Software Engineering

Activity Diagrams

Dr. Sayed AbdelGaber
Professor
Faculty of Computers and Information
Helwan University

What are they?

- A diagramming technique that models different dynamic aspects of a system.
- Represents flow for control required to carry out:
  - ✓ High-level → business activities.
  - ✓ Low-level → internal details of an operation.
- 'Essentially a flowchart, showing flow of control from activity to activity'.
- Notation richer than flowcharts through use of:
  - ✓ States
  - ✓ Swim lanes
  - ✓ Object flows
  - ✓ Control icons

Note on workflow

- ‘The flow of control through a processing activity as it moves among people, organizations, computer programs, and specific processing steps.‘

[Source: Satzinger et al., 2000]

- Work flow describes the sequence of steps to perform a particular activity.

- Collection of ordered tasks and related decisions to achieve a particular goal.

Purpose

- **Represent high level view of business activities:**
  - ✓ **Describing workflows in the development of a business model.**
  - ✓ **Visualize the workflow of a business use case.**
- **Describe dynamic complex behavior of an object.**
- **Describe the flow within an individual operation.**
- **Complement interaction diagrams:**
  - ✓ **Recall difference in perspective.**
- **Document development tasks performed by development organization.**

Advantages

- Able to describe complex flows and use cases.
- Can identify pre- and post- conditions for use cases.
- Relatively simple to construct and interpret.
- Can be used at different times in development process:
  - ✓ Analysis → business activities
  - ✓ Design → operation flow
- Can be presented to stakeholders for checking and confirmation.
- Can be used for forward, reverse, or roundtrip engineering.

