Introduction to
SOFTWARE ENGINEERING

Nguyễn Huy Hoàng - Phan Gia Phước

[09 . 2021]

Adapted from the Slides of Software Engineering, 10th Ed. by Ian Sommerville

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Chapter 4.

REQUIREMENTS ENGINEERING (cont.)

4.7 (cont.)

ACTIVITY DIAGRAM

Objectives

- When you complete this module, you should be able to:
  - Demonstrate the use of activity diagrams in requirement modeling.
  - Apply basic and advanced elements of activity diagram notations.
  - Demonstrate the use of activity diagrams in requirement modeling.
  - Awareness of typical mistakes made by students.

What is an Activity Diagram?

What is an Activity Diagram?

- An activity diagram in the use-case model can be used to capture the activities and actions performed in a use case.
- It is essentially a flow chart, showing flow of control from one activity or action to another.
- It consists of actions, nodes and transitions between activities and states.

Activity Diagrams

- Model business workflows.
- Identify candidate use cases, through the examination of business workflows.
- Identify pre- and post-conditions for use cases.
- Model workflows between/within use cases.

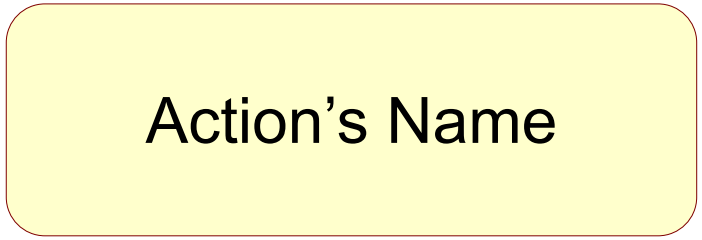
The notation | The basics

Activities

- An Activity is the process being modeled.
- An Activity is a unit of work that needs to be carried out.
- Any Activity takes time.

Actions

- An Action is **a step** in the overall activity.
- The work can be documented as Actions in the activity.

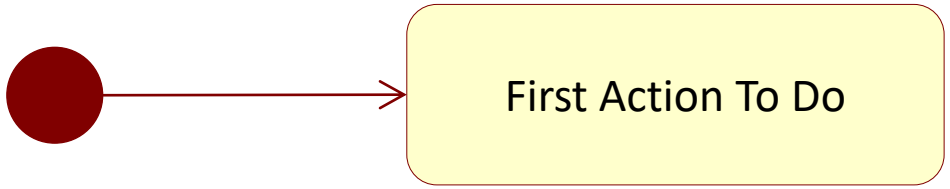


Transitions

- With arrows indicating direction, the **transition** lines on an activity diagram show the sequential flow of actions in the modeled activity.
- The arrow will always point to the next action in the activity's sequence.

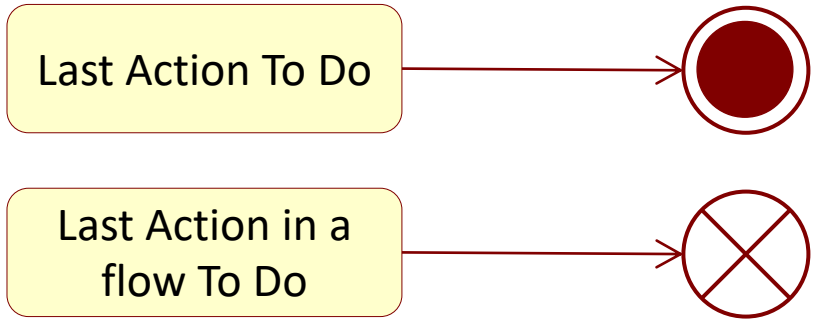
Initial Nodes

- The **initial node** clearly shows the starting point for the action sequence within an activity diagram.
- The initial node is drawn as a solid circle with a transition line (arrow) that connects it to the first action in the activity's sequence of actions.
- There can be **more than one** initial node. In this case, invoking the activity starts multiple flows, one at each initial node.
- There can be **only one** transition line connecting the initial node to an action.



