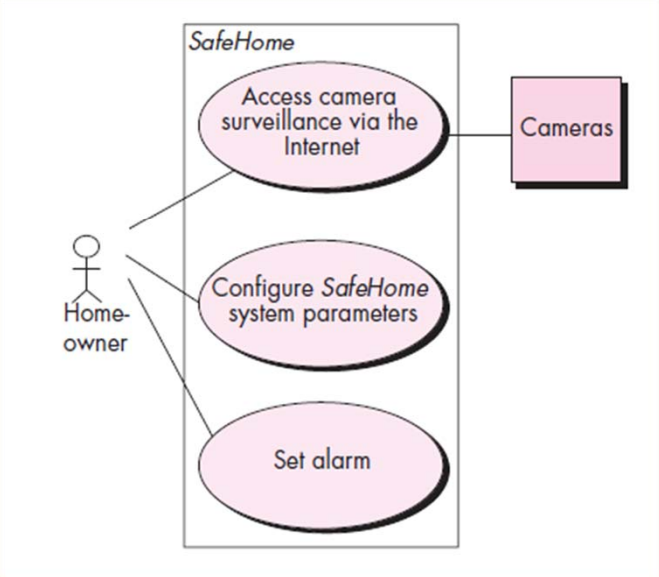
Requirements Modeling:
Scenarios, information, and analysis classes
(II)

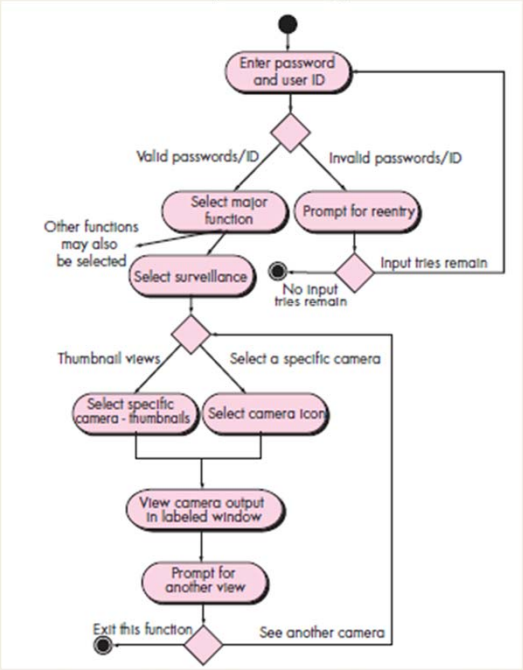
Use-Cases

- a scenario that describes a “thread of usage” of a software system
- *actors* represent roles people or devices play as the system functions
- *users* can play a number of different roles for a given scenario

