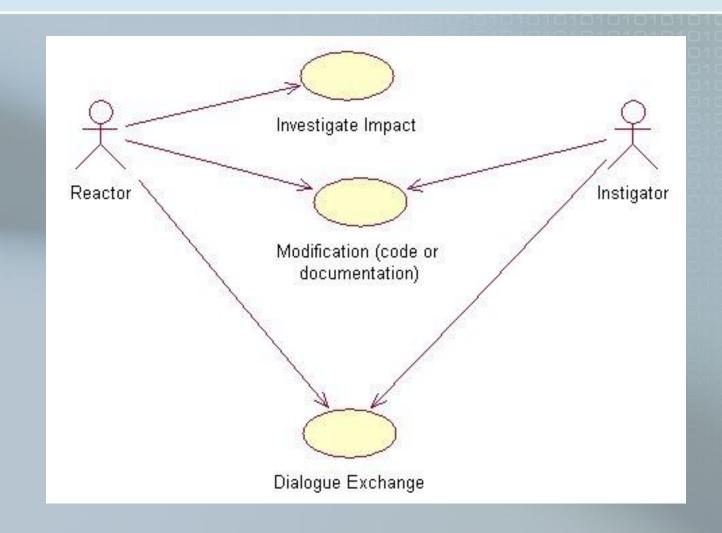
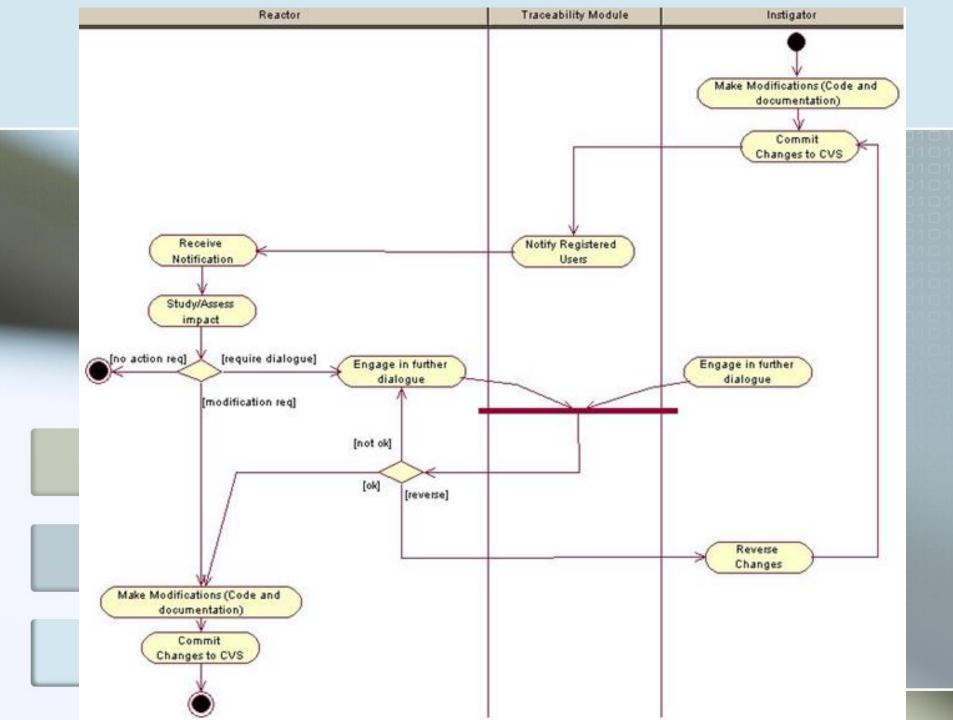


Lecture 09 **ACTIVITY DIAGRAM**

Scenario

- 1. Instigator makes a change to some code
- 2. Instigator then enters a dialogue with Reactor to notify of proposed change.
- **3. Reactor** investigates impact of change
- **4. Instigator** and **Reactor** may have further dialogue to clarify issues
- 5. If **Reactor** agrees to go ahead with change, the associated code and code documentation will be modified. This concerns modifying, testing and finally committing changes back into Code Versioning system (CVS).
- **6. Reactor** has final task to update associated modelling documentation, including text and related UML diagrams (if appropriate)
- 7. If **Reactor** disagrees, then **Instigator** may be forced to undo the changes (if the developers cannot agree amongst themselves, then the problem may be escalated).





What is an Activity?

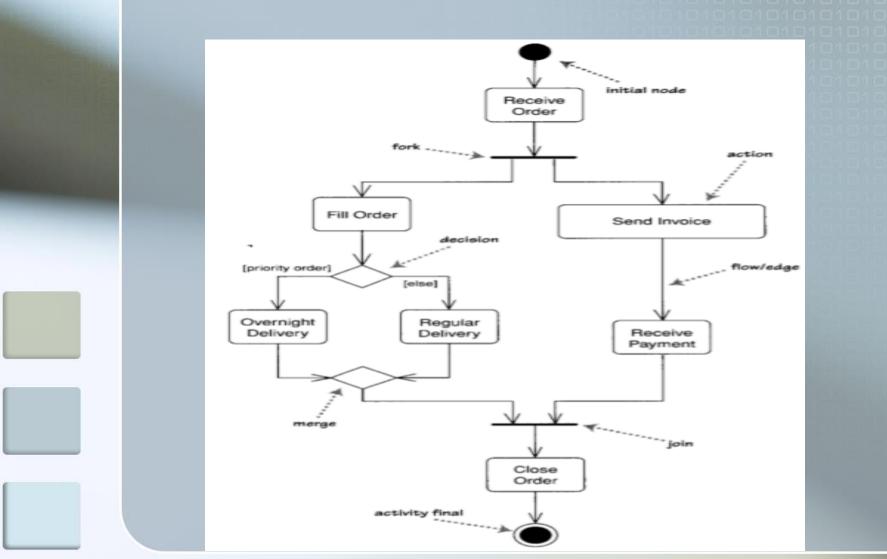
- In a conceptual diagram, an activity is some task that needs to be done, whether by a human or a computer
- In an implementation-perspective diagram, an activity is a method on a class
- Activity Diagrams are used to describe activities
 - Activity Diagrams are useful for describing complicated methods
 - Activity Diagrams are useful for describing use cases, since, after all, a use case is an interaction, which is a form of activity