Final Nodes

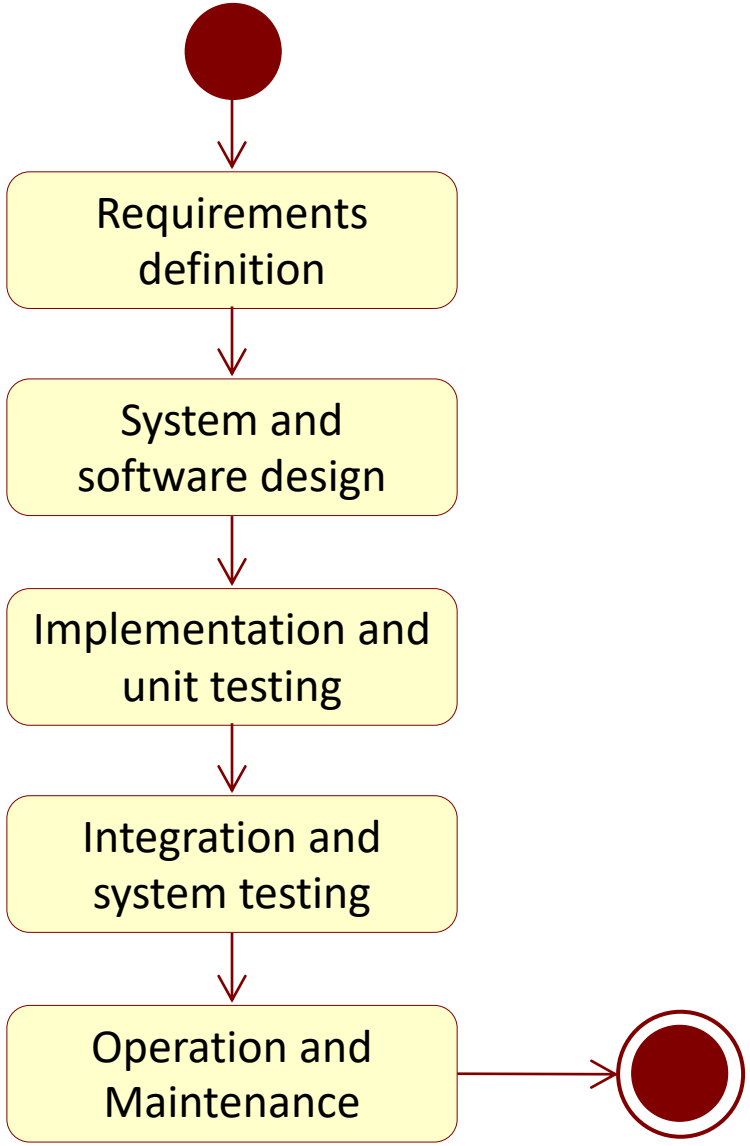
- **Final node** indicates that the activity's action sequence or a flow has reached its end.
- There are two types of final nodes: activity final node and flow final node.
- **Activity final node** is drawn as a circle surrounding a smaller solid circle.
- **Flow final node** is drawn as a circle with a cross inside.



Final Nodes (cont.)

- An **activity final node** stops all flows in an activity and terminates the entire activity.
- A **flow final node** terminates a path through an activity, but not the entire activity.
- Every activity diagram should have **at least one** final node symbol.
- It is possible for an activity diagram to show **multiple** final nodes. In other words, the activity may terminate in different manners.

