name of the diagram.
4. Double-click on the diagram in the browser to open it.
5. Re-size the sequence diagram as needed.

Sequence diagrams contain actors, objects and messages.
To add an actor to a sequence diagram:
1. Click to select the actor in the browser.
2. Drag the actor onto the diagram.

To add an object to the sequence diagram:
1. Click to select the object icon from the toolbar.
2. Click on the diagram to place the object.
3. While the object is still selected, enter its name.

To create a message:
1. Click to select the message icon from the toolbar.
2. Click on the object representing the client (sender of the message).
3. Drag the message to the object representing the supplier (receiver of the message).
4. While the message line is still selected, enter the name of the message.

To create a reflexive message:
An object may send a message to itself. This is called a reflexive message.
1. Click to select the message to self-icon from the toolbar.
2. Click on the object that needs a reflexive message.
3. While the message line is still selected, enter the name of the message.

Collaboration Diagrams.
A scenario may also be represented in a collaboration diagram. Like sequence diagrams, collaboration diagrams are associated with use cases.

To create a collaboration diagram directly from a sequence diagram:
Select the Browse: Create Collaboration Diagram menu command or use the F5 accelerator.

1. Re-size the diagram as needed.
2. Re-arrange the objects as needed.

Collaboration diagrams may also be created from scratch.

To create a new collaboration diagram:
1. Right-click on the use case in the browser to make the shortcut menu visible.
2. Select the New: Collaboration Diagram menu command. This will add a new collaboration diagram called NewDiagram to the browser.
3. While the new diagram is still selected, enter the name of the diagram.
4. Double-click on the diagram in the browser to open it.
5. Re-size the collaboration diagram as needed.

Collaboration diagrams contain actors, objects, links, messages, and optional data flows.

To add an actor to a collaboration diagram:
1. Click to select the actor in the browser.
2. Drag the actor onto the diagram.

To add an object to the Collaboration diagram:
1. Click to select the object icon from the toolbar.
2. Click on the diagram to place the object.
3. While the object is still selected, enter its name.

To add a link:
1. Click to select the link icon from the toolbar.
2. In the collaboration diagram, click on one actor or object that participates in the link.
3. Drag the link to the other participating actor or object.

An object may be linked to itself. This is called a reflexive link.

To create a link message:
1. Click to select the message to self-icon from the toolbar.
2. Click on the object in the collaboration diagram.

To create a message:
1. Click to select the message icon from the toolbar.
2. Click on the link between the communicating objects
3. While the message line is still selected, enter the name of the message.

To create a data flow:
1. Click to select the data flow icon from the toolbar.
2. Click to select the message it modifies.
3. While the data flow is still selected, enter the data flow information

Exercise:

1. Draw a Use-Case diagram for an Online Shopping System

Hint: Web Customer actor uses some web site to make purchases online. Use cases are **View Items**, **Make Purchase** and **Client Register**. View Items use case could be used by customer if he only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site through **<<Service>> Authentication** actor. Note, that **Checkout** use case is included use case not available by itself - checkout is part of making purchase.
View Items use case is extended by several optional use cases - customer may search for items and browse catalog.
Checkout use case includes **Customer Authentication**. It could be done through user login page (sign in), Remember me page. **Actor:** *Service Authentication*
Checkout use case also includes **Payment** use case which could be done either by using credit card and external credit payment service or with PayPal.(**Actors:** *Credit Payment Service, PayPal*)

2. Draw a sequence diagram for Hotel Reservation System

Hint: The object initiating the sequence of messages is **a Reservation window**. The **Reservation window** sends a makeReservation() message to a **HotelChain**. The **HotelChain** then sends a makeReservation() message to a **Hotel**. If the **Hotel** has available rooms, then it makes a **Reservation** and a **Confirmation**.

.

3. Draw communication diagram of an ATM machine. How user interact with the machine and how objects collaborate with each other to complete transaction.

Lab # 4

Object:
To understand Class Diagram.

Theory:
The browser provides a textual view of the classes in a system. Class diagrams are created to graphically view the classes and packages in the system. Rose automatically creates a class diagram called Main in the Logical View. This diagram may be opened by double-clicking on it in the browser. The Main class diagram typically contains packages thus, by the end of development it is a graphical view of the major architectural elements of the system. A package may be added to a class diagram by selecting it in the browser and dragging it onto the class diagram.

Each package typically has its own main diagram which is a picture of its key packages and classes. To create the Main class diagram for a package, double-click on the package on a class diagram. Once the Main diagram is created for a package, you can add packages and classes to the diagram by selecting them in the browser and dragging them onto the diagram.

Classes from any package may be added to a class diagram by selecting the class on the browser and dragging it onto the open class diagram. If the Show Visibility option is set to true either as a default using the Tool: Options menu or individually by using the shortcut menu for the class, the name of the "owning" package is displayed. The package name will be visible for all classes that do not belong to the package owning the class diagram.

Packages and classes may also be created using the class diagram toolbar. To create a package or class using the toolbar:

1. Click to select the icon (package or class) on the class diagram toolbar.
2. Click on the class diagram to place the package or class.
3. While the new package or class is still selected, enter its name.
4. Multiple packages or classes may be created by depressing and holding the Shift key.

Packages and classes created on class diagrams are automatically added to the browser.

To create a class diagram:
1. Right-click to select the owning package and make the shortcut menu visible.
2. Select the New: Class Diagram menu command. This will add a class diagram called NewDiagram to the browser.
3. While the new class diagram is still selected, enter its name.
4. To open the class diagram, double-click on it in the browser.

Class Structure.
The structure of a class is represented by its set of attributes. Attributes may be created in the browser, via the Class Specification or on a class diagram.

To create an attribute in the browser:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. To create an attribute, select the New: Attribute menu command. This will add an attribute called name to the browser.
3. While the new attribute is still selected, enter its name.
4. Attribute data types and default values may not be entered via the browser.

To create an attribute using the Class Specification:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Attributes tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an attribute called name.
6. While the new attribute is still selected, enter its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

To create an attribute on a class diagram:
1. Right-click to select the class on the class diagram and make the shortcut menu visible.
2. Select the Insert New Attribute menu command. This will insert an attribute in the form.

   name : type = initval
3. While the attribute is still selected, fill in its name. Type and initial value may be filled in at this time or you may choose to fill in this information later in the development cycle.

Attributes of a class may be viewed in the browser. The class will be initially collapsed. To expand the class to view its attributes, click the + next to the class.

Attributes should be documented. **To add the documentation for an attribute:**
1. Click to select the attribute in the browser.
2. Position the cursor in the Documentation Window. If the Documentation Window is not visible, select the View: Documentation menu command.
3. Enter the documentation for the attribute.

To delete an attribute:
1. Right-click to select the attribute in the browser or on the Attributes tab of the Class Specification and make the shortcut menu visible.
2. Select the Delete menu command.

Class Behavior.
The behavior of a class is represented by its set of operations. Operations may be created in the browser, via the Class Specification or on a class diagram. To create an operation in the browser:

1. Right-click to select the class in the browser and make the shortcut menu visible.
2. To create an operation, select the New: Operation menu command. This will add an operation called opname to the browser.