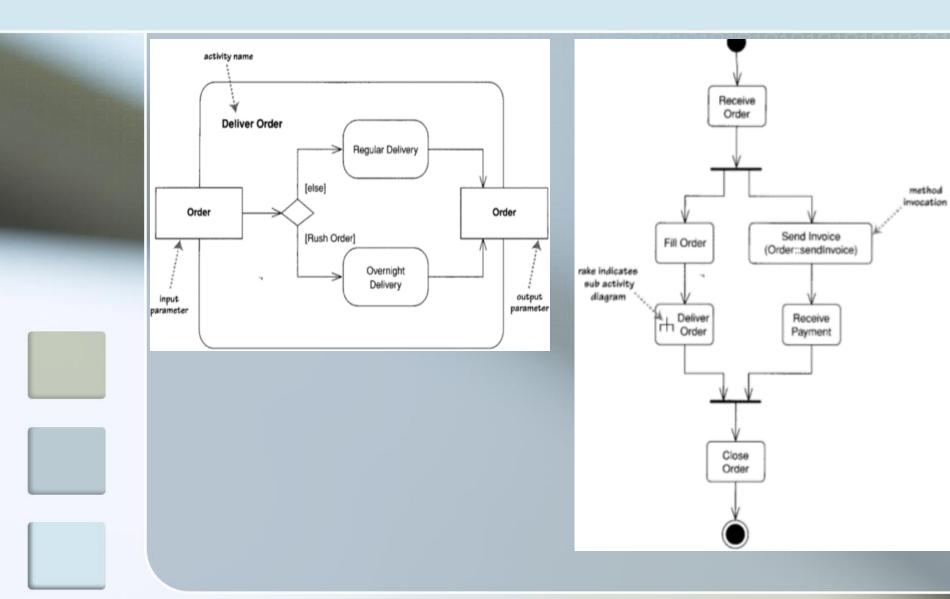
Activity arrangement

- Sequential one activity is followed by another
- Parallel two or more sets of activities are performed concurrently, and order is irrelevant

Processing an Order

Once the order is received the activities split into two parallel sets of activities. One side fills and sends the order while the other handles the invoice and payments. On the Fill Order side, the method of delivery is decided conditionally. Depending on the condition either the Overnight Delivery activity or the Regular Delivery activity is performed. Finally the parallel activities combine to close the order





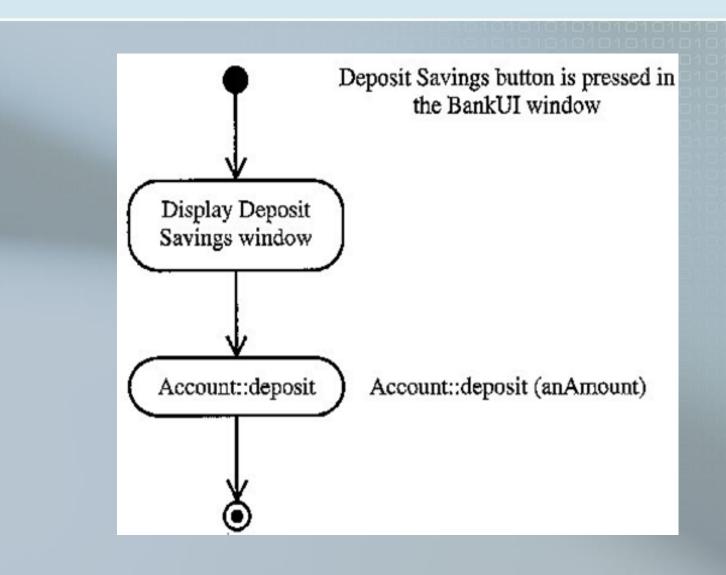
Transition

When the action or activity of a state completes, flow of control passes immediately to the next action or activity state

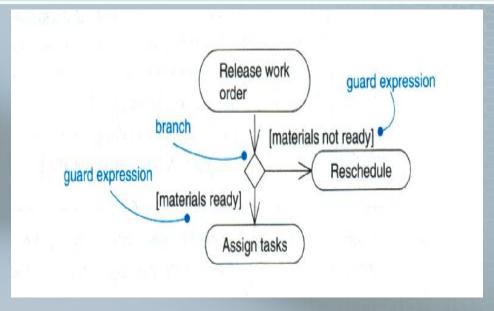
A flow of control has to start and end someplace

- initial state -- a solid ball
- stop state -- a solid ball inside a circle

Transition



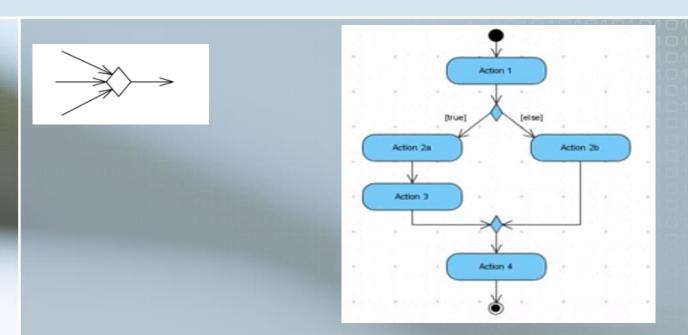
Branching



A branch specifies alternate paths taken based on some Boolean expression

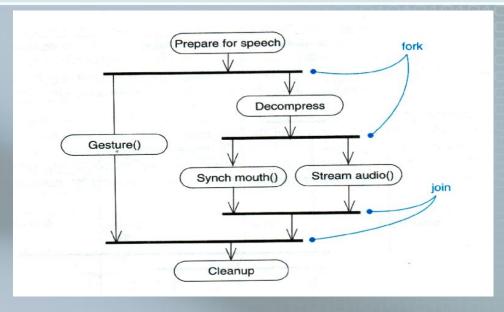
A branch may have one incoming transition and two or more outgoing ones

Merge Node



Merge node is a control node that brings together multiple incoming alternate flows to accept single outgoing flow. There is no joining of tokens. Merge should not be used to synchronize concurrent flows.

