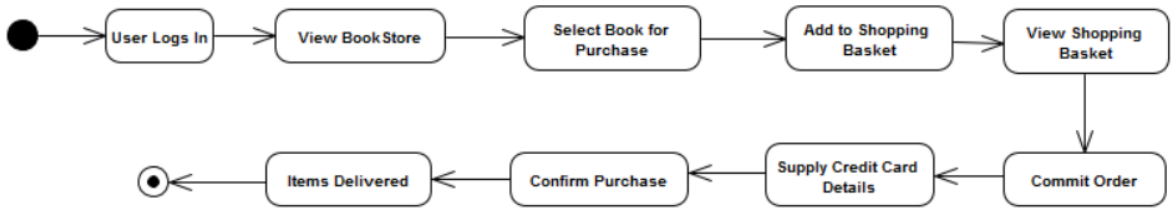
UML Activity Diagram

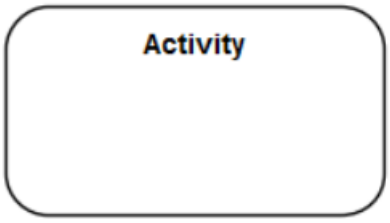
Activity Diagram

- Describes the workflow behavior of a system, focuses on flows.
- A Process describes a sequence or flow of Activities in an organization with the objective of carrying out work.



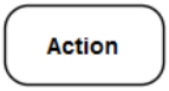
Activity

- describes a **sequence of actions** based on control models and object flow models
- contains edges and activity nodes (e.g. actions)
- represented by a rectangle with rounded corners



Action

- is a fundamental unit of executable functionality contained within an Activity
- represents a **single step** within an activity



Activity contains **nodes**:

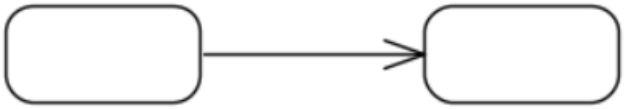
- Action
- Object
- Control Node

And **edges**:

- Control Flow
- Object Flow

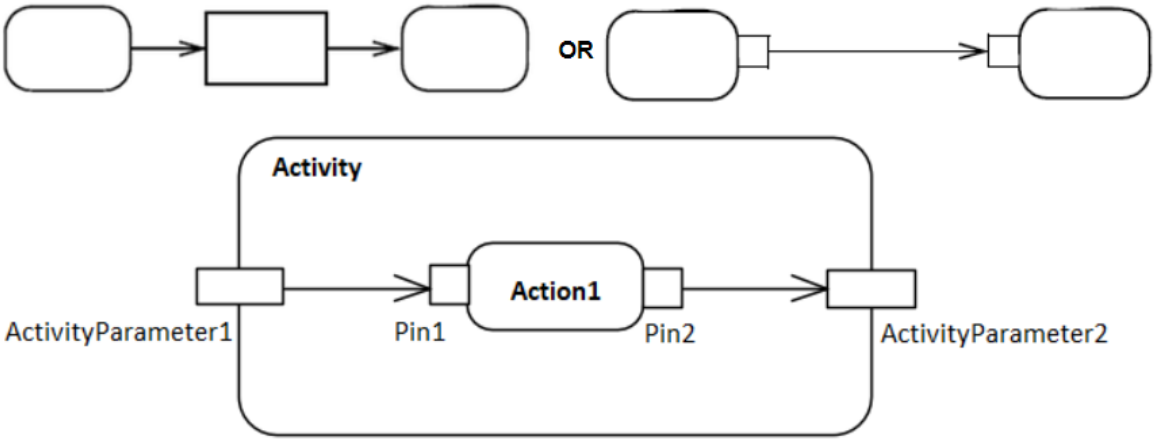
Control Flow

- A control flow is an edge that starts an activity node after the previous one is finished