Use-Case Diagram

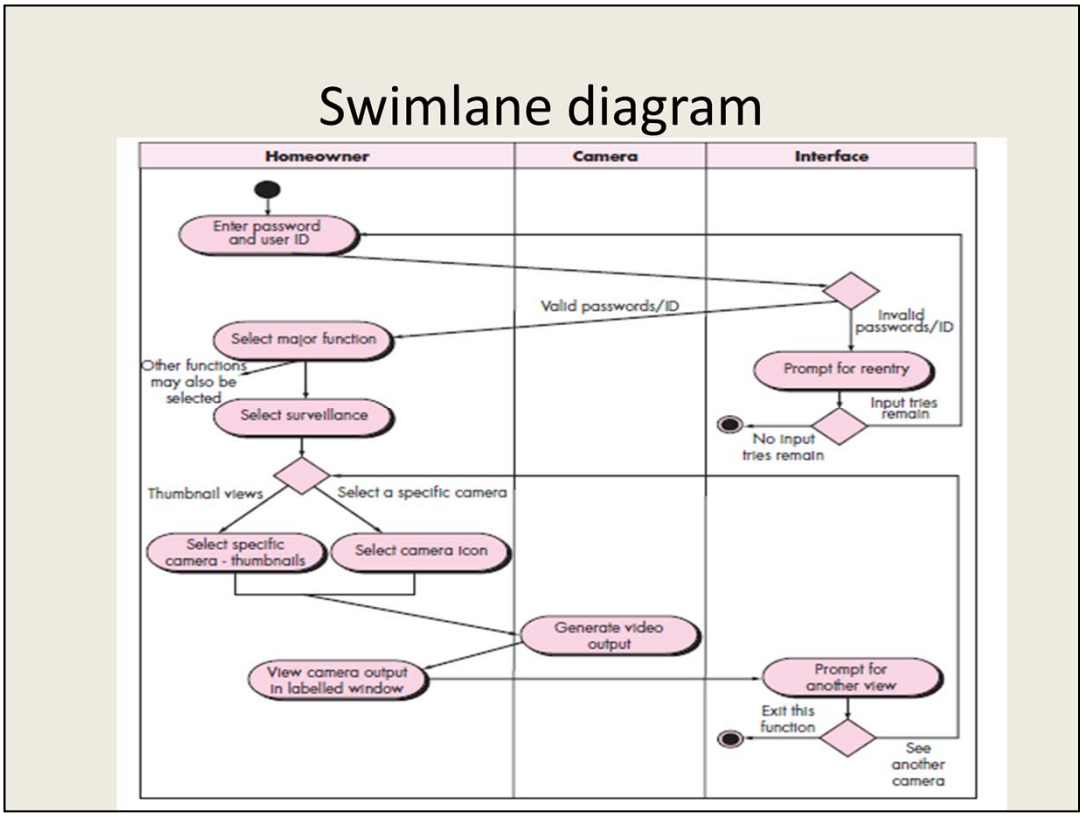


Activity Diagram



Supplements the use-case by providing a diagrammatic representation of procedural flow

There are many requirements modeling situations in which a text-based model—even one as simple as a use case—may not impart information in a clear and concise manner.



Extension of activity diagram.

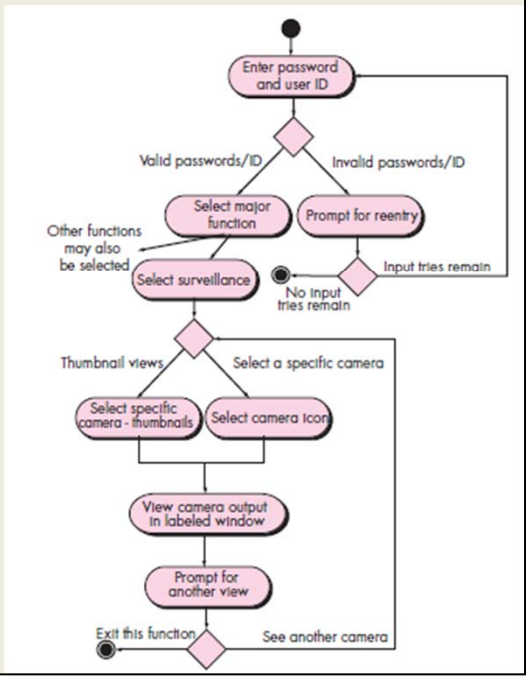
Swimlane diagram Allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are multiple actors involved in a specific use-case) or analysis class has responsibility for the action described by an activity rectangle

Use-case representation

- How are these diagrams related?
  - Use-case diagram
  - Activity diagram
  - Swimlane diagram
- When will you need the latter two?
  - Why?

Activity Diagram

- An activity diagram shows the flow of events within the system.
- Incorporates procedural and parallel behavior.



Is a special case of a state diagram

In which all (or most) of the states are action or subactivity states

And in which all (or most) of the transitions are triggered by completion of the actions or subactivities in the source states.