3. While the new operation is still selected, enter its name.
4. The operation signature may not be entered via the browser

To create an operation using the Class Specification:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Operations tab.
4. Right-click to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an attribute called opname.
6. While the new operation is still selected, enter its name. The signature and return value may be filled in at this time or you may choose to fill in this information later in the development cycle.

To create an operation on a class diagram:
1. Right-click to select the class on the class diagram and make the shortcut menu visible.
2. Select the Insert New Operation menu command. This will insert an attribute in the form
   opname (argname : argtype = default) : return
3. While the operation is still selected, fill in its name. The signature and return value may be filled in at this time or you may choose to fill in this information later in the development cycle
4. Operations of a class may be viewed in the browser. The class will be initially collapsed. To expand the class to view its operations, click the + next to the class.

To enter the signature of an operation:
1. Right-click to select the operation in the browser and make the shortcut menu visible.
2. Select the Specification menu command.
3. Select the Detail tab.
4. Right-click in the Arguments field to make the shortcut menu visible.
5. Select the Insert menu command. This will insert an argument called argname of type argtype with a default value of default.
6. Click to select the name, type, or default and enter the desired information.
7. Click the OK button to close the Specification.

To delete an operation:
1. Right-click to select the operation in the browser or on the Operations tab of the Class Specification and make the shortcut menu visible.
2. Select the Delete menu command.

Operations should be documented.

To add the documentation for an operation:
1. Click to select the operation in the browser.
2. Position the cursor in the Documentation Window. If the Documentation Window is not visible, select the View: Documentation menu command.
3. Enter the documentation for the operation.

Relationships.
Relationships provide a conduit for communication. There are four different types of relationships: association, aggregation, dependency, and inheritance. Relationships may only be created on a class diagram using the toolbar.

- In generalization, the child class is based on the parent class. This relationship indicates that the two classes are similar but have some differences.
- Association means that two classes are connected i.e. related in some way.
- Aggregation indicates a relationship between a whole and its part.
- Composition is a strong form of aggregation. In this kind of relationship each part may belong to only one whole. In this relationship when the whole is destroyed its parts are destroyed as well.
- Multiplicity shows the number of objects that can participate in a relationship.

Exercise:

1. Draw a class diagram for Retail Catalog Order

Hint: The central class is the **Order**. Associated with it is the **Customer** making the purchase and the **Payment**. A **Payment** is one of three kinds: **Cash, Check, or Credit**. The **Order** contains **OrderDetails**, each with its associated **Item**.
Note: Show relationship among classes.

Lab # 5

Object:
To understand State Transition Diagram.

Theory:
State transition diagrams show the life history of a given class, the events that cause a transition from a state, and the actions that result from a state change. They are created for classes whose objects exhibit significant dynamic behavior.

To create a state transition diagram:
1. Right-click to select the class in the browser and make the shortcut menu visible.
2. Select the State Diagram menu command.

To open a state transition diagram.
1. Click the + next to the class to expand the tree
2. Double-click on the State Diagram for the class

States.
A state is represented by an oval.

To create a state.
1. Click to select the State icon from the toolbar.
2. Click on the state transition diagram to place the state.
3. While the state is still selected, enter its name.

State Transitions.
A state transition is represented as an arrow which points from the originating state to the successor state.

To create a state transition.
1. Click to select the state transition arrow from the toolbar.
2. Click on the originating state and drag the arrow to the successor state.

State Actions.
Behavior that occurs while an object is in a state can be expressed in three ways: entry actions , activities , and exit actions . The behavior may be a simple event or it may be an event sent to another object.

To create an entry action, exit action or activity.
1. Point to the state and double click to make the State Specification dialog box visible.
2. Select the Detail tab.
3. Click-right in the Actions field to make the pop-up menu visible.
4. Select the Insert menu choice to insert a new action called entry.
5. Double-click on the action to make the State Action Specification visible.
6. If the action is a simple action, enter the name of the action in the Action field.
7. If the action is a send event action, enter the name of the event to be sent in the Send Event field. If the event has arguments, enter the arguments in the Send Arguments

field. Enter the name of the target object (object receiving the event) in the Send Target field.

8. Select the appropriate radio button in the When field (On Enty to create an entry action, On Exit to create an exit action, or Entry Until Exit to create an activity.
9. Click the OK button to close the Action Specification.
10. Click the OK button to close the State Specification.

Start and Stop States.
There are two special states associated with state transition diagrams – the start state and the stop state. The start state is represented by a filled in circle and the stop state is represented by a bull's eye.

To create a start state.
1. Click to select the start state icon from the toolbar.
2. Click on the diagram to place the start state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the start state and drag the state transition arrow to the successor state.

To create a stop state.
1. Click to select the stop state icon from the toolbar.
2. Click on the diagram to place the stop state on the diagram.
3. Click to select the state transition icon from the toolbar.
4. Click on the originating state and drag the state transition arrow to the stop state.

Exercise:

1. Draw State and Activity Diagram for Airline Reservation System. Show different activities involved for making Reservation, Changing Reservation and Passenger Checking in for flight.

2. Draw State and Activity Diagram for Online Shopping System. Online customer can browse or search items, view specific item, add it to shopping cart, view and update shopping cart, checkout, User can view shopping cart at any time. Checkout is assumed to include user registration and login.

Lab # 6

Object:
To understand System Modeling.

Theory:
Modeling consists of building an abstraction of reality. These abstractions are simplifications because they ignore irrelevant details and they only represent the relevant details (what is relevant or irrelevant depends on the purpose of the model).

Why Model Software?
Software is getting larger, not smaller; for example, Windows XP has more than 40 million lines of code. A single programmer cannot manage this amount of code in its entirety. Code is often not directly understandable by developers who did not participate in the development; thus, we need simpler representations for complex systems (modeling is a mean for dealing with complexity).

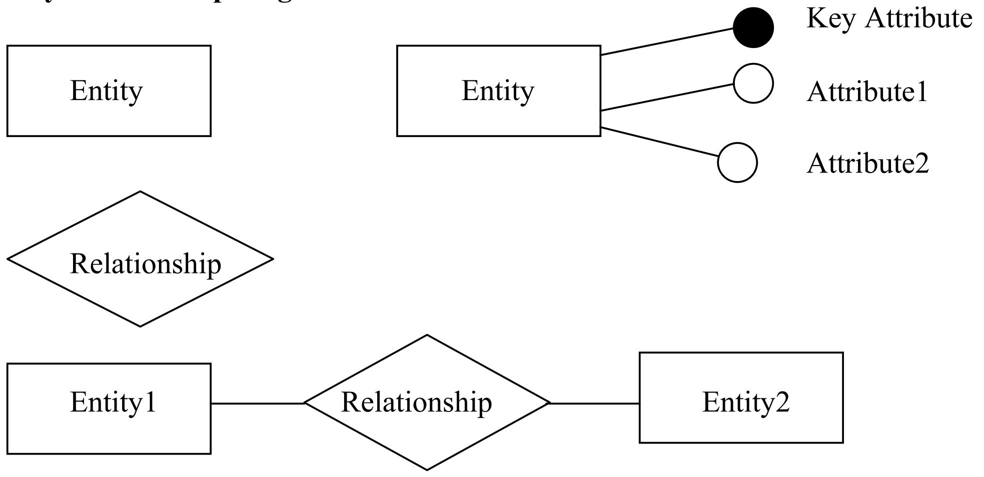
A wide variety of models have been in use within various engineering disciplines for a long time. In software engineering a number of modeling methods are also available.