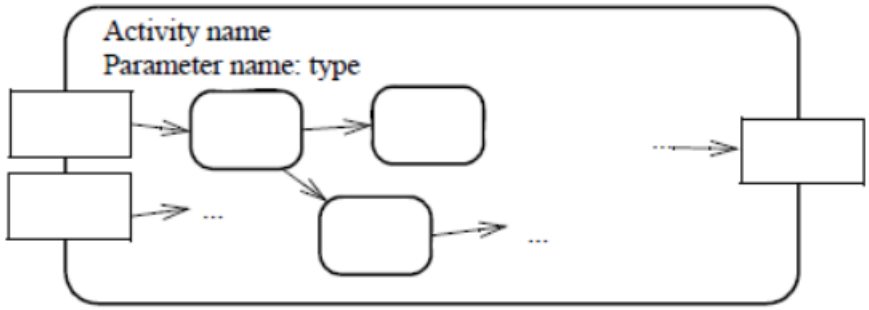
Object Flow

- An object flow is an activity edge that can have objects or data passing along it

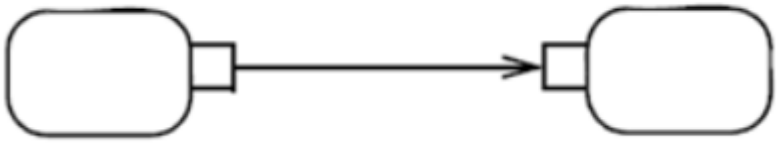


Objects

- Incoming or outgoing objects are parameters
- Placed (as rectangles) on the border



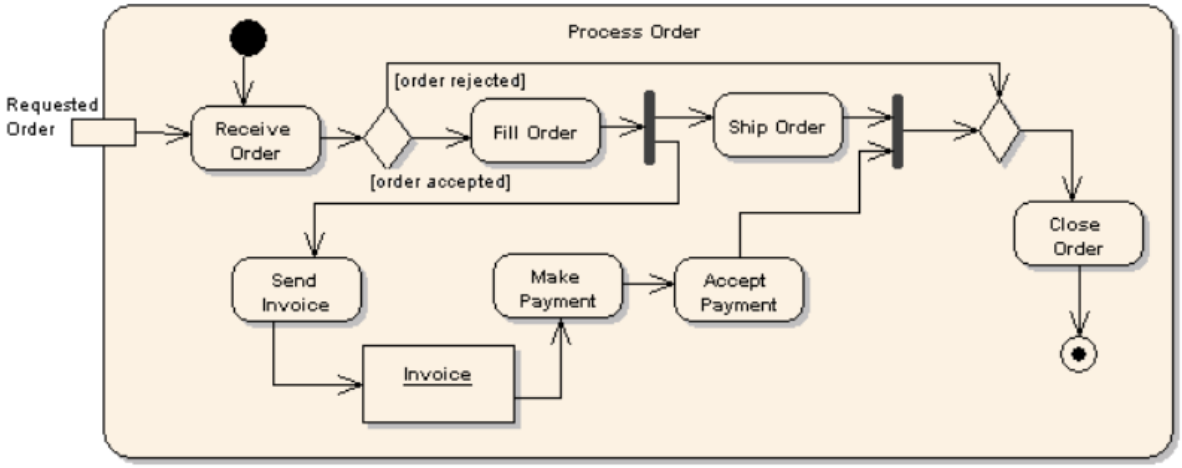
- Or as small rectangle, called a pin (**input** or **output**)



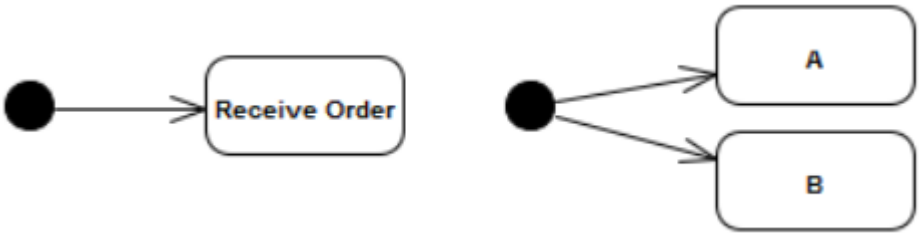
Token

- The semantics of an activity is based on a token flow
- Control tokens and object tokens flow **between** nodes **over** edges