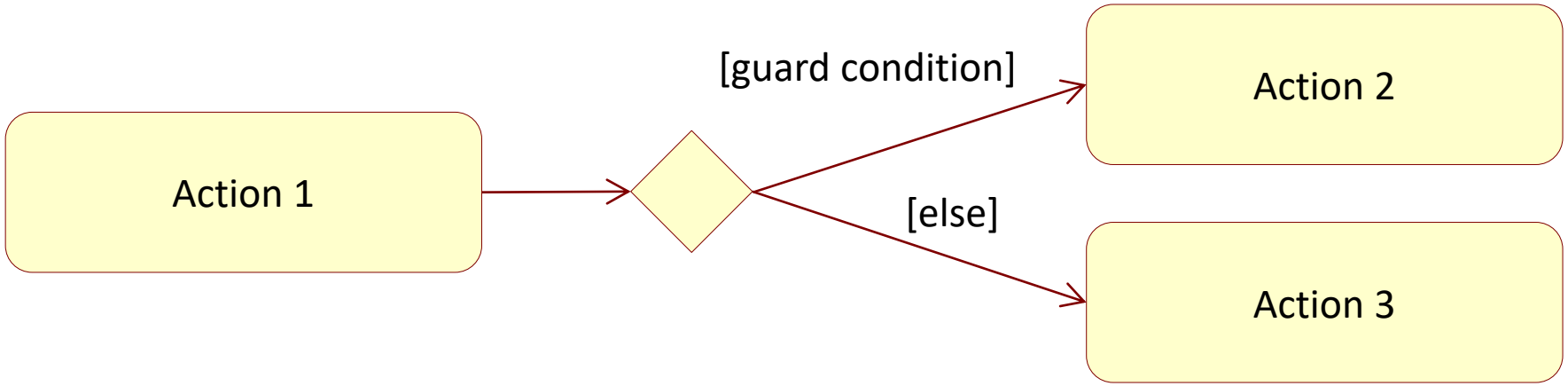
Example


Requirements definition
System and software design
Implementation and unit testing
Integration and system testing
Operation and Maintenance

The notation | Beyond the basics

Decision nodes

- A **decision** node shows where the exit transition from a action may branch in alternative directions depending on a condition.
- A **decision** is drawn as a diamond on an activity diagram.
- Since a decision will have **at least two different outcomes**, the decision symbol will have multiple transition lines connecting to different actions.