Forking and Joining

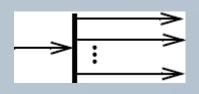


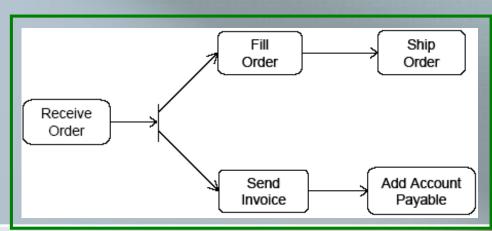
Use a synchronization bar to specify the forking and joining of parallel flows of control

A synchronization bar is rendered as a thick horizontal or vertical line

Fork

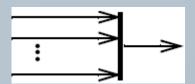
- A fork may have one incoming transitions and two or more outgoing transitions
 - each transition represents an independent flow of control
 - conceptually, the activities of each of outgoing transitions are concurrent
 - either truly concurrent (multiple nodes)
 - or sequential

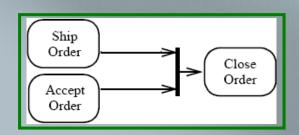




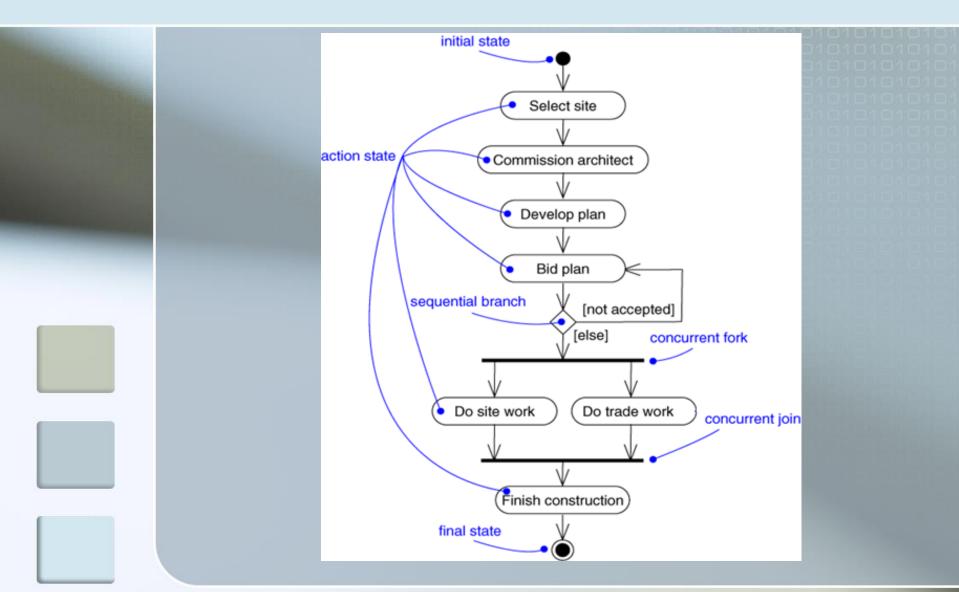
Join

- A join may have two or more incoming transitions and one outgoing transition
 - above the join, the activities associated with each of these paths continues in parallel
 - at the join, the concurrent flows synchronize
 - each waits until all incoming flows have reached the join, at which point one flow of control continues on below the join





Bid for Site



Swimlane

- A swimlane specifies a locus of activities
- To partition the action states on an activity diagram into groups
 - each group representing the business organization responsible for those activities
 - each group is called a swimlane
- Each swimlane is divided from its neighbor by a vertical solid line

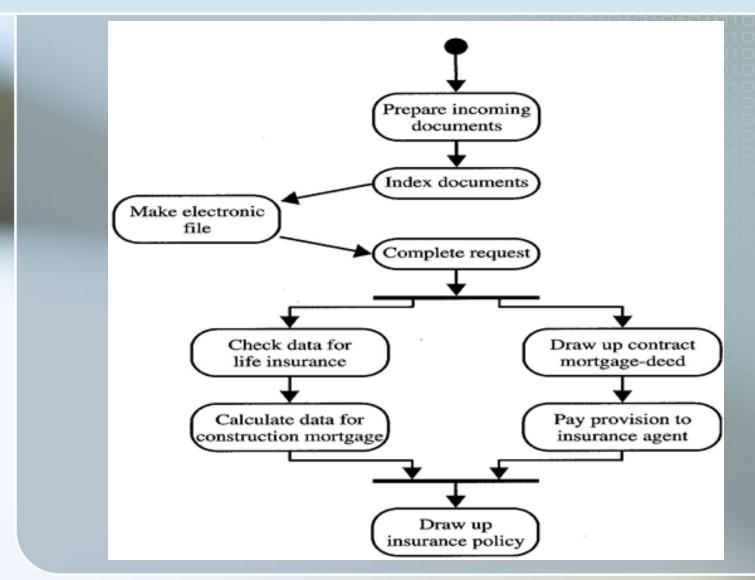
Swimlane

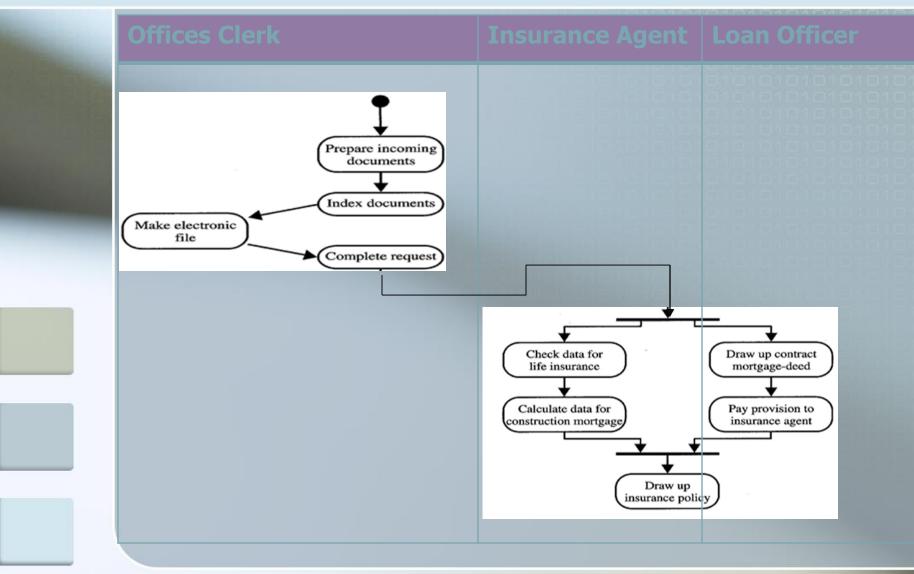
- Each swimlane has a name unique within its diagram
- Each swimlane may represent some real-world entity
- Each swimlane may be implemented by one or more classes
- Every activity belongs to exactly one swimlane, but transitions may cross lanes

Insurance Policy

Office clerk prepare the documents for insurance policy by indexing the document, make a electronic file and complete the request. After the request is completed insurance agent check data for life insurance as well as loan officer make a contract of an advance loan deeds, insurance agent calculate data for advance loan deeds, loan officer pay provision to insurance agent. At the end the insurance agent draw up the policy plan.