Activity Example



Initial Node

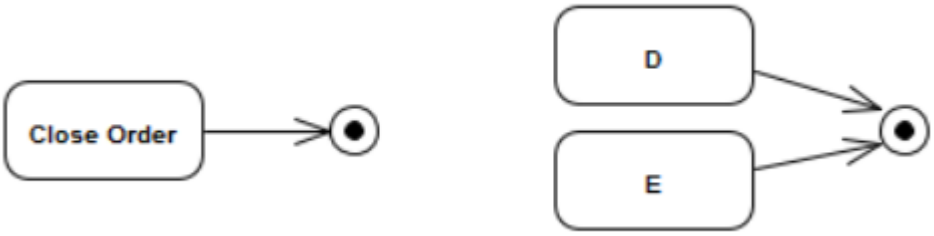


- It has outgoing edges but no incoming edges
- An activity may have more than one initial node

  Each generates a concurrent flow
- An initial node can have more than one outgoing edge

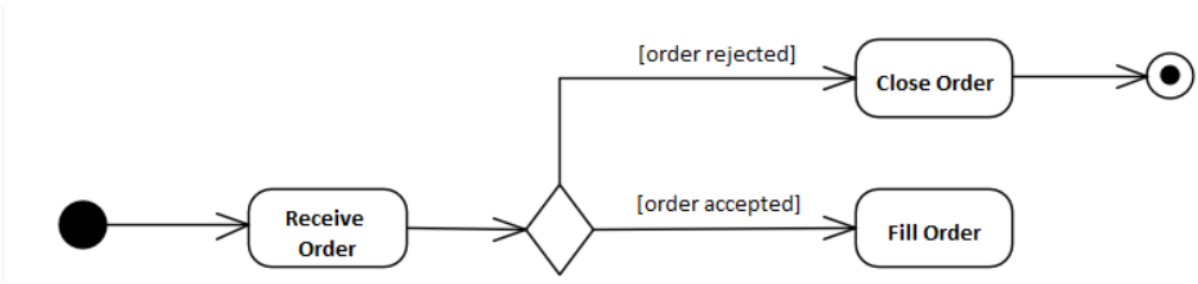
  Semantics: a control token at each outgoing edge
- Activity diagrams do not have to have initial nodes
- Notation: a filled circle

Final Node



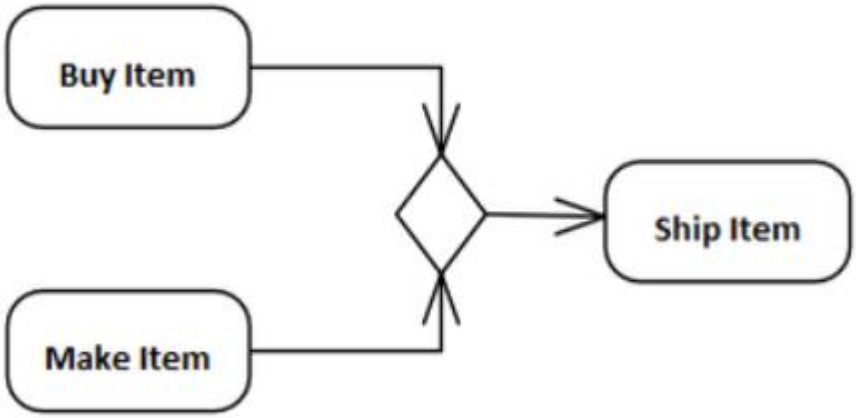
- An activity may have more than one activity final node
- The first one reached stops all flows in the activity
  - Regardless of the number of tokens in activity
- At least one incoming edge and no outgoing edges
- If several incoming edges only one must carry a token
- Activity diagrams do not have to have final nodes
- Notation: a filled circle with an outer ring