Describe activities which involve concurrency and synchronization

Activity Diagrams

- Can be used
  - To model business workflows
  - To describe a system function that is represented within a use case or between use cases.
  - In operation specifications, to describe the logic of an operation.
- MSDN reference
  - http://msdn.microsoft.com/en-us/library/vstudio/dd409360.aspx

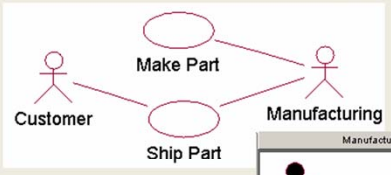
A UML *activity diagram* depicts the dynamic behavior of a system or part of a system

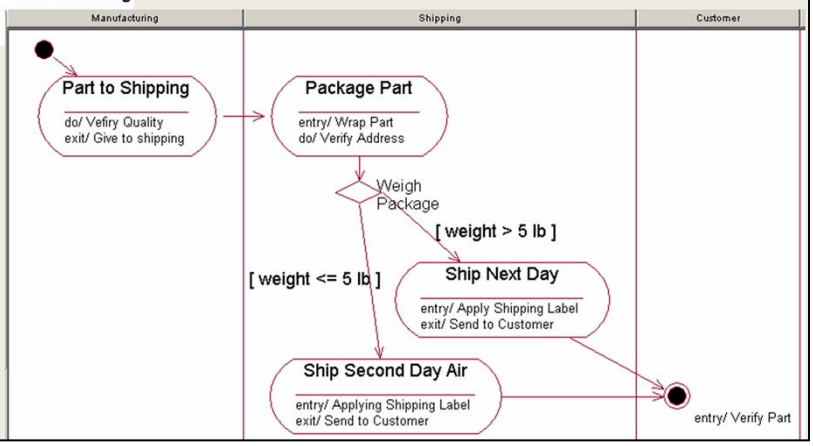
through the flow of control between actions that the system performs. It is similar to

a flowchart except that an activity diagram can show concurrent flows.

Example 1

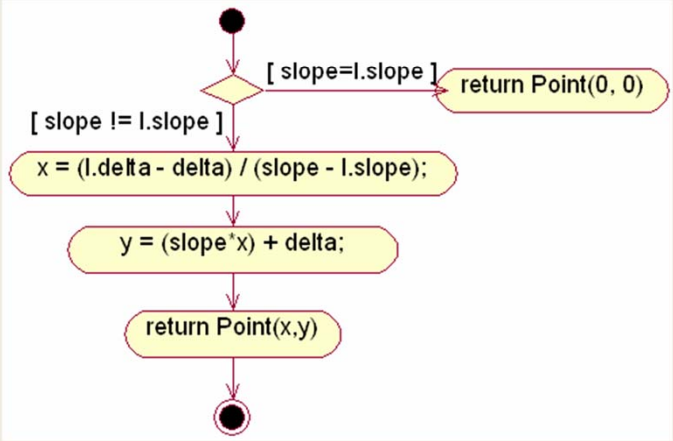
- Build an activity diagram to model business workflow





Example 2

- Build an activity diagram to model logic of operation
  - Intersection operation of the class “Line”

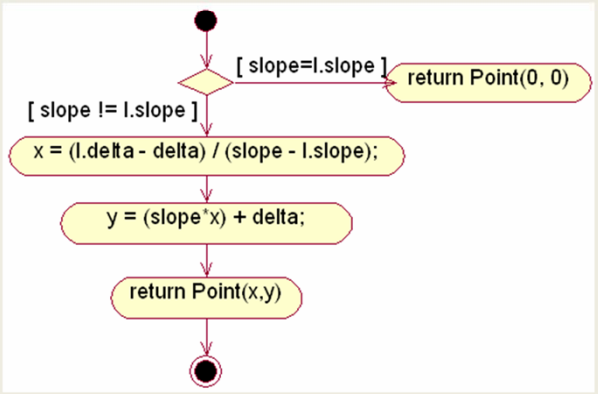


Activity Diagram Elements

- Action
- Branch
- Partition
- Fork and join
- Object flow

Action – action node

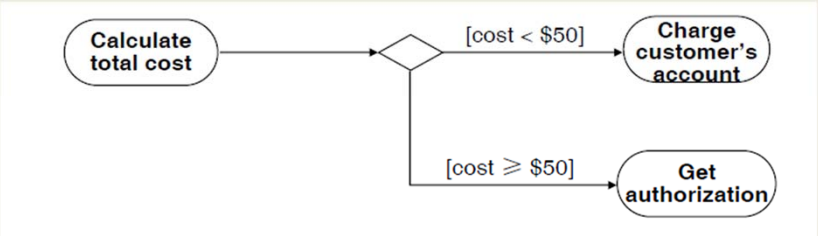
- The main component of an activity diagram is an *action* node, represented by a rounded rectangle, which corresponds to a task performed by the software system.
- Special action nodes
  - Initial node
  - Final node