Analysis Model Objectives.
- To describe what the customer requires.
- To establish a basis for the creation of a software design.
- To define a set of requirements that can be validated once the software is built.

The Elements of the Analysis Model.
The generic analysis model consists of:
- An entity-relationship diagram (data model).
- A data flow diagram (functional model).

Entity Relationship Diagram.
An entity relationship diagram (ERD) is one means of representing the objects and their relationships in the data model for a software product.

Entity Relationship diagram notation.



Creating an ERD.
Here are the steps you may follow to create an entity-relationship diagram.

Identify Entities.
Identify the entities. These are typically the nouns and noun-phrases in the descriptive data produced in your analysis. Do not include entities that are irrelevant to your domain.

Find Relationships.
Discover the semantic relationships between entities. These are usually the verbs that connect the nouns. Not all relationships are this blatant; you may have to discover some on your own. The easiest way to see all possible relationships is to build a table with the entities across the columns and down the rows, and fill in those cells where a relationship exists between entities.

Draw Rough ERD.
Draw the entities and relationships that you have discovered.

Fill in Cardinality.
Determine the cardinality of the relationships. You may want to decide on cardinality when you are creating a relationship table.

Define Primary Keys.
Identify attribute(s) that uniquely identify each occurrence of that entity.

Draw Key-Based ERD.
Now add them (the primary key attributes) to your ERD. Revise your diagram to eliminate many-to-many relationships, and tag all foreign keys.

Identify Attributes.
Identify all entity characteristics relevant to the domain being analyzed.

Map Attributes.
Determine which to entity each characteristic belongs. Do not duplicate attributes across entities. If necessary, contain them in a new, related, entity.

Draw fully attributed ERD.
Now add these attributes. The diagram may need to be modified to accommodate necessary new entities.

Check Results.
Is the diagram a consistent and complete representation of the domain.

Data Flow Diagram
A **data flow diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modeling its *process* aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

The data flow diagrams should also have some associated documentation. This is necessary as the diagrams are meant as a visual representation of the way in which information is processed. There is limited space on the diagrams so that documentation to explain, refine and describe further details of what is shown need to be kept somewhere in the proposed system documentation. The data flow diagrams and the associated documentation together combine to form a data flow model. This is also commonly called a process model.

The user requirements when complete are used as a basis for the development of the system. Later when the system has been developed it can be tested against the initial requirements to see whether the user's needs have been met.

Development and purpose of DFDs

Before attempting to construct an initial DFD it is necessary to gather and digest information that helps us to understand how data is processed in the current system. As the DFD is constructed a systems analyst will often come across areas of doubt where the precise way to model the system is unclear. This is a natural part of the development and should not be regarded with alarm. In fact, it is expected and it is a consequence of attempting to model the current situation that questions will be asked to clarify the exact processes which are taking place. Sometimes the analyst will make an assumption and then check this with the user at a subsequent meeting.

Results of interviews, documents, reports, questionnaires etc. will all play a part in helping the analyst to gain an insight into the current processes. Where a system is being developed from scratch the analyst will work with the user to develop the proposed DFDs. When all the information about the current system is gather it should be possible to construct the DFDs to show:

- the information that enters and leaves the system
- the people/roles/other systems who generate and/or receive that information
- the processes that occur in the system to manipulate the information
- the information that is stored in the system
- the boundary of the system indicating what is (and what is not) included

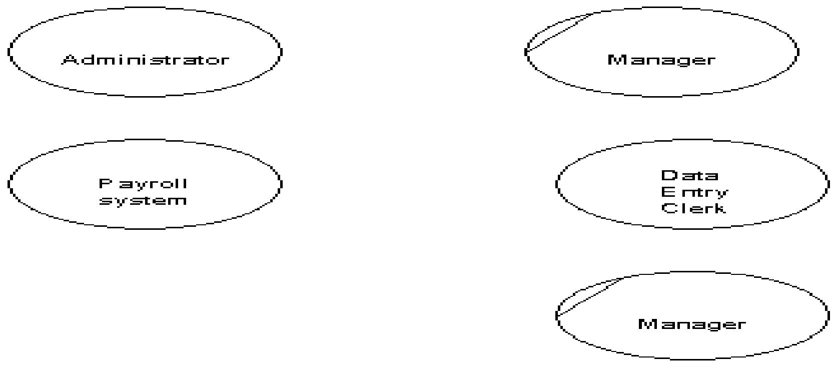
Components of Data Flow Diagrams

The components of a Data flow Diagram are always the same but there are different diagrammatic notations used. The notation used here is one adopted by a methodology known as SSADM (Structured Systems Analysis and Design Methods). There are only four different symbols that are normally used on a DFD. The elements represented are:

- External entities
- Processes
- Data stores
- Data flows

External Entities

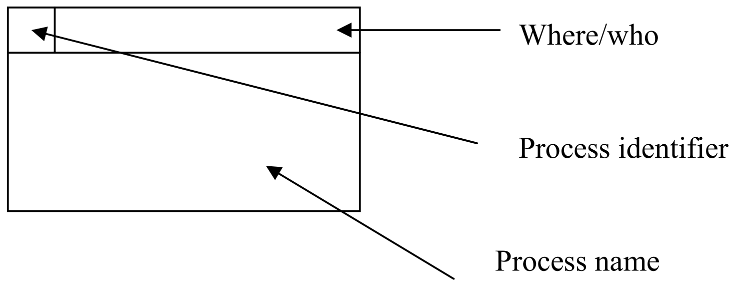
External entities are those things that are identified as needing to interact with the system under consideration. The external entities either input information to the system, output information from the system or both. Typically they may represent job titles or other systems that interact with the system to be built. Some examples are given below. Notice that the SSADM symbol is an ellipse. If the same external entity is shown more than once on a diagram (for clarity) a diagonal line indicates this.



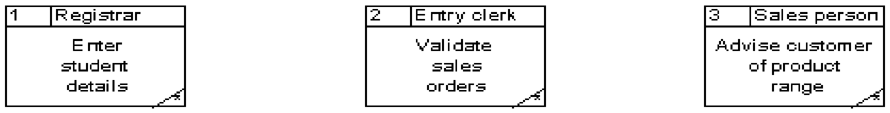
Processes
Processes are actions that are carried out with the data that flows around the system. A process accepts input data needed for the process to be carried out and produces data that it passes on to another part of the DFD. The processes that are identified on a design DFD will be provided in the final artifact. They may be provided for using special screens for input and output or by the provision of specific buttons or menu items. Each identifiable process must have a well-chosen process name that describes what the process will do with the information it uses and the output it will produce. Process names must be well chosen to give a precise meaning to the action to be taken. It is good practice to always start with a strong verb and to follow with not more than four or five words. Examples of good process names would be:

- Enter customer details
- Register new students
- Validate sales orders.