Decision



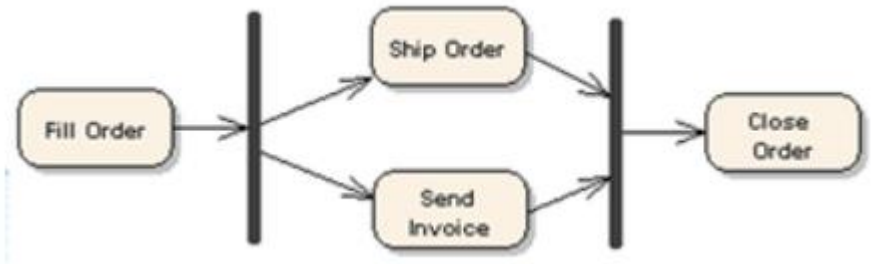
- is a control node that chooses between outgoing flows
- One incoming edge and several outgoing edges
- When a token is supplied guards on outgoing edges are evaluated
  - Token goes to first true outgoing
- Notation: a rhombus

Merge



- A merge node is a control node that brings together multiple alternate flows
  - It is not used to synchronize concurrent flows
- Several incoming edges, one outgoing edge
- Nothing calculated, nothing expected
- Notation: an empty rhombus

Fork, Join

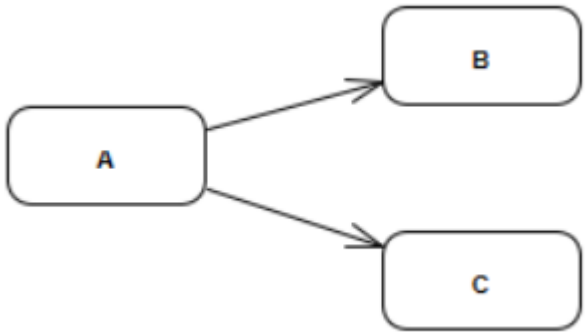


- **Fork** – one incoming edge and multiple outgoing edges
- **Join** – multiple incoming edges and one outgoing edge

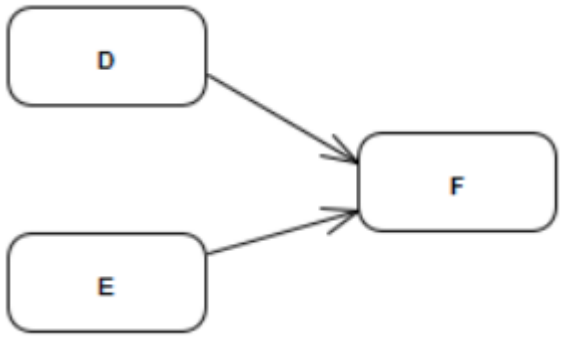
Implicit splitting

Once action A terminates

- a control token is available at both outgoing edges
- actions B and C start concurrently

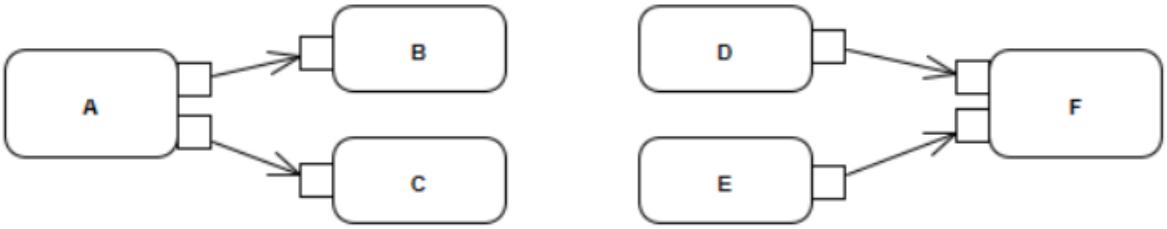


Implicit synchronization



Action F doesn't start until tokens are available at both incoming edges - actions D and E have to terminate

Implicit split/synchronization - object flow



AND semantics for object flows

- Once action A terminates it provides two object nodes
- Action F doesn't start until object tokens are available at both incoming edges - actions D and E have to terminate