A solid black dot forms the *initial node* that indicates the starting point of the activity. A black dot surrounded by a black circle is the *final node* indicating the end of the activity.

Branch – decision node

- A *decision* node corresponds to a branch in the flow of control based on a condition.



Dead end: there may be transitions in an activity diagram with no destination state; this can mean that:

- Not all processing has been specified
- Or, that another activity diagram will take over

an incoming arrow and two or more outgoing arrows. Each outgoing arrow is labeled with a guard (a condition

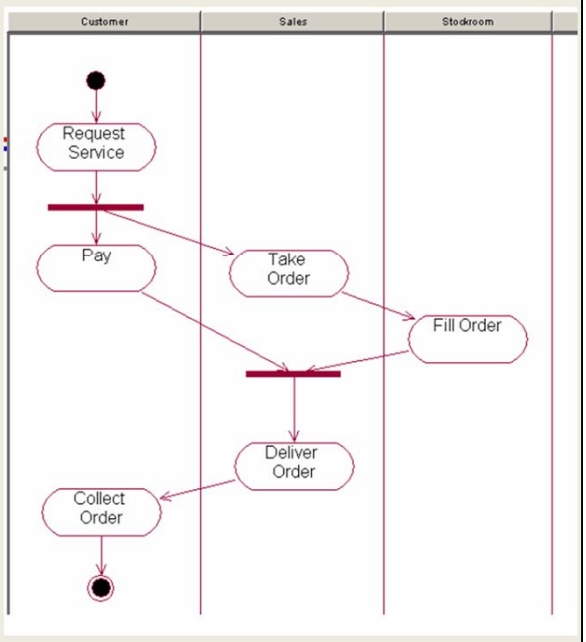
inside square brackets). The flow of control follows the outgoing arrow whose guard

is true. It is advisable to make sure that the conditions cover all possibilities so that

exactly one of them is true every time a decision node is reached.

Partition - Swimlane

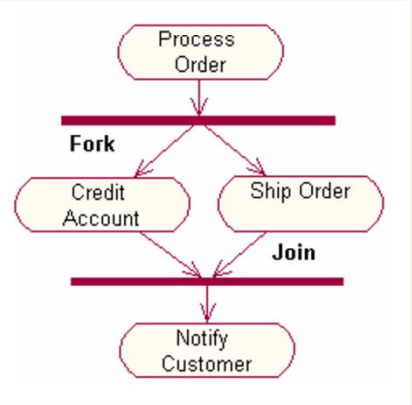
- Swimlanes form a partition of activity diagram for dissecting activities by corresponding actors.
- All actions in one lane are done by the corresponding actor.
- The order of the swimlanes does not matter.

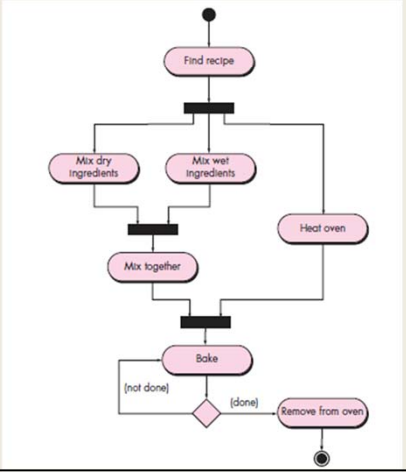


indicate how the actions are divided among the participants, you can decorate the activity diagram with swimlanes

Fork and Join – bar/arrowed lines

- A *fork* represents the separation of activities into two or more concurrent activities.
- A *join* is a way of synchronizing concurrent flows of control.