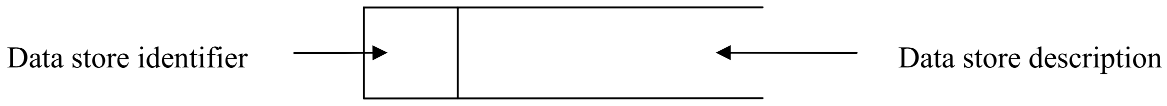
The process symbol has three parts:

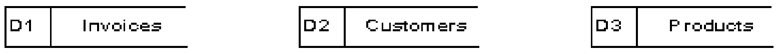


The process identifier is allocated so that each process may be referred to uniquely. The sequence of the process identifiers is usually unimportant but they are frequently to be seen as 1., 2., 3., etc. The top right hand section of the box is used to describe where the process takes place or who is doing the process.



Data Stores
Data stores are places where data may be stored. This information may be stored either temporarily or permanently by the user. In any system you will probably need to make some assumptions about which relevant data stores to include. How many data stores you place on a DFD somewhat depends on the case study and how far you go in being specific about the information stored in them. It is important to remember that unless we store information coming into our system it will be lost. The symbol for a data store are:.





As data stores represent a person, place or thing they are named with a noun. Each data store is given a unique identifier D1, D2 D3 etc.

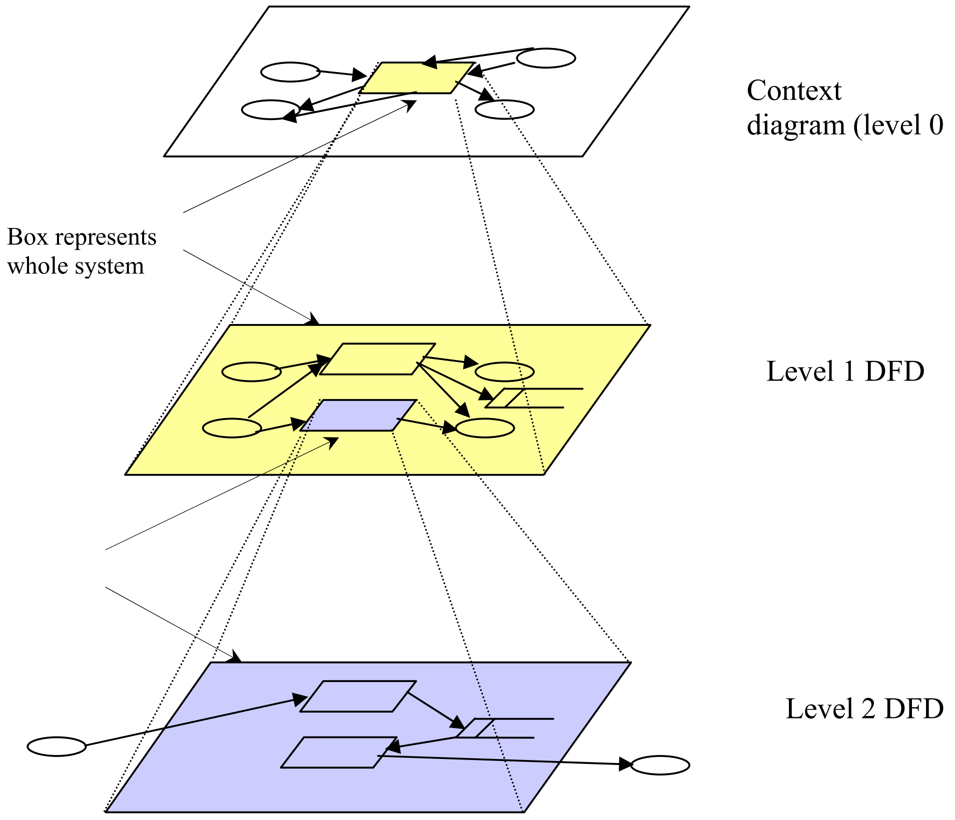
Data flows
The previous three symbols may be interconnected with data flows. These represent the flow of data to or from a process. The symbol is an arrow and next to it a brief description of the data that is represented. There are some interconnections, though, that is not allowed. These are:

- Between a data store and another data store
    - This would imply that one data store could independently decide to send some of information to another data store. In practice this must involve a process.
- Between an external entity and a data store
    - This would mean that an external entity could read or write to the data stores having direct access. Again in practice this must involve a process.

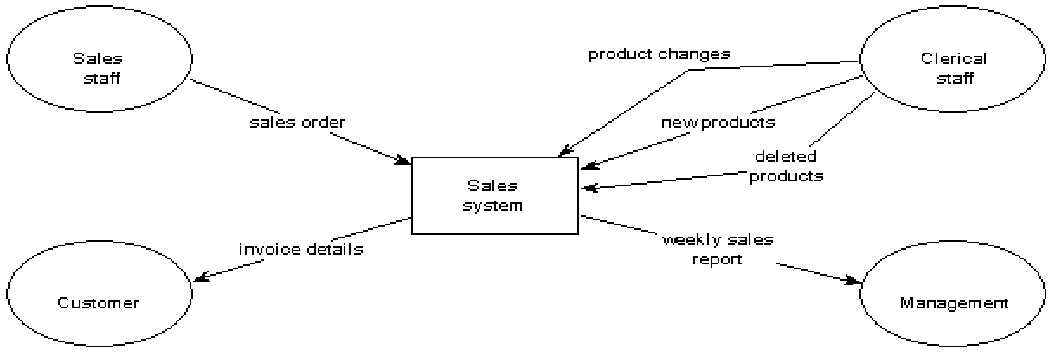
Also, it is unusual to show interconnections between external entities. We are not normally concerned with information exchanges between two external entities as they are outside our system and therefore of less interest to us. Below are some examples of data flows.

Applicant's name
Customer details
Employee record
Payment

Developing Data Flow Diagrams
Data flow diagrams usually occur in sets. The set consist of different levels. To start with a context diagram is drawn. This shows the people and/or systems that interact with the system under construction. By interaction we mean putting information in or taking information out of our system. This diagram gives an overview of the information going in and coming out of the system. The system is represented by a box. In this DFD (level 0 DFD) no processes or data stores are shown. Another diagram (level 1 DFD) is now developed which shows the processes taking place to convert the inputs shown in the context diagram to the outputs. In this DFD detail is given to show which processes are responsible for accepting the different inputs and producing the different outputs. Any process shown that is complicated by a number of data flows and requires further refinement is shown on another diagram (level 2 DFD) where sub-processes are shown together with any necessary extra data stores. The figure below shows how these levels of detail are related to one



Context diagram
This DFD provides an overview of the data entering and leaving the system. It also shows the entities that are providing or receiving that data. These correspond usually to the people that are using the system we will develop. The context diagram helps to define our system boundary to show what is included in, and what is excluded from, our system. The diagram consists of a rectangle representing the system boundary, the external entities interacting with the system and the data which flows into and out of the system.