Insurance Policy

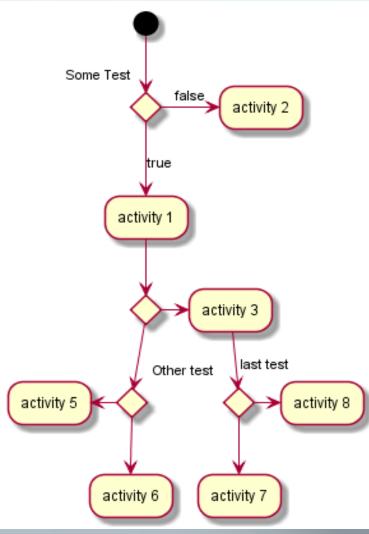




Implementation of nested branching

```
(*) --> if "Some Test" then -->[true] "activity 1"
```

```
if "" then
  -> "activity 3" as a3
 else
  if "Other test" then
   --> "activity 5"
  else
   --> "activity 6"
  endif
 endif
else
 ->[false] "activity 2"
endif
a3 --> if "last test" then
 --> "activity 7"
else
 -> "activity 8"
endif
```

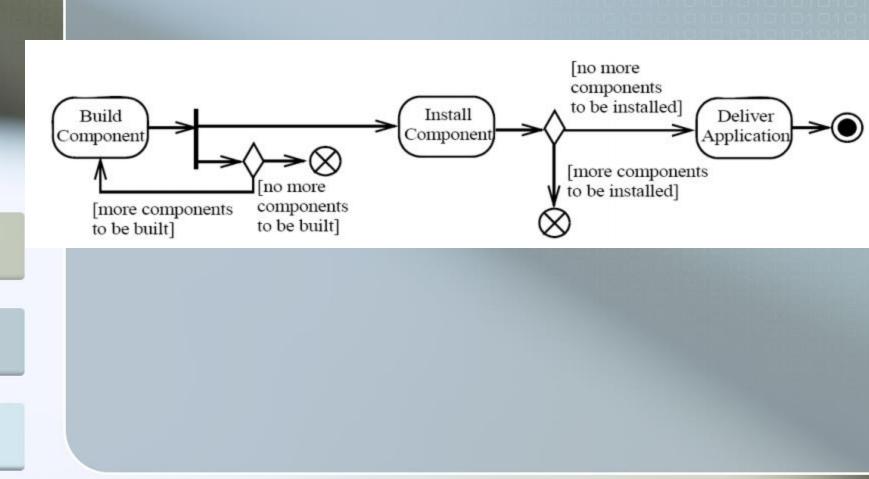


Advance Activity Diagram

Final nodes

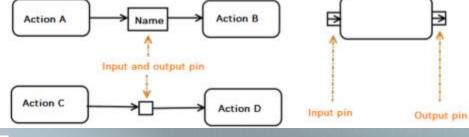
Flow final:

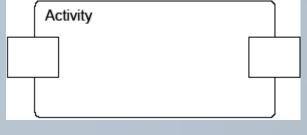
- A final node that terminates a flow.
- Destroys all tokens that arrive at it.
- It has no effect on other flows in the activity.
- Has no outgoing edges.



Object nodes

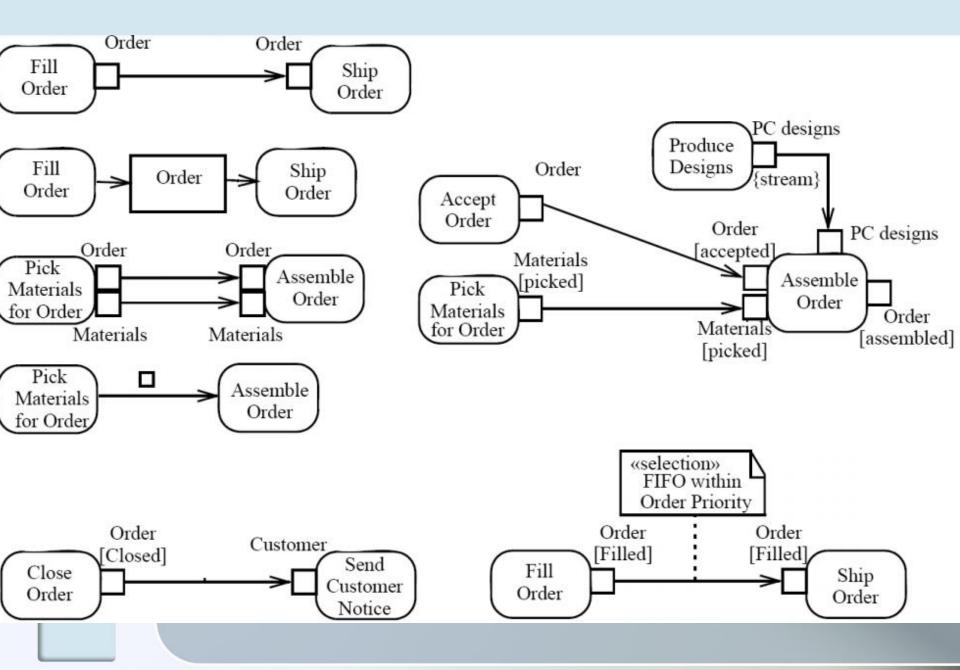
- Hold data temporarily while they wait to move through the graph
- Specify the type of values they can hold (if no type is specified, they can hold values of any type)

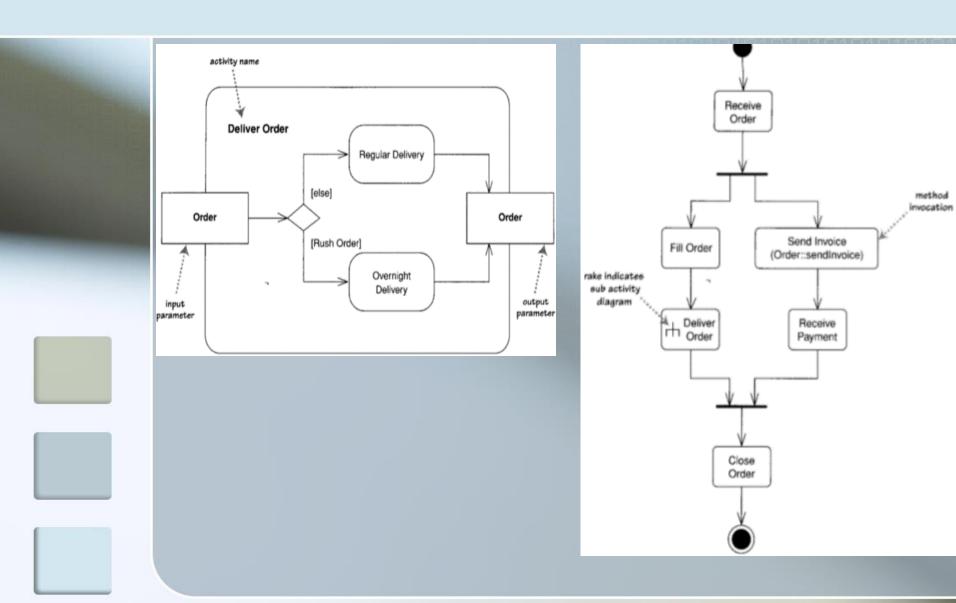




Pins (three differents notations)