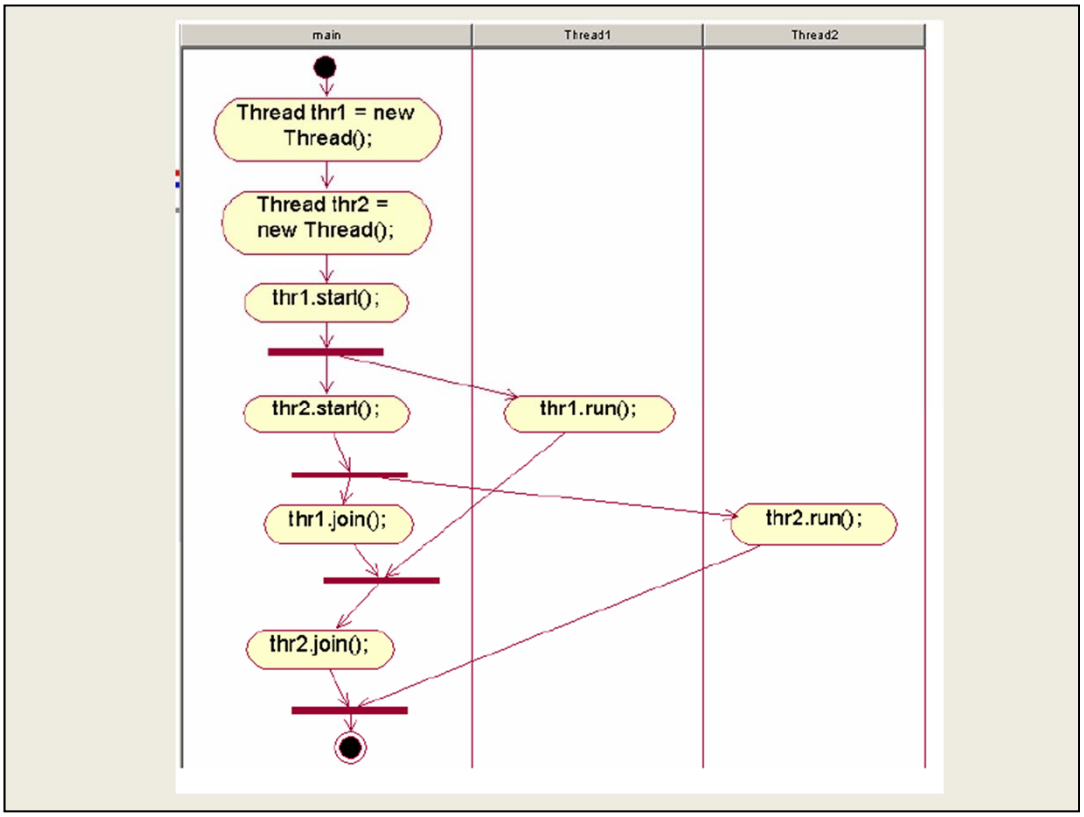


Fork: each outgoing arrow represents a flow of control that can be executed concurrently with the flows corresponding to the other outgoing

Arrows. These concurrent activities can be performed on a computer using different

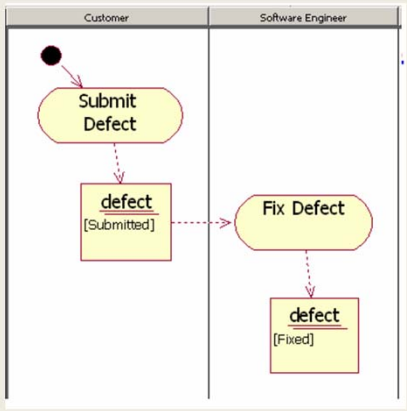
threads or even using different computers.