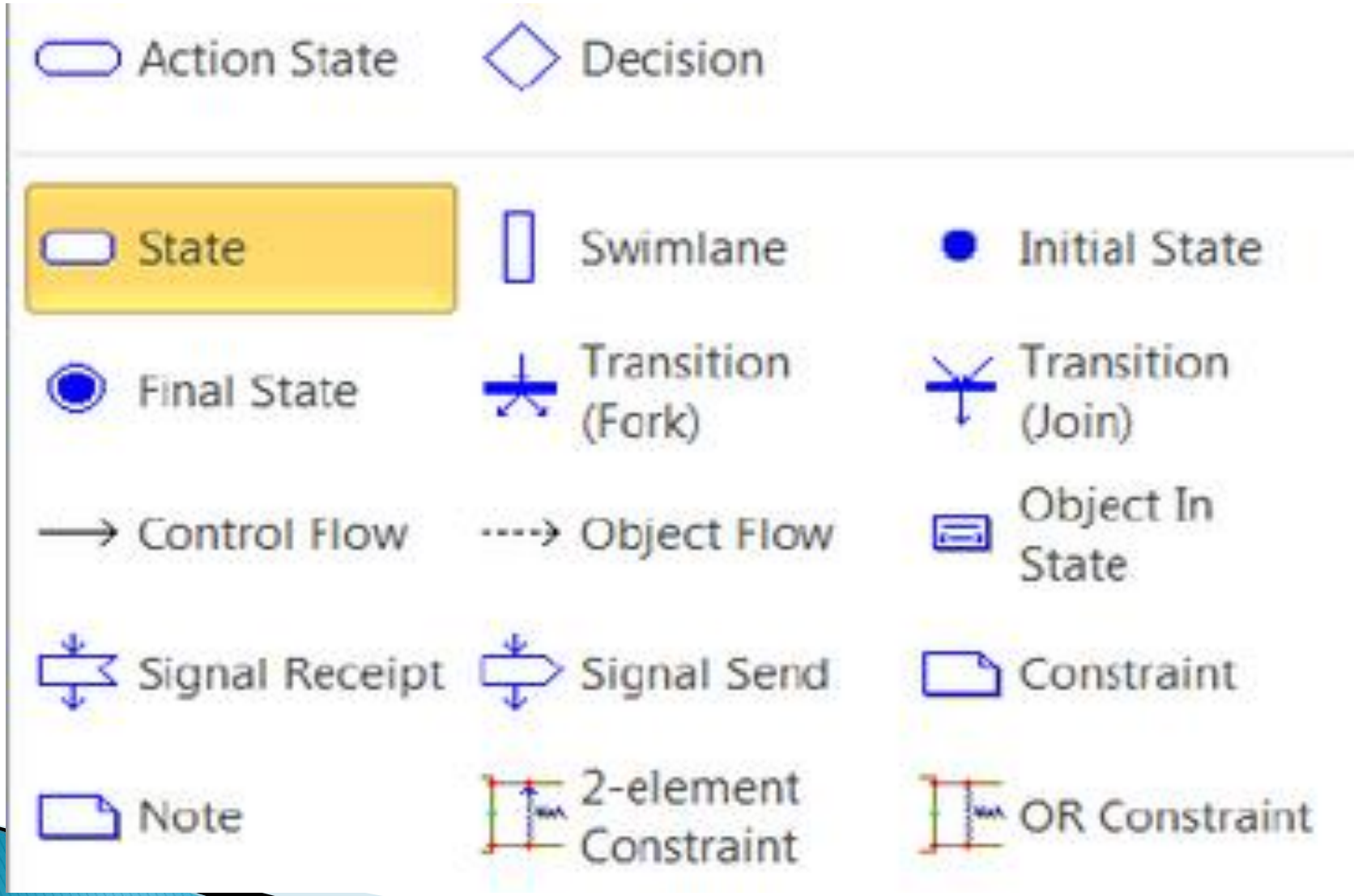
Relationship with other UML diagrams

- **Use case diagrams:**
  - ✓ Documents (realizes) a single use case.
  - ✓ Documents multiple scenarios for a single use case.
    - Primary path.
    - Alternative paths.
  - ✓ Can identify pre- and post- conditions for a use case:
    - Related to states.

- **Class diagrams:**
  - ✓ Shows impact on objects from specific events and decisions in the form of dependencies.
  - ✓ Complex behavior for a class.

- **Interaction diagrams:**
  - ✓ Shows dynamic aspects of 'a society of objects'.

Activity diagram notations



Activities

- ‘is a unit of work to be carried out’
- Take place over a period of time:
  - ✓ Weeks
  - ✓ Milliseconds
- Label describes actions taking place in activity.
- Actions:
  - ✓ On Entry: triggered as soon as activity starts.
  - ✓ Do: collection of actions that take place during lifetime.
  - ✓ On event: triggered in response to an event.
  - ✓ On exit: take place just before activity completes.

Transitions

- **Represents movement from one activity to another:**
  - ✓ **Once an activity is completed, flow of control is immediately moved to the another.**
- **Denoted by a simple directed line.**
- **Typically triggerless:**
  - ✓ **They do not require an explicit trigger event.**
- **Can also set conditions for transitions:**
  - ✓ **Guard conditions.**
  - ✓ **Evaluate to true for transition to occur.**

State

- **Represent certain condition of system or environment on entry and exit of workflow.**

- **Start state:**
  - ✓ **Single entry point to flow.**
  - ✓ **Denoted by black dot.**
  - ✓ **Matches pre-condition in use case description.**

- **End state:**
  - ✓ **Can have multiple endings.**
  - ✓ **Denoted by bulls-eye shape.**
  - ✓ **Goal matches post-condition in use case description.**