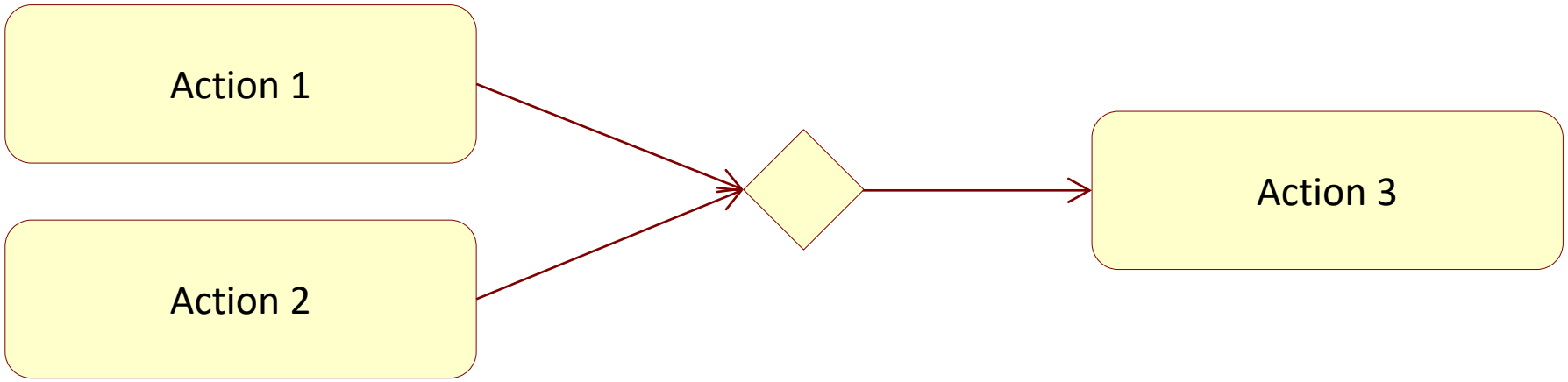


Decision nodes > Guard conditions

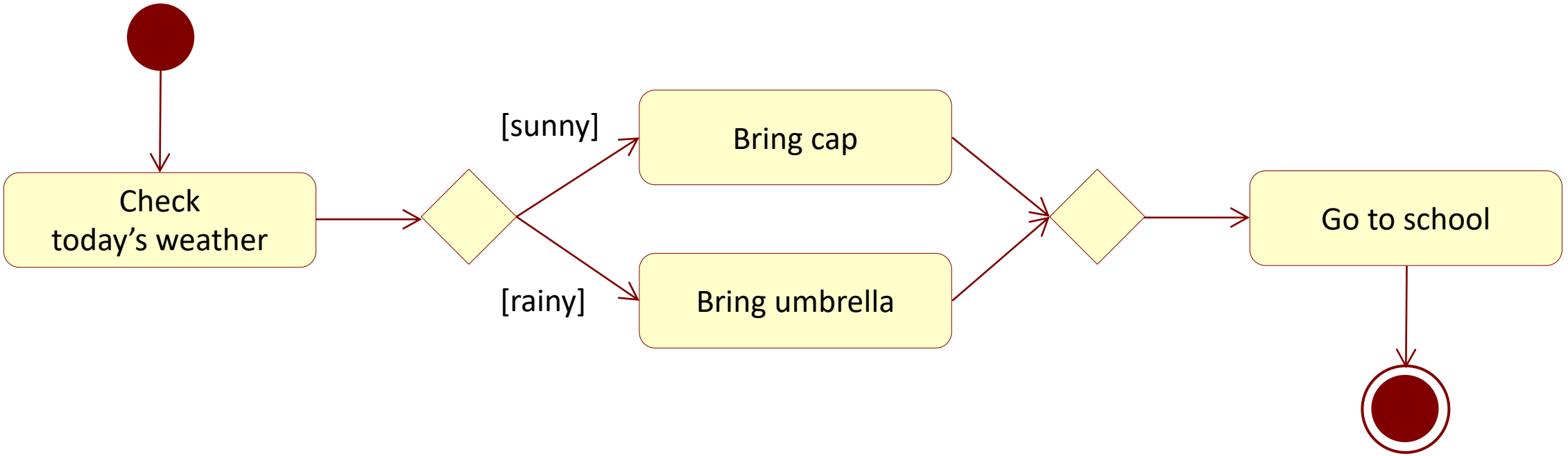
- A **guard condition** explicitly tells when to follow a transition line to the next action.
- **Guard condition text** is always placed in brackets.
  For example, [guard condition text].
- The **[else]** guard is commonly used in activity diagrams to mean "if none of the other guarded transition lines matches the actual condition," then follow the [else] transition line.

Merge nodes

- A **merge** node brings together alternate flows into a single output flow.
- A **merge** node using the same diamond icon with multiple paths pointing to it, but with only one transition line coming out of it.
- A **merge** does not synchronize multiple concurrent flows.



Example

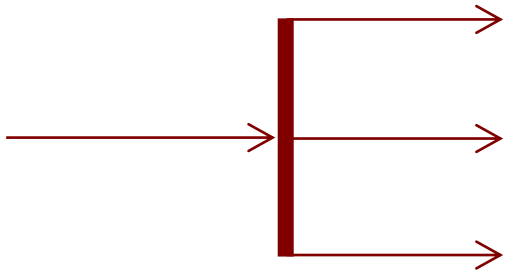

Check
today's weather
[sunny]
Bring cap
[rainy]
Bring umbrella
Go to school

Synch states

- Certain action sequences can be done in parallel.
- Parallel action sequences are officially named **synch states**.
- A synch state is where a transition **forks** into multiple paths or multiple paths are **joined** into a single transition.