Join: the flow of control represented by the outgoing arrow cannot begin execution until all flows represented by incoming arrows have been completed


main
Thread1
Thread2
Thread thr1 = new Thread();
Thread thr2 = new Thread();
thr1.start();
thr2.start();
thr1.run();
thr1.join();
thr2.run();
thr2.join();

Object flow – dotted arrowed line

- An object flow on an activity diagram represents the relationship between an activity and the object that creates it (as an output) or uses it (as an input).



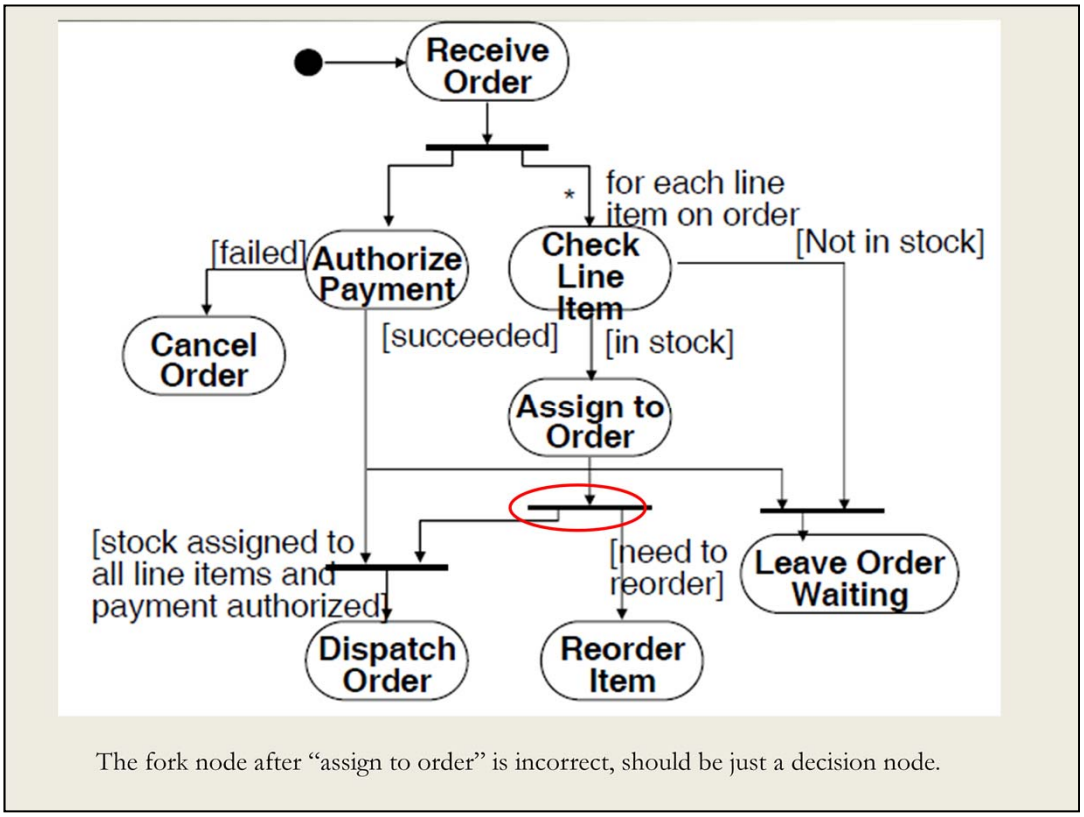
Example: Order Processing

When we receive an order, we check each line item on the order to see if we have the goods in stock. If we do, we assign the goods to the order. If this assignment sends the quantity of those goods in stock below the reorder level, we reorder the goods. While we are doing this, we check to see if the payment is OK. If the payment is OK. and we have the goods in stock, we dispatch the order. If the payment is OK. but we do not have the goods, we leave the order waiting. If the payment is not OK., we cancel the order.