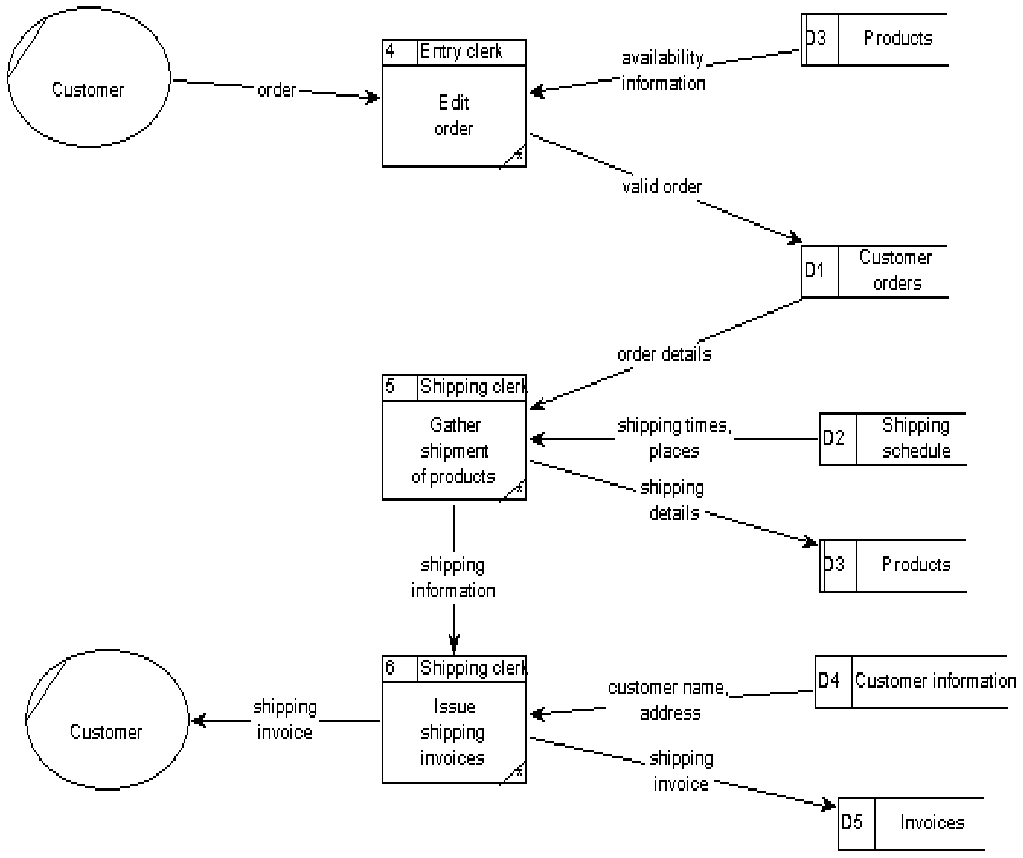
Level 1 Data Flow Diagram

Now we wish to develop further a model of what the system will do with the information the external entities will supply to it. We construct a diagram which is, in effect, taking a magnifying glass to the system boundary represented by the rectangle in the context diagram. We will be looking inside the rectangle and describing what is done with the data inputs in order to provide the data outputs required. The level 1 DFD we construct is a 'child diagram' of the context diagram. We should see all the inputs outputs and external entities that are present in the context diagram. This time we shall include processes and data stores. When students come to construct these level 1 DFDs for the first time they are often unsure as to how many processes to show. It somewhat depends on the case study. However, as a guide it is probable that you would not want more than eight or nine processes as more than this would make the diagram too cluttered. In addition the process names should be chosen so that there are at least three. Fewer than this would mean that the system was unrealistically simple.

Note the following features:

- Every data flow on the context diagram, to or from an external entity, is also shown on the Level 1 DFD.
- Each process has a good strong verb describing what the process is doing with the information received.
- Some data stores appear more than once. This is indicated by a double vertical line on the left hand side of the symbol.
- Each process has access to the relevant information to be able to produce the required output
- Each process has at least one data flow input and one output.
- Information input to the system is always stored somewhere and so never lost
- At this stage the invoice data store has an input data flow but no output. This would indicate that the invoice data is never used. Later on in the development of this system we might define a process for checking the invoices have been paid and so this information in the invoice data store would then be used.

One of the processes (process 8) has no asterisk in the bottom right hand corner although all the others do. This is explained in the next section.



Steps to draw a Data Flow Diagram:

1. Start from the context diagram. Identify the parent process and the external entities with their net inputs and outputs
2. Place the external entities on the diagram. Draw the boundary.
3. Identify the data flows needed to generate the net inputs and outputs to the external entities.
4. Identify the business processes to perform the work needed to generate the input and output data flows.
5. Connect the data flows from the external entities to the processes. Identify the data stores.
6. Connect the processes and data stores with data flows.
7. Continue to decompose to the nth level DFD. Draw all DFDs at one level before moving to the next level of decomposing detail. You should decompose horizontally first to a sufficient nth level to ensure that the processes are partitioned correctly; then you can begin to decompose vertically

Exercise:

1. Draw context diagram and Level 1 diagram of the following case study

Case Study – Pizza Supreme

A large pizza business makes pizzas and sells them. The pizzas are manufactured and kept in cold storage for not more than two weeks.

The business is split into a number of functional units. There is Production Control, Manufacturing, Stores, Accounts, Sales, Shipping and Purchasing. Production Control are responsible for organizing which pizzas to produce in what order and in what quantity. They need to schedule the production of the pizzas according to the current and expected sales orders together with the number of pizzas already in Stores. Manufacturing take the raw materials from the Stores and manufacture pizzas returning the completed goods to the Stores. Accounts deal with the payments for the pizzas when delivered to the customer and the payment to the suppliers of the raw materials. Sales deal with customer orders whilst Purchasing organize the buying of raw material from suppliers. Shipping manage the packing and delivery of the goods to the customer with a delivery note.

When a sales order is received by sales they record what is being ordered and by whom. They also record the details of the expected date of delivery. Production Control access this information and make sure that, if required, pizzas are produced by Manufacturing and are ready in Stores for when the delivery needs to be made.

After the delivery is made Accounts make sure that the customer receives an invoice and that payment for the invoice is received at which time a receipt is issued. Purchasing look at the current stock of raw materials and by using current stock levels, supplier turnaround times and quantity to be ordered decide what needs to be ordered on a daily basis. Their aim is never to run out of an ingredient but to minimize the amount of raw material kept in stock.

.

Lab # 7

Object:
User Interface Prototyping using GUI Design Studio.

Theory:
User interface (UI) prototyping is an iterative analysis technique in which users are actively involved in the mocking-up of the UI for a system. UI prototypes have several purposes:

- As an analysis artifact that enables you to explore the problem space with your stakeholders.
- As a requirements artifact to initially envision the system.
- As a design artifact that enables you to explore the solution space of your system.
- A vehicle for you to communicate the possible UI design(s) of your system.
- A potential foundation from which to continue developing the system (if you intend to throw the prototype away and start over from scratch then you don't need to invest the time writing quality code for your prototype).

GUI Design Studio
GUI Design Studio is a code-free, drag and drop user interface design and prototyping tool for creators of Web, Desktop, Mobile and Embedded software applications.

- Design screens or web pages
- Add user interaction behavior.
- Test as a running prototype

Software Designers, User Experience Professionals, Business Analysts, Developers, Project Managers and Consultants can use it to.

- Document product ideas
- Create project proposals
- Create design mock-ups

We can use it to quickly lay down ideas and test them out in a low-cost, risk-free environment and avoid expensive implementation rework by getting the right design agreed upon first.

- Verify designs and requirements
- Explore alternatives
- Evaluate different usage scenarios

The Project panel on the Design Bar provides access to all of your project design documents and image files. From here you can quickly create new project folders to organize your files, create new design documents, duplicate existing designs and import images from the clipboard.