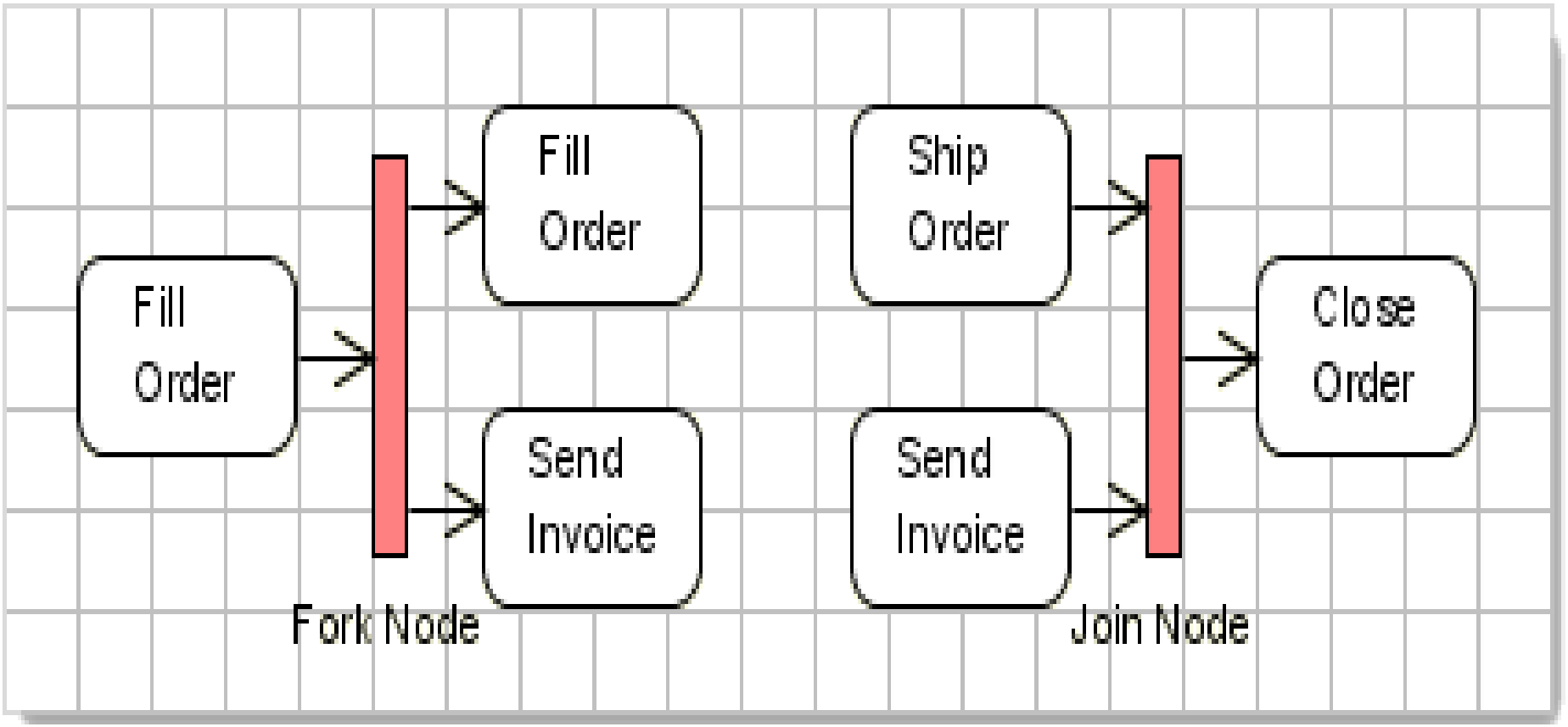
Decision point

- Used to represent conditional branching.
- An exit transition that results in alternative directions depending on a Boolean condition.
- Can result in two or more different transitions.
- Denoted by diamond shape.
- Each new transition branching from decision point requires a guard condition:
  - ✓ Example: [book incomplete]
- Document logic in operation.