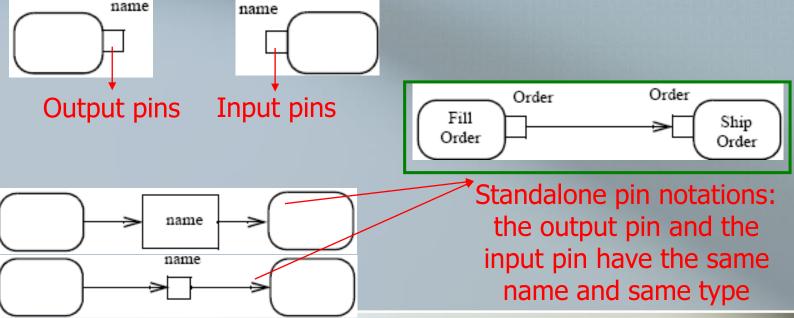
Activity Parameter Nodes



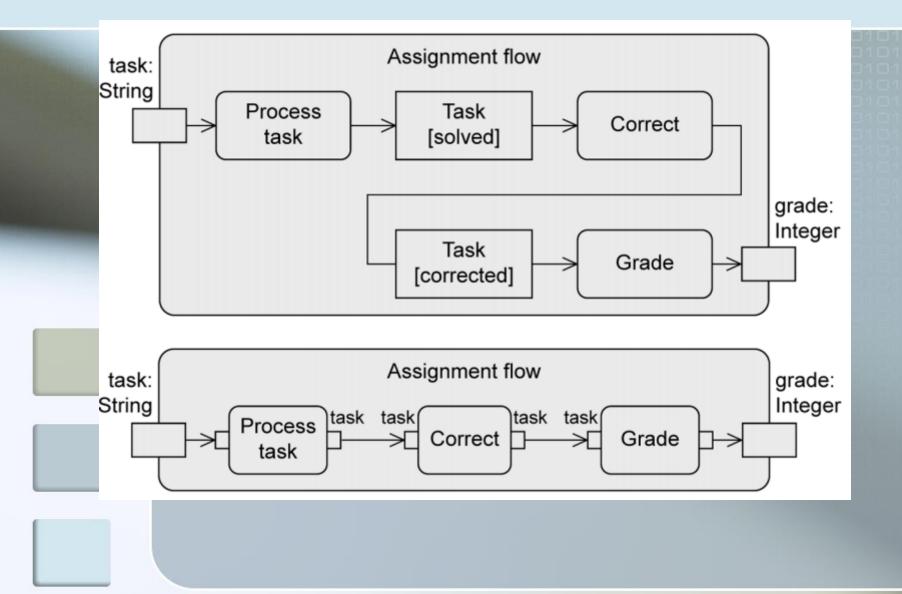


Pins

- Actions can have inputs and outputs, through the pins
- Hold inputs to actions until the action starts, and hold the outputs of actions before the values move downstream
- The name of a pin is not restricted: generally recalls the type of objects or data that flow through the pin

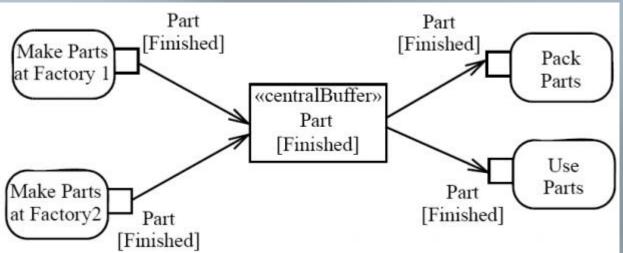


Example: Assignment Flow



Object node: Central Buffer

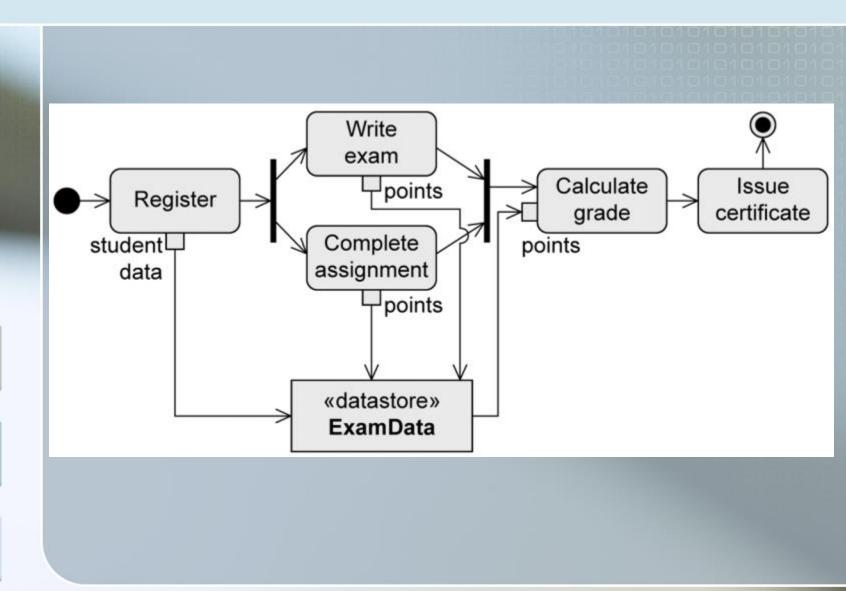
- For saving and passing on object tokens
- Transient memory
- Accepts incoming object tokens from object nodes and passes them on to other object nodes.
- When an object token is read from the central buffer, it is deleted from the central buffer and cannot be consumed again



Object nodes – Datastore

- For saving and passing on object tokens
- Permanent memory
- Saves object tokens permanently, passes copies to other nodes
- If arrives a token containing an object already present in the data store, this replaces the old one
- Tokens in a data store node cannot be removed (they are removed when the activity is terminated)