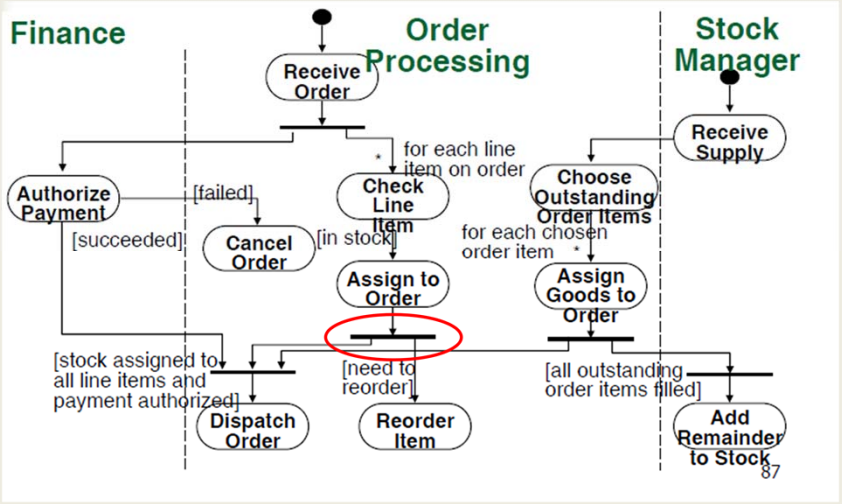
Example: Order Processing

When we receive an order, we check each line item on the order to see if we have the goods in stock. If we do, we assign the goods to the order. If this assignment sends the quantity of those goods in stock below the reorder level, we reorder the goods. While we are doing this, we check to see if the payment is OK. If the payment is OK. and we have the goods in stock, we dispatch the order. If the payment is OK. but we do not have the goods, we leave the order waiting. If the payment is not OK., we cancel the order.

Narrative form of requirement description



The fork node after “assign to order” is incorrect, should be just a decision node.



Similar errors as the previous slide.

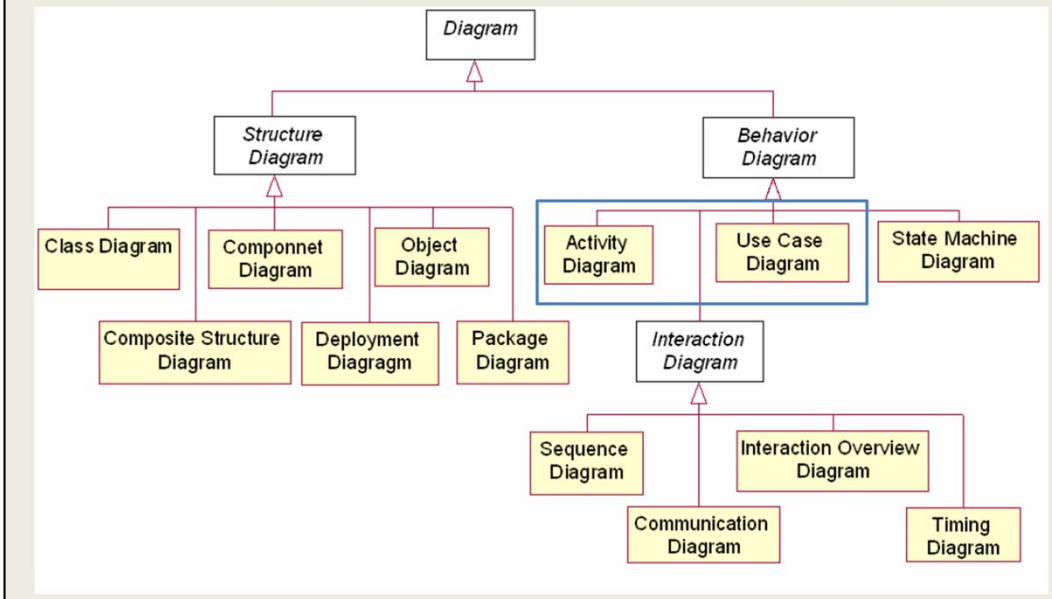


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Componnet Diagram
Object Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Composite Structure Diagram
Deployment Diagragm
Package Diagram
Interaction Diagram
Sequence Diagram
Interaction Overview Diagram
Communication Diagram
Timing Diagram

Summary

- Scenario-based requirement modeling
  - Use case diagram
    - Activity diagram
      - Composition
      - Examples