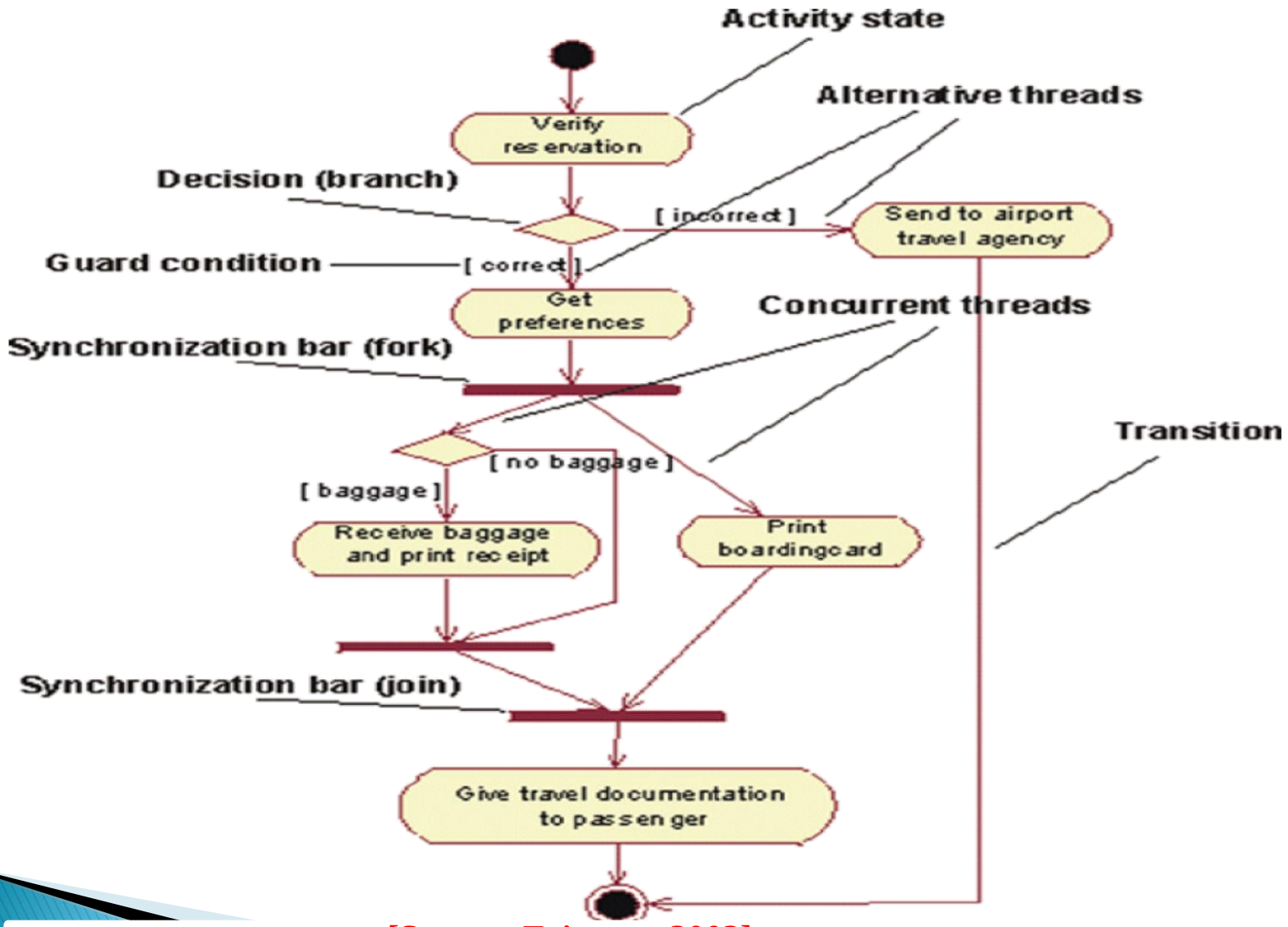
Fork and joins

- Represents a number of activities running in parallel.
- Transitions can be split into multiple (concurrent) paths.
- These multiple paths can be recombined into a single transition.
- If a workflow is split then it must be recombined on the same diagram.