The Elements panel on the Design Bar provides access to all of the windows and controls that can be used to create an application GUI.Elements are organized into categories. When you select a category from the list, its elements appear in the palette window below.

The Icons panel on the Design Bar provides access to common icon images and those within the main project and any others that have been linked in from the Project panel.

The common icons are organized into categories. You can create additional category folders for new icons.

The Annotations panel on the Design Bar provides access to special elements such as overlay text boxes, highlight rings and markers. The annotation elements always appear on top of other design elements.

The Storyboard panel on the Design Bar provides access to special elements for building flow control in a design to create a working prototype. The storyboard elements always appear on top of other design elements.

The Data panel on the Design Bar provides access to the Data Tables within your project.

The Notes panel on the Design Bar provides editors for recording notes associated with a design document and its elements. These can be used to provide popup descriptions and to generate specification documents. Unlike all of the other panels, the content of the Notes panel changes to reflect the active design document.

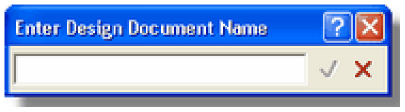
By default, the working area background for designs is plain white. You can change this in the Preference settings.

With GUI Design Studio, all of your work will normally be done within **projects.** A project has its own folder structure on disk and may contain design documents, bitmap image files and project-specific icons. The project file is stored within the project folder and has the .GDP extension. Sometimes you will want to access the files within one project from another project, such as:

- Using a library of common design components.
- Referencing a set of GUI design guidelines.
- Building on an older version of a project within a new development project
- Using another project as a reference example.

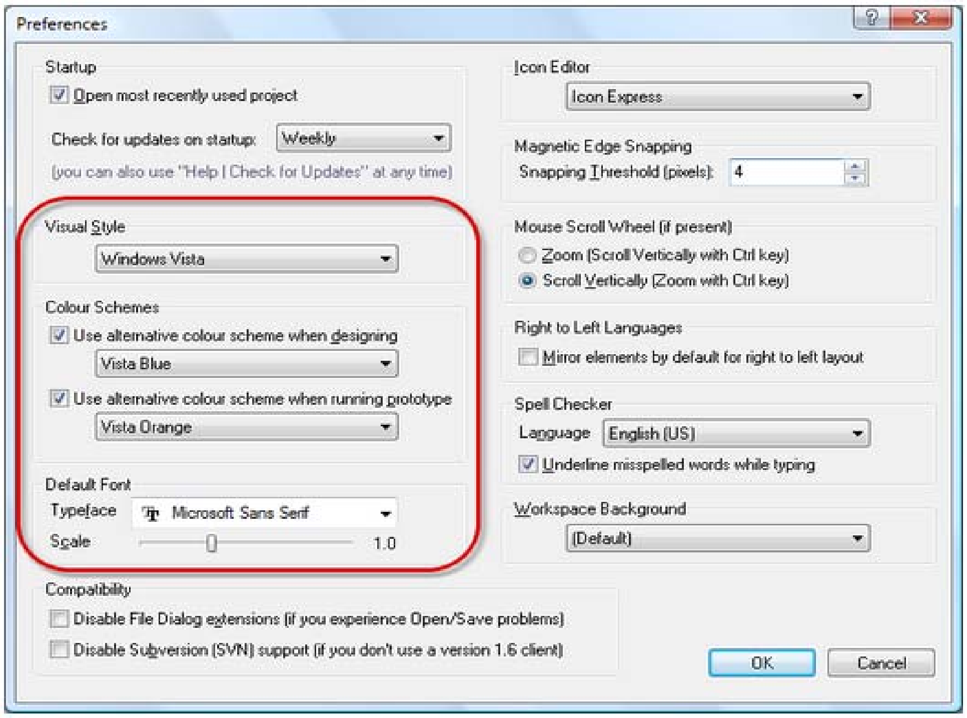
Creating a new project
Menu Command:	File then New Project
When you create a new project you must enter a name and select a root folder for the project files.

Creating a new project
Menu Command:	File then New Project then New Design
There are two methods for creating design documents. The first uses the File | New menu command or "New" toolbar button to create and open a blank, untitled document ready for editing. You will then be prompted for just the document name without having to browse any further. The file will then be created and opened ready for editing.



Visual Styling

Using the "File | Preferences..." menu command, GUI Design Studio allows you to choose a default visual style, color scheme and font. All color schemes work with all visual styles and a different color scheme may be selected for design and presentation


Preferences

Startup
[✓] Open most recently used project
Check for updates on startup: Weekly
(you can also use "Help | Check for Updates" at any time)

Visual Style
Windows Vista

Colour Schemes
[✓] Use alternative colour scheme when designing
Vista Blue
[✓] Use alternative colour scheme when running prototype
Vista Orange

Default Font
Typeface: Microsoft Sans Serif
Scale: 1.0

Compatibility
[ ] Disable File Dialog extensions (if you experience Open/Save problems)
[ ] Disable Subversion (SVN) support (if you don't use a version 1.6 client)

Icon Editor
Icon Express

Magnetic Edge Snapping
Snapping Threshold (pixels): 4

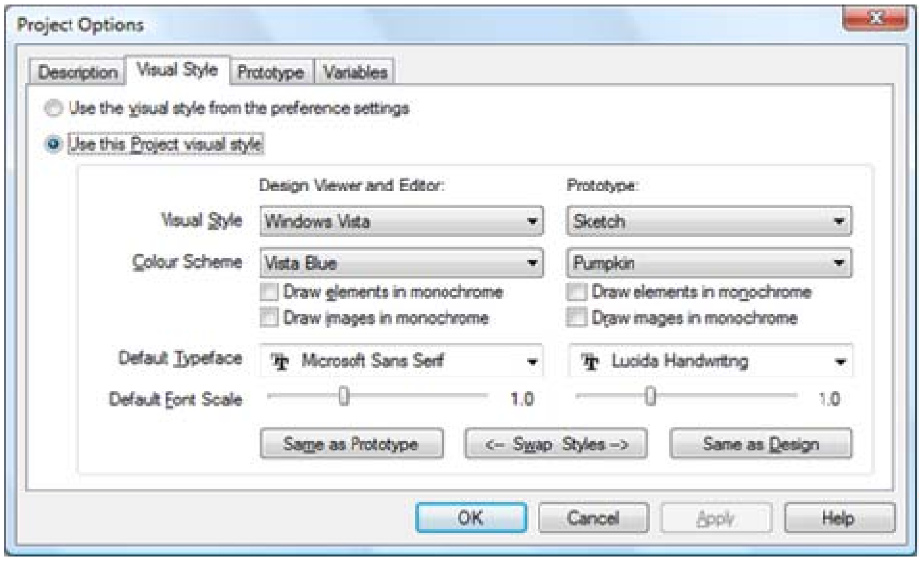
Mouse Scroll Wheel (if present)
○ Zoom (Scroll Vertically with Ctrl key)
● Scroll Vertically (Zoom with Ctrl key)

Right to Left Languages
[ ] Mirror elements by default for right to left layout

Spell Checker
Language: English (US)
[✓] Underline misspelled words while typing

Workspace Background
(Default)

You can also set the visual style for a project, to override the default, from the Visual Style tab of the Project Options dialog ("Project | Project Options..." menu command). In that case, you can select a different visual style as well as a different color scheme for editing and presentation:


Project Options

○ Use the visual style from the preference settings
● Use this Project visual style

	Design Viewer and Editor:	Prototype:
Visual Style	Windows Vista	Sketch
Colour Scheme	Vista Blue	Pumpkin
	[ ] Draw elements in monochrome	[ ] Draw elements in monochrome
	[ ] Draw images in monochrome	[ ] Draw images in monochrome
Default Typeface	Microsoft Sans Serif	Lucida Handwriting
Default Font Scale	1.0	1.0

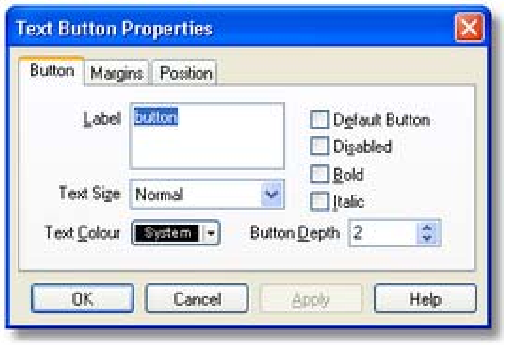
Adding Elements
Design elements include all of the windows, dialogs, controls, menus, toolbars and other widgets that appear in an application. These can be found on the elements panel on the design bar. To add an element to a design document, simply double click it or drag it to the desired location

Selecting an element
To select an element for manipulation, click on it using the left mouse button. If you hold the Ctrl key down then this will toggle the selection of the element without affecting any other selected elements allowing for multiple selections.

You can also select multiple elements by dragging a selection box using the left mouse button. All elements fully enclosed by the box will be selected. Again, the Ctrl key can be used to toggle selections and add to or remove elements from the selection set

Selected elements have a thin border with square handles for resizing. When multiple elements are selected, one of them is known as the **Primary Selected Element** and displays with normal, solid handles. All others are called **Secondary Selected Elements** and display with hollow handles.

Menu Command: Edit | Properties Alt + Enter
Each type of element has its own set of properties. You can change the properties of only one element at a time. Select the element with the mouse then use the Alt+Enter shortcut key, or simply double click the element, to open its property editor dialog.



Label: button
Text Size: Normal
Text Colour: System
Button Depth: 2

Any changes you make in the dialog are immediately applied to the element so that you get an instant preview, except in the case of edit boxes where you must use the "Apply" button to see the change.

Exercise:

1. Make an interface for a simple calculator.

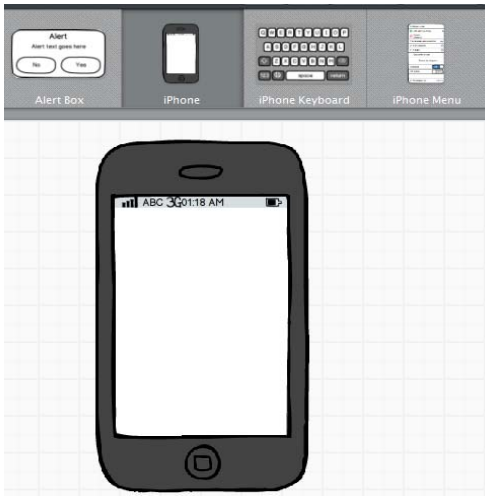
2. Make an interface for a login window.

Lab # 8

Object:
Interface prototyping using BALSAMIQ.

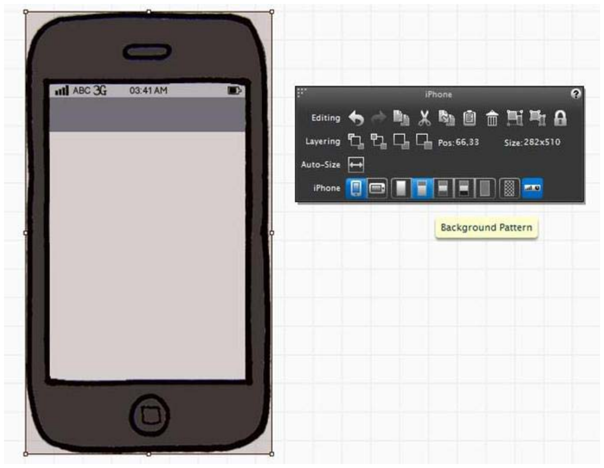
Theory:
BALSAMIQ is an interface mockup tool that will help to quickly prototype your ideas and communicate them to others.

BALSAMIQ UI Library
We will be using BALSAMIQ to create a simple prototype for a mobile application. Click on the File menu and select New Blank Mockup. Click the iPhone tab of the UI Library, select the "iPhone" element, and drag it onto the grid below.



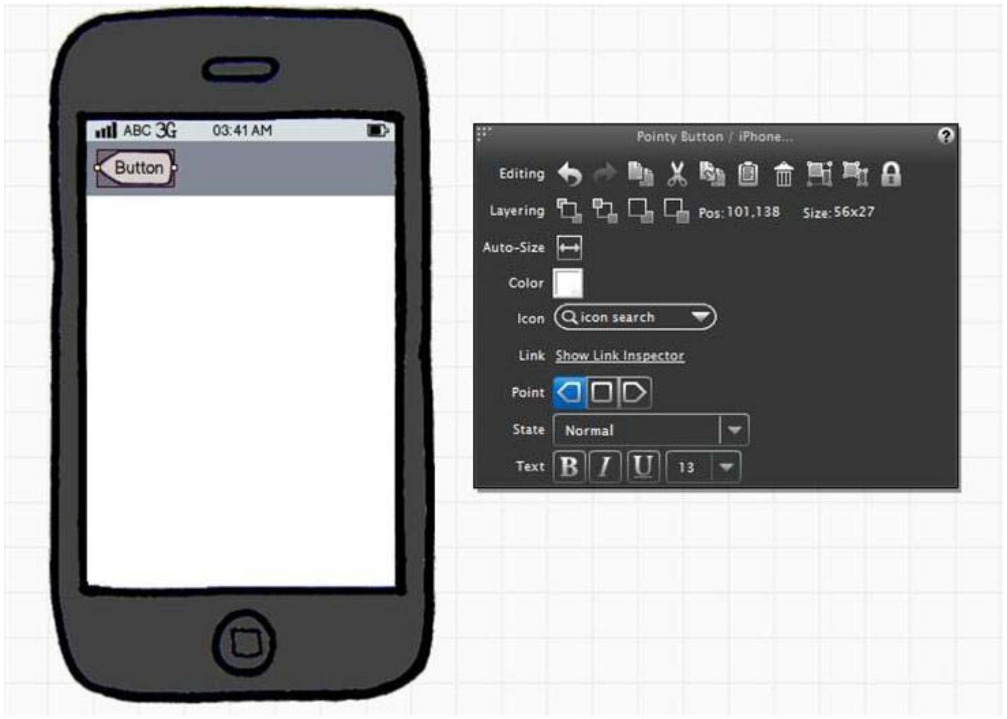
When you click on the iPhone, an options box appears. Each control in the UI Library has a set of standard options (layering, grouping, and clipboard operations) and additional options unique to that particular control.

With the iPhone control, we can change the orientation of the phone, select a different background pattern, or toggle the top bar. For now, let's select the second background pattern, in order to give the screen a colored bar at top.