Example: Register and Withdraw

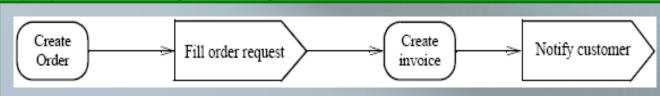


SendSignalAction

- Creates a signal instance from its inputs, and transmits it to the target object (local or remote)
- A signal is an asynchronous stimulus that triggers a reaction in the receiver in an asynchronous way and without a reply

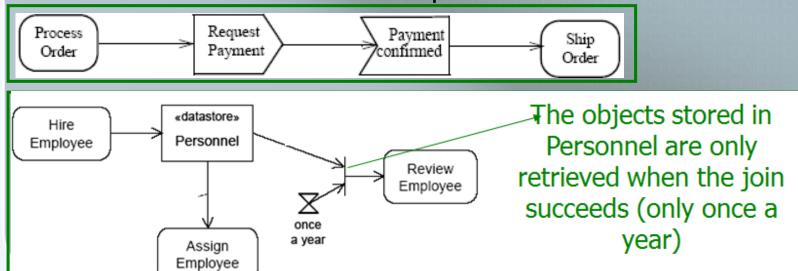
Any reply message is ignored





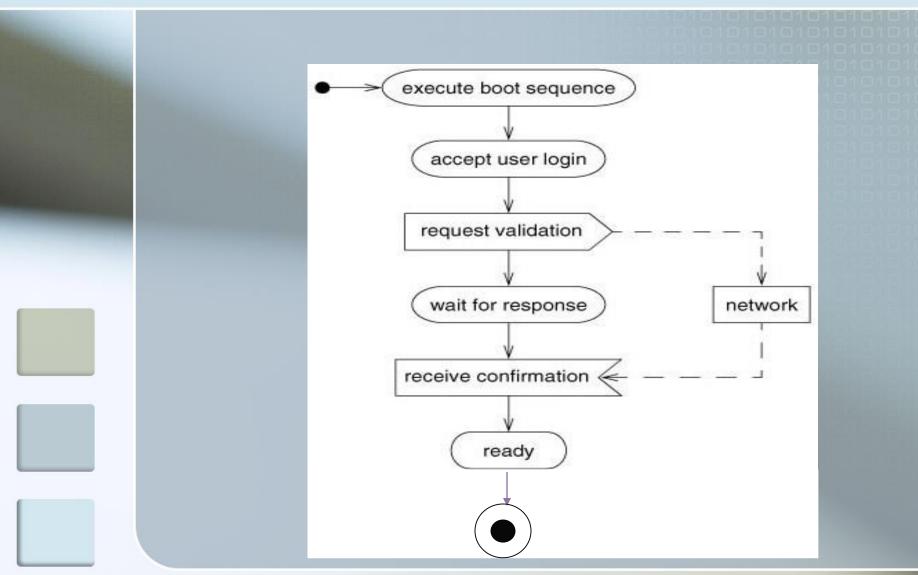
AcceptEventAction

- Waits for the occurrence of an event meeting specified conditions
- Two kinds of AcceptEventAction: Accept event action
 - Accept event action accepts signal
 events generated by a SendSignalAction
 - Wait time action accepts time events



Example

Consider a workstation that is turned on. It goes through a boot sequence and then accepts the user login. After entry of a name and password, the workstation queries the network to validate the user. Upon validation, the workstation then finishes its startup process.



Time triggers and Time events

- A Time trigger is a trigger that specifies when a time event will be generated
- Time events occur at the instant when a specified point in time has transpired
- This time may be relative or absolute
 - Relative time trigger. is specified with the keyword 'after' followed by an expression that evaluates to a time value
 - Absolute time trigger: is specified as an expression that evaluates to a time value

after (5 seconds)

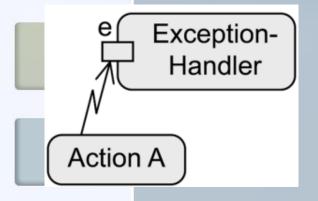
Relative time trigger

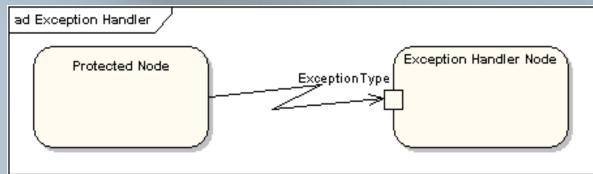
Absolute time trigger

Jan, 1, 2000, Noon

Exception Handling- Exception Handler

- Predefined exceptions.
- Defining how the system has to react in a specific error situation.
- The exception handler replaces the action where the error occurred

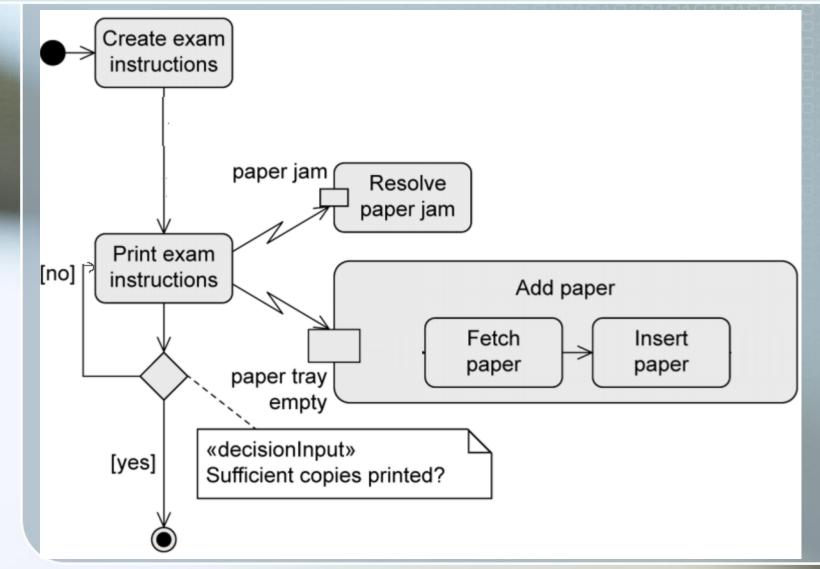




Exception Handling- Exception Handler

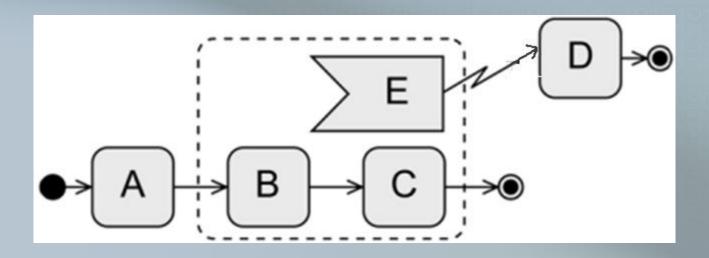
- If the error e occurs...
 - All tokens in Action A are deleted
 - The exception handler is activated
 - The exception handler is executed instead of Action A
 - Execution then continues regularly

- Your system is asked to create the exam instructions. After the creation of exam instruction, your system is ask to print the exam instruction copies. As soon as the exam instruction copies are printed your system in terminated.
- During printing the copies, your system may face the paper Jam issue or paper not available in the printer issue. If the paper are not available, your system will fetch the paper and will insert the paper into the printer.