Forks and joins example



Forks and joins example



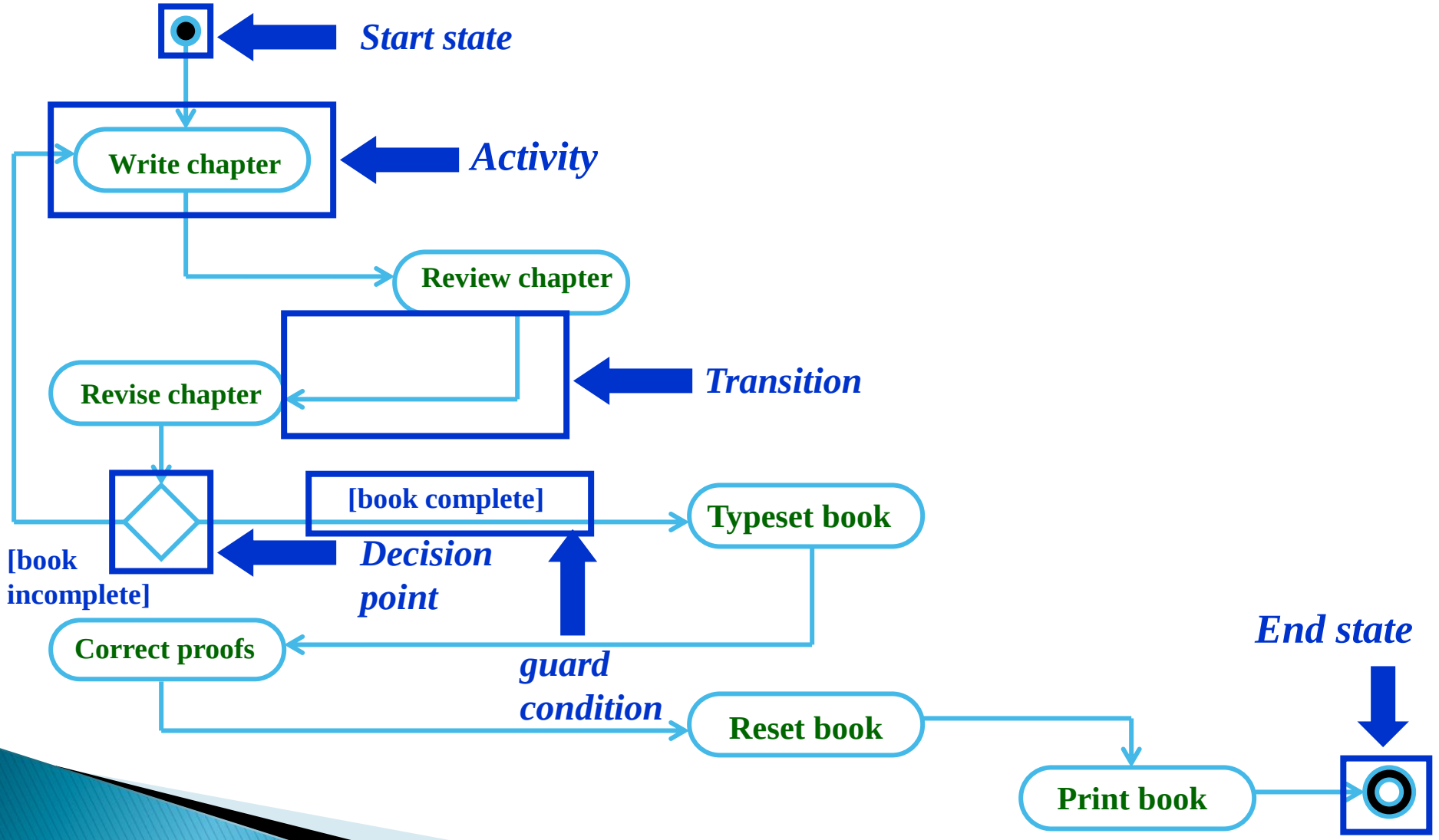
[Source: Ericsson, 2002]

Producing a book example

1. Author writes chapter.
2. Reviewer reviews chapter.
3. Author revises chapter.
4. If book not complete:
   - ✓ author writes another chapter.
5. If book complete:
   - ✓ typesetter typesets book.
6. Author corrects proofs.
7. Typesetter resets book.
8. Printer prints book.

[Adapted from: Bennet et al., 2002]