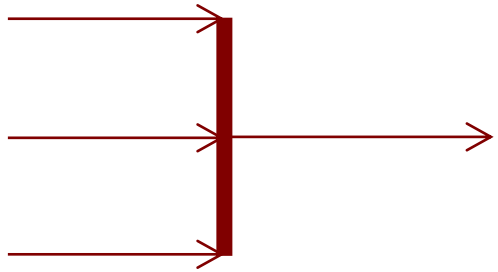
Fork Nodes

- A **fork** is where a path splits.
- A **fork** node splits incoming flow into multiple **concurrent** flows.

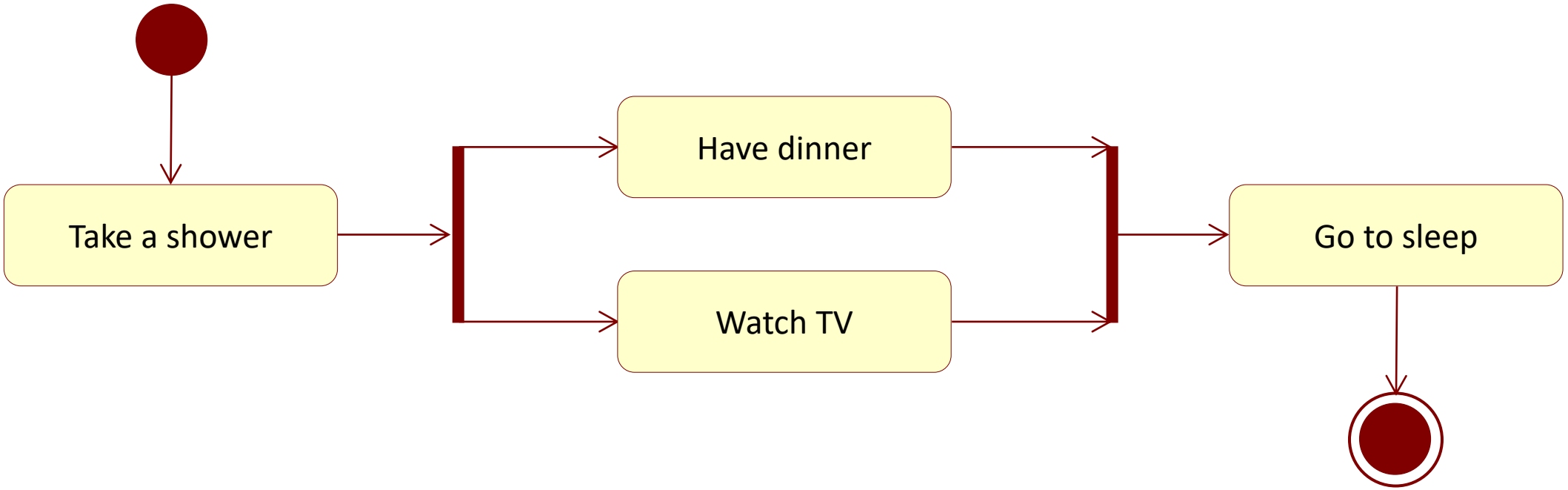


Join Nodes

- A **join** is where multiple concurrent paths meet.
- A join synchronizes multiple inflows and produces a single outflow. The outflow from a join cannot execute until all inflows have been received.