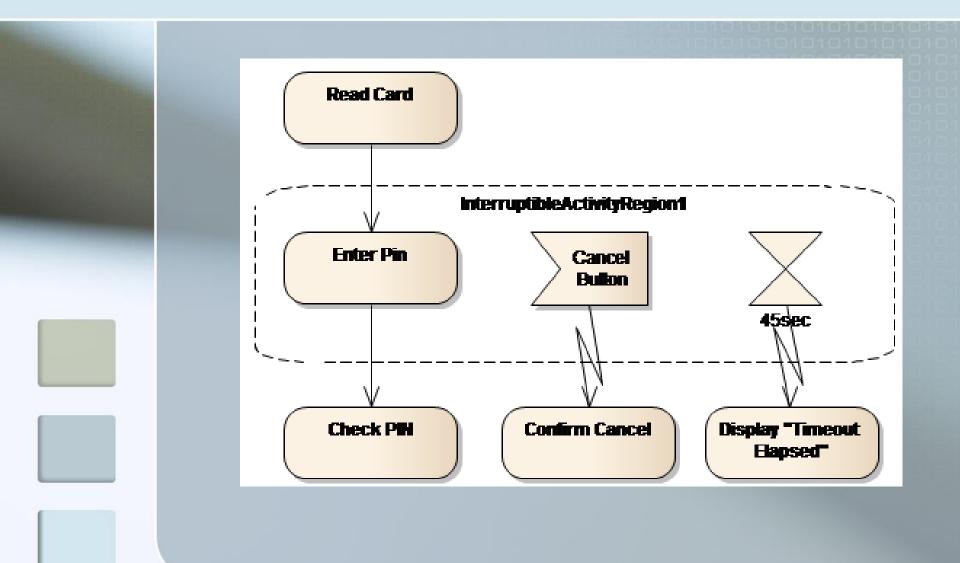
Exception Handling— Interruptible Activity Region

- Defining a group of actions whose execution is to be terminated immediately if a specific event occurs. In that case, some other behavior is executed.
- No "jumping back" to the regular execution!



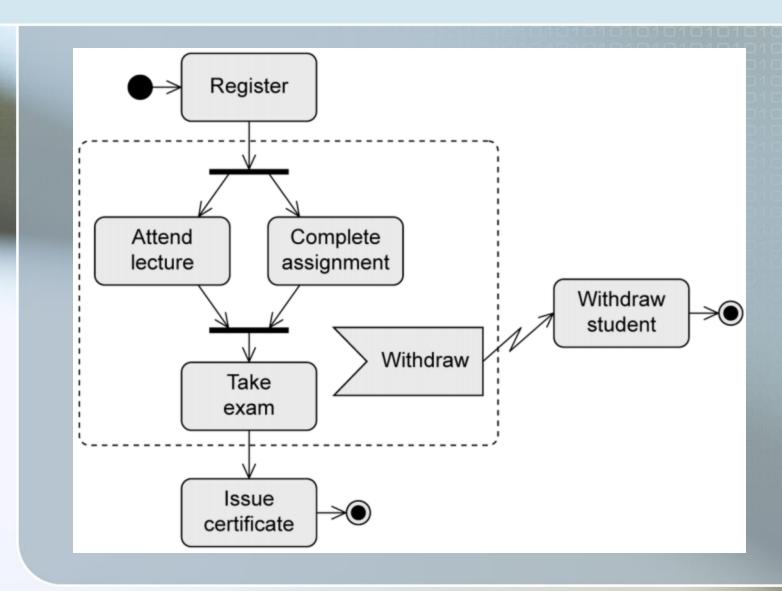
Exception Handling— Interruptible Activity Region

- if F occurs while B or C are executed
 - Exception handling is activated
 - All control tokens within the dashed rectangle (= within B and C) are deleted
 - D is activated and executed



For taking a course, students needs to do registration. After the registration, students are going to attend lectures as well as complete different assignment. After the completion of the assignment and lectures student may give the respective course exam. After giving the final exam, certificates will be issued to the students. During the entire course student may withdraw the course.

Example: Register and Withdraw



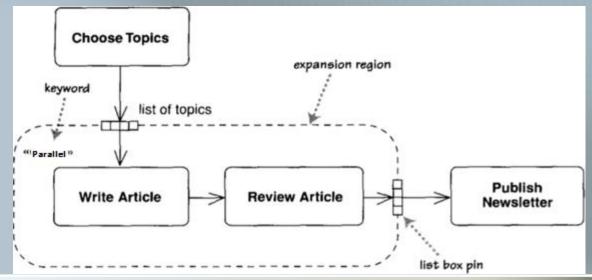
Expansion Regions

- An Expansion Region surrounds a process to be imposed multiple times on the incoming data, once for every element in the input collection.
- An expansion region is a structured activity region that executes multiple times for collection items.
- Input and output expansion nodes are drawn as a group of three / four boxes representing a multiple selection of items.

Concurrency of Multiple Executions

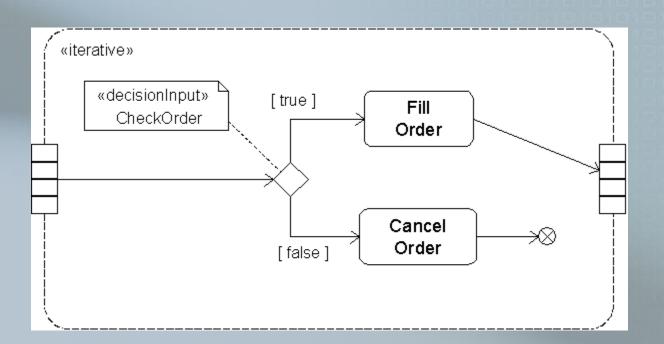
- The concurrency of the Expansion Region's multiple executions can be specified as type parallel, iterative, or stream.
- The keyword iterative, parallel or stream is shown in the top left corner of the region

reflects that the elements in the incoming collections can be processed at the same time or overlapping.