Producing a book diagram



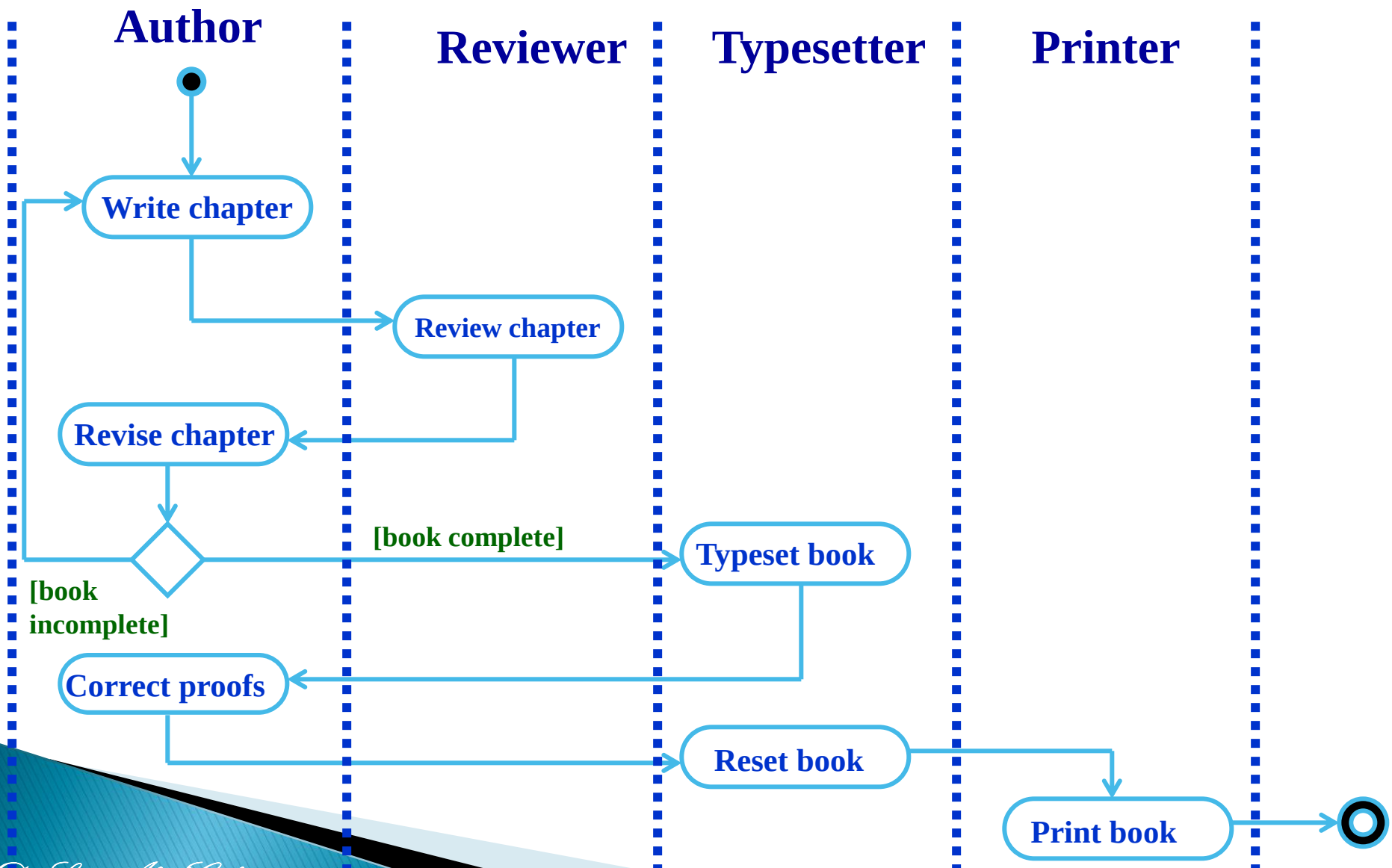
Swim lanes

- Partitioning of activities to show who or what is responsible for carrying out activity.

- Can represent:
  - ✓ business organizations
  - ✓ departments
  - ✓ People → actors

- Can simplify complex systems by grouping related activities.