Synch states > Example

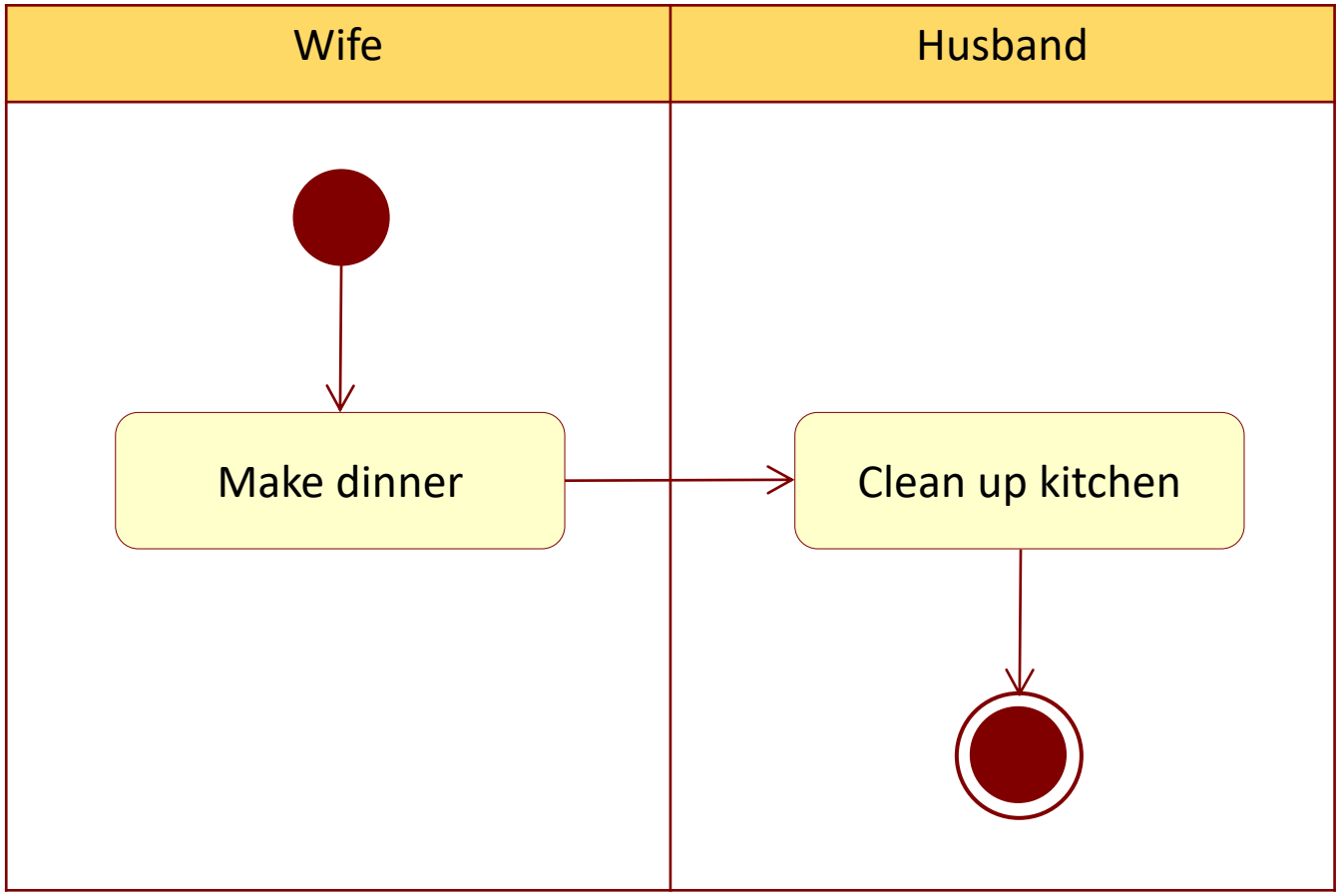


Swimlanes

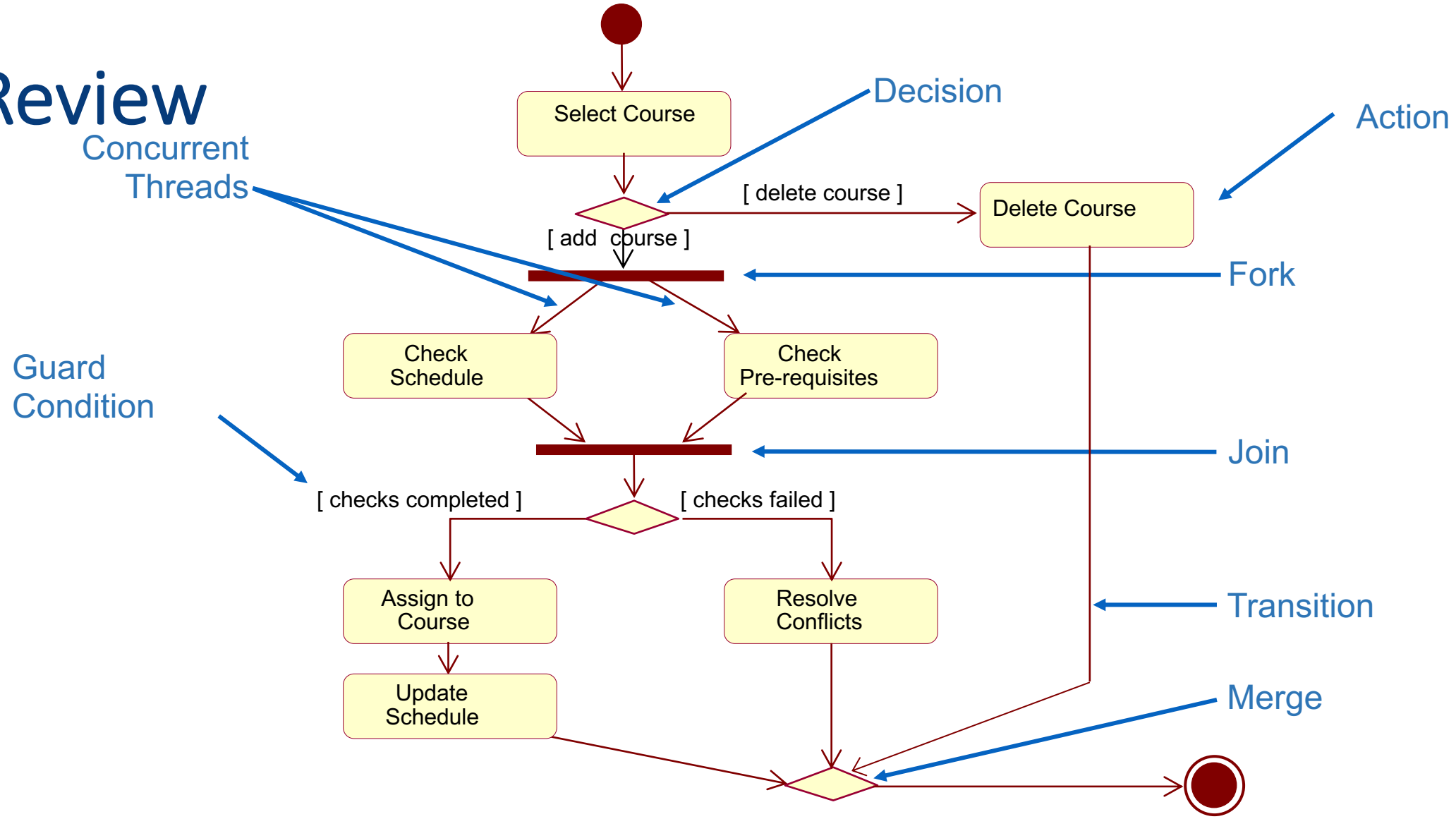
- Swimlane is used to model the activity's procedural flow of control between the objects (persons, organizations, or other responsible entities) that actually execute the action.

Object 1	Object 2	Object 3

Swimlanes



Review


Concurrent Threads
Decision
Action
Select Course
[delete course]
Delete Course
[add course]
Fork
Guard Condition
Check Schedule
Check Pre-requisites
Join
[checks completed]
[checks failed]
Assign to Course
Resolve Conflicts
Transition
Update Schedule
Merge

Thanks for your attention!

FACULTY OF
INFORMATION TECHNOLOGY
THU DUC COLLEGE OF TECHNOLOGY

Phone: (+848) 22 158 642
Email: fit@tdc.edu.vn
Website: fit.tdc.edu.vn
Facebook: facebook.com/tdc.fit
Youtube: youtube.com/fit-tdc