Concurrency of Multiple Executions

 An iterative concurrency type specifies that execution must occur sequentially.



Concurrency of Multiple Executions

A stream-type Expansion Region indicates that the input and output come in and exit as streams, and that the Expansion Region's process must have some method to support streams.

How to create Activity Diagram

- 1. Identify activities (steps) of a process
- 2. Identify who/what performs activities (process steps)
- Draw swimlanes
- 4. Identify decision points (if-then)
- 5. Determine if a step is in loop (*For each...,* or if-then loop)
- 6. Determine if step is parallel with some other
- 7. Identify order of activities, decision points

How to create Activity Diagram

- 8. Draw the start point of the process in the swimlane of the first activity (step)
- Draw the oval of the first activity (step)
- 10. Draw an arrow to the location of the second step
- 11. Draw subsequent steps, while inserting decision points and synchronization/loop bars where appropriate
- 12. Draw the end point after the last step.

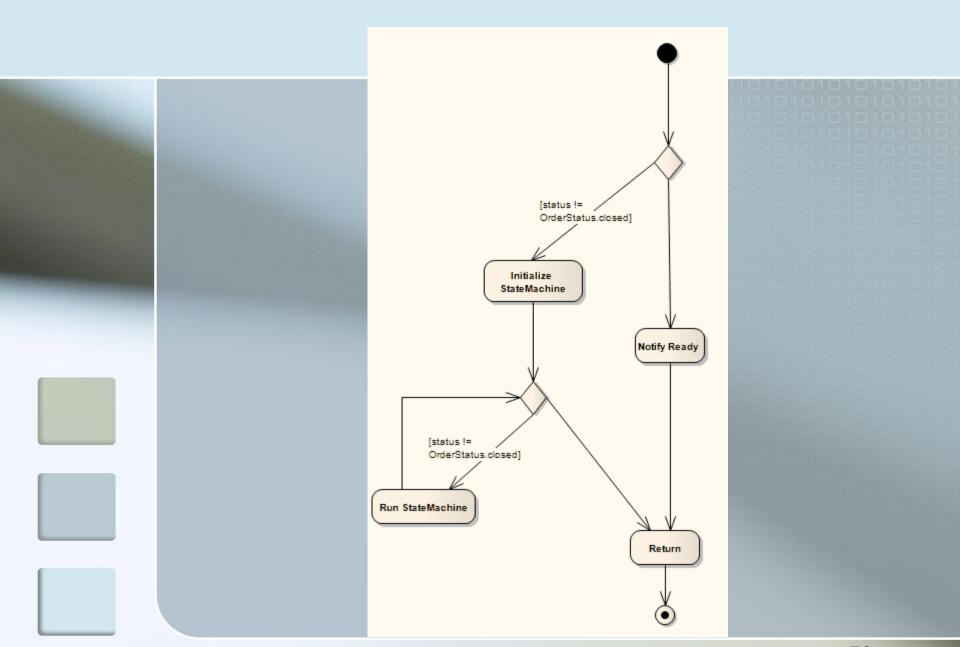
the applicant. On the basis of application, the system verifies the application completeness i.e., either the data given in application is completed or not. On the basis of application completeness and personal history, the system checks if the applicant is eligible or not. Applicant who are not eligible are rejected from the further process. Those applicant who are eligible for the specific post, are processed further. After accepting all the eligible applicant applications, the valid applicants are stored in the database and higher management review the valid applicant using the system. At the end system generates the email to the selected applicant. During the higher management review, the system may get failed due to I/O or the SQL exception, which the system is capable to handle.

The organization is hiring employees for its organization. For

this purpose, an automated system of an organization must

accept multiple application as well as the personal history of

```
public void doCheckForOutstandingOrders()
         if (status != closed)
              initializeStateMachine(); while (status !=
              closed)
                   runStateMachine();
         else
                  Ready();
         return;
```



Ticket Purchasing

Activity is started by Commuter actor who needs to buy a ticket. Ticket vending machine will request trip information from Commuter. Based on the info machine will calculate payment due and request payment options. After payment is complete, ticket is dispensed to the Commuter.

Document Management System

Document is created, reviewed, updated, approved, and at some point archived. This activity diagram should show responsibilities of different roles and a flow of document changes. Partitions represent different roles participating in the activity - Author, Reviewer, Approver, and Owner.

Quotation of a software

• Customer request for the quotation of the software from the salesperson. salesperson check notes for the requirement information, if the salesperson needs help in requirement, he ask technical expert to check the requirement and enter the data into the system, otherwise salesperson himself enter the data in to system. System then calculates the quote and the final quote is then reviewed by the customer, if its ok then customer will accept the quote else if any changes are required then again salesperson will develop a notes for requirements.

Online Shopping

 Online customer can browse or search items, view specific item, add it to shopping cart, view and update shopping cart, do checkout. User can view shopping cart at any time.

Creating Document

- Open the word processing package.
- Create a file.
- Save the file under a unique name within its directory.
- Type the document.
- If graphics are necessary, open the graphics package,
- If graphics are necessary, open the graphics package, create the graphics, and paste the graphics into the document.
- If a spreadsheet is necessary, open the spreadsheet package, create the spreadsheet, and paste the spreadsheet into the document.
- Save the file.
- Print a hard copy of the document.
- Exit the word processing package.

Enrollment in University

- An applicant wants to enroll in the university.
- The applicant hands a filled out copy of form U113 University Application Form to the registrar.
- The registrar inspects the forms.
- The registrar inspects the forms.
- The registrar determines that the forms have been filled out properly.
- The registrar informs student to attend in university overview presentation.
- The registrar helps the student to enroll in seminars
- The registrar asks the student to pay the initial.

Business Process of Meeting a New Client for create a proposal.

- A salesperson calls the client and sets up an appointment.
- If the appointment is onsite (in the consulting firm's office), corporate technicians prepare conference room for a presentation
- If the appointment is offsite(at the client's office), a consultant prepares a presentation on a laptop.
- The consultant and the salesperson meet with the client at the agreed-upon location and time.
- The salesperson follows up with a letter.
- If the meeting has resulted in a statement of a problem, the consultant create a proposal and sends it to the client.