Swim lanes example

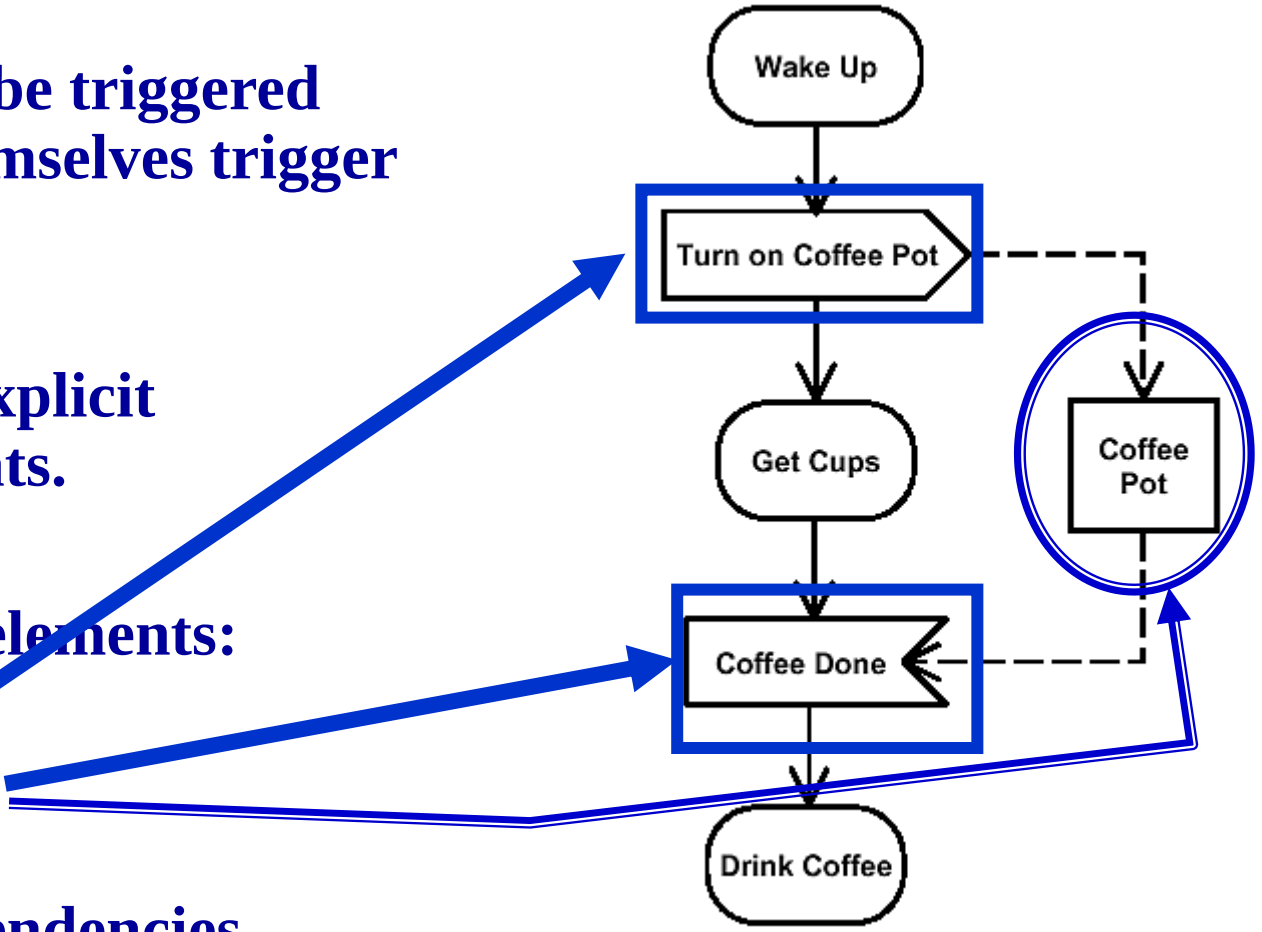


Object flows

- Activity diagrams seem removed from objects.
- Objects will carry out activities in resulting system.
- Can show what objects are affected by linking to a particular activity using dependencies.
- Useful for showing which workflows change the state of objects.
- Explicit state changes documented in state chart diagram.

Control icons

- Transitions can be triggered by events or themselves trigger events.
- Can represent explicit handling of events.
- Two additional elements:
  - ✓ Send signal
  - ✓ Receive signal
- Can record dependencies between events and objects.



[Source: OMG UML 1.4 tutorial]

Development approach

- Discover actors and use cases.
- Identify scenarios:
  - ✓ Including primary and alternative paths.
- Combine scenarios to produce comprehensive workflow → Activity diagrams.
- Add object flows where object behavior trigger by workflow.
- Map business flows to activities using swim lanes.
- Refine complex business activities in same way.

[SOURCE: Bennet et al., 2001]