Next, we'll add some typical iOS interface components. Select the "Pointy Button" control from the iPhone tab of the UI Library, and drag it onto the top bar of the iPhone.

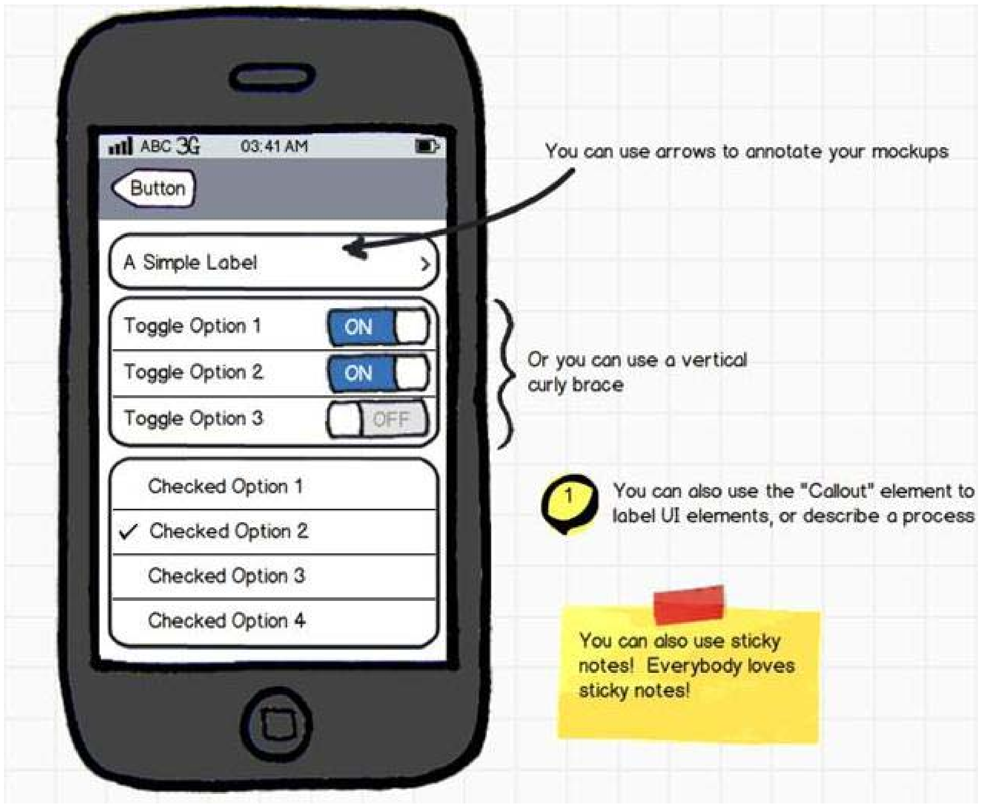
After placing the control, you will have the option to edit it's text. You can change this to anything you'd like, but for this tutorial, we'll keep it as "Button". In the options box for the Pointy Button, you can select the direction of the button, and select an icon to be displayed on it. Here, we've created a left-pointing button with no icon.



Next, we'll add an "iPhone Menu" Control from the UI Library. The iPhone Menu has many options, most of which can be activated by typing certain characters into the text field.

Once you've placed the iPhone Menu, take a look at the syntax given in the menu by default to see how to reproduce a number of different menu effects. (Don't forget to put a comma between the label for the menu item, and the additional element!) For example, you can type "Random Text, ON" to create a menu item with the text "Random Text" and a toggle switch set to *on.* Below,

I've added three iPhone menus, each with multiple items, to create a hypothetical "Settings" page for an iPhone application. I've also used some elements from the "Markup" tab of the UI Library to annotate my mockup.

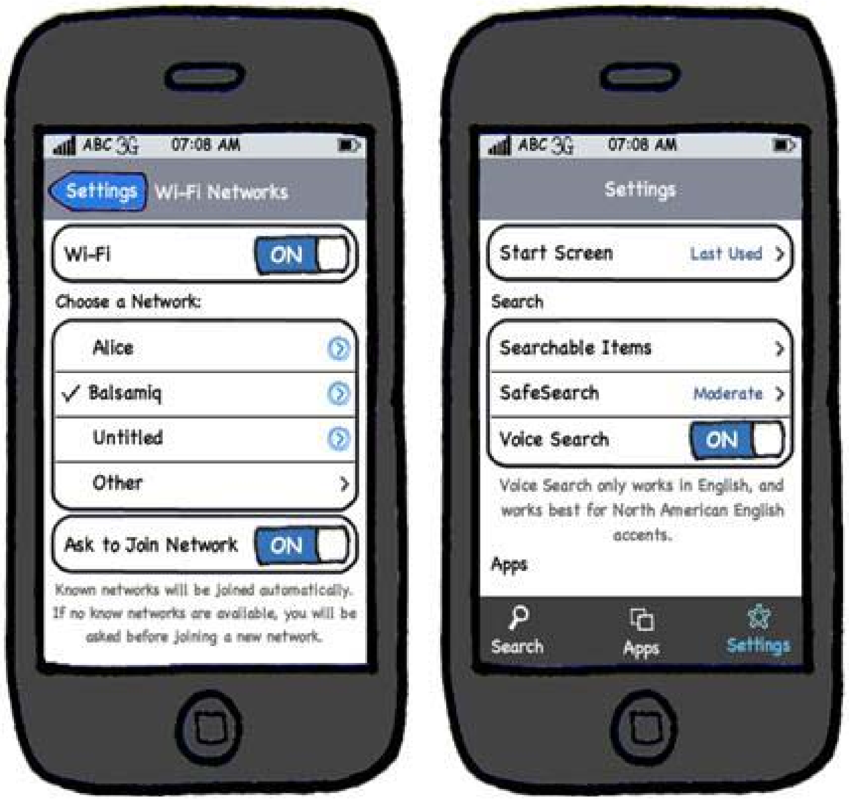

ABC 3G 03:41 AM
Button
A Simple Label
Toggle Option 1 ON
Toggle Option 2 ON
Toggle Option 3 OFF
Checked Option 1
Checked Option 2
Checked Option 3
Checked Option 4
You can use arrows to annotate your mockups
Or you can use a vertical curly brace
1
You can also use the "Callout" element to label UI elements, or describe a process
You can also use sticky notes! Everybody loves sticky notes!

Once you're finished, go to the "Mockup" menu in the top left and download your mockup as either a PDF or PNG file, and save it wherever you'd like.

BALSAMIQ offers *many* common UI controls, as well as elements which you can use to annotate your mockups. And BALSAMIQ isn't just for prototyping mobile applications– it also has the ability to produce mockups for websites, web applications, and more. Explore the UI Library to see what else is available.

Exercise:

1. Design the following iPhone Prototypes.



ABC 3G 07:08 AM

Settings Wi-Fi Networks

Wi-Fi ON

Choose a Network:

Alice

✓ Balsamiq

Untitled

Other

Ask to Join Network ON

Known networks will be joined automatically. If no know networks are available, you will be asked before joining a new network.

ABC 3G 07:08 AM

Settings

Start Screen Last Used

Search

Searchable Items

SafeSearch Moderate

Voice Search ON

Voice Search only works in English, and works best for North American English accents.

Apps

Search Apps Settings