Example 2

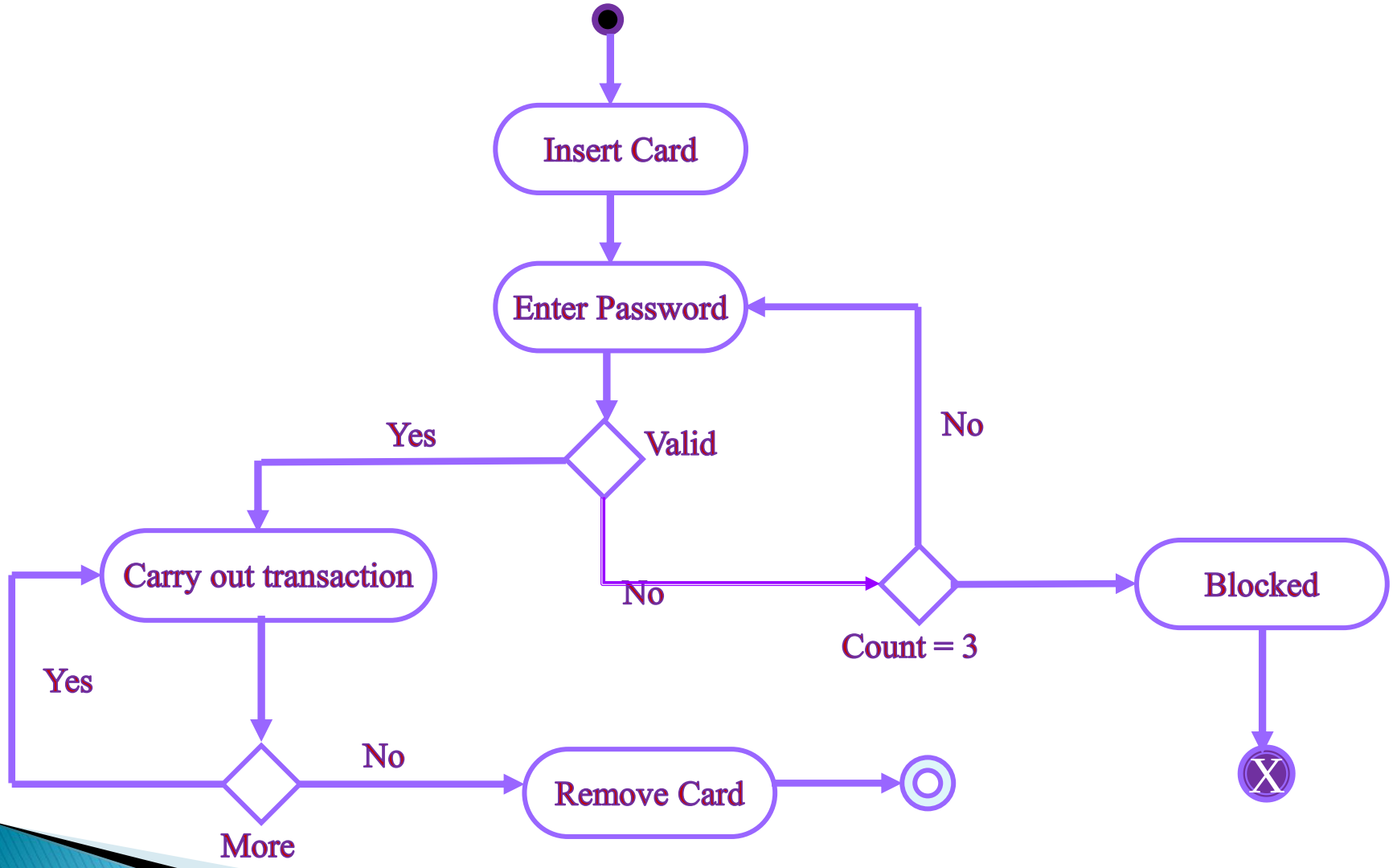
Consider the following description of the processes involved when a customer withdraws cash from an ATM. The customer inserts their card into the ATM machine which then prompts them to enter their PIN. After the

customer enters their PIN the ATM then checks the validity of the PIN entered. If the PIN is incorrect, the ATM machine displays a message inviting the customer to re-enter their PIN, but if the customer enters a PIN incorrectly three times, the ATM retains the card and the transaction is terminated. If the PIN is valid, the ATM displays a prompt for the customer to enter the required amount. The bank then checks the amount entered against the balance of the account. If the amount is greater than the balance, the ATM displays the available balance and allows the customer to either cancel the transaction or enter a new amount. If the amount is less than or equal to the balance, the ATM ejects the card while the bank debits the account. The customer then removes the card, the ATM ejects the cash and the customer takes the cash.

Activity diagram – example atm

- ATM Machine Operation
  - Step 1
    - You insert the card
  - Step 2
    - You provided the pin/password
  - Step 3
    - If password is valid then you have to choose the transaction
    - If password is invalid then you have to provide the password again
    - If you provide wrong password more than three times card is blocked
  - Step 4
    - You carry out the transaction in case password is valid
  - Step 5
    - You are asked for more transactions
  - Step 6
    - Remove the card
  - Process is over

Activity diagram - ATM Machine Operation



ACTIVITY DIAGRAM HOMEWORK

- **Patient Visits a Diabetes Hospital**
  - **Step 1**
    - **Doctor sees the Patient and studies Patient History**
  - **Step 2**
    - **Doctor prescribes Lab-Test**
  - **Step 3**
    - **Patient shows symptoms or no symptoms for disease**
  - **Step 4**
    - **For no symptoms process ends**
  - **Step 5**
    - **For symptoms : check BP, Classify Diabetic Type, Classify Patient work type**
  - **Step 6**
    - **prescribe Treatment**
  - **Step 7**
    - **Process Ends**

Lecture Summary

- Activity Diagrams are good for describing synchronization and concurrency between activities
- Activity diagrams are useful for capturing detailed activities, but they can also capture elements of the high level workflow the system is intended to support
- Partitioning can be helpful in investigating responsibilities for interactions and associations between objects and actors

Questions