# Modeling Flow-Based Behavior with Activities (Part 1 – SysML Concepts)

# Section Objectives

- In this Section, you will learn:
  - How to model Activity Diagrams in SysML

# Overview

- This section will discuss:
  - Activity Diagram Concepts
    - Why model Activities?
    - Activity Diagram Components
    - How to model Activities
  - Activity modeling for In-Class Project

# Acknowledgments

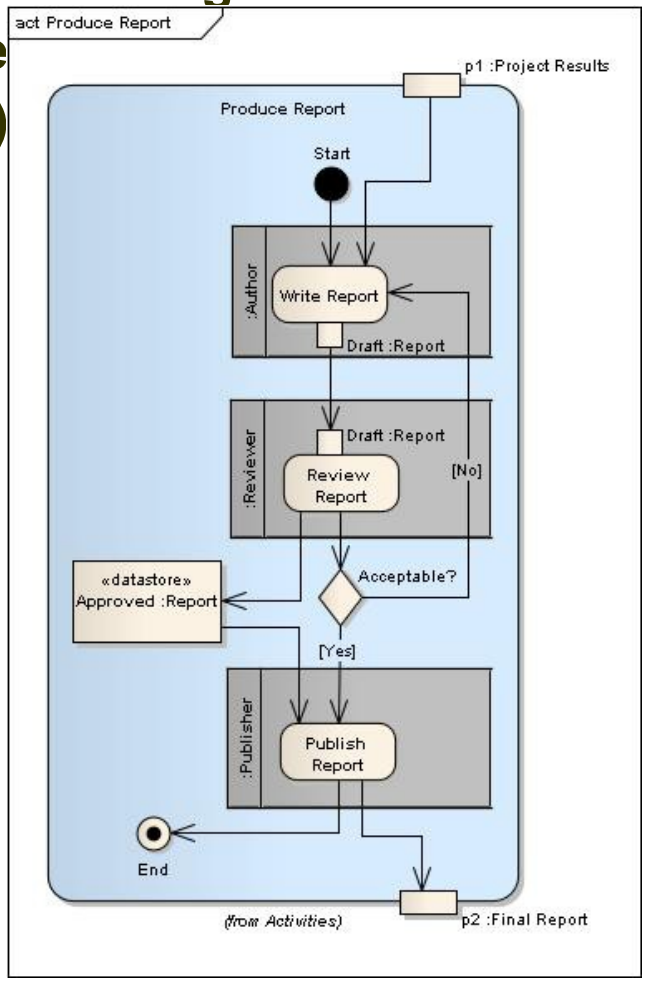
- Portions of this work are from the book, *A Practical Guide to SysML*, by Sanford Friedenthal, Alan Moore, and Rick Steiner, published by Morgan Kaufmann Publishers, Copyright 2009 Elsevier Inc. All rights reserved.
  - This section is based primarily on Chapter 8 of *A Practical Guide to SysML*

# Why Model Activities?

- Used to model behavior that specifies the transformation of inputs to outputs through a controlled sequence of actions.
- Used to elaborate Use Cases
- Graphical depiction of steps of a process
- Help to define functional requirements that system components or actors will perform
- Used to model workflows
- Clarification, Elaboration, Communication

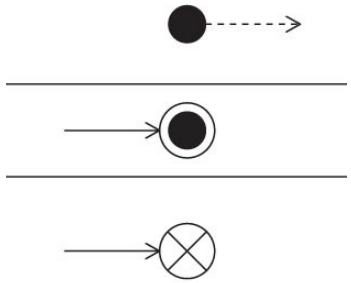
# Activity Diagram Components

- Activity diagrams can be comprised of the following:
  - Initial, Activity Final, and Flow Final Nodes
  - Actions (including Call Behavior Actions)
  - Input and Output Parameters
  - Object Flows
  - Central Buffer Nodes
  - Data Store Nodes
  - Fork Nodes and Join Nodes
  - Decision Nodes and Merge Nodes
  - Control Flow
  - Partitions (aka Swimlanes)



# Initial, Activity Final, and Flow Final Nodes

- Initial Node – denotes where execution begins
  - Depicted by black circle
- Activity Final Node – denotes where execution terminates
  - Depicted by a bulls-eye
- Flow Final Node – terminates a particular sequence of actions without terminating the entire activity
  - Depicted by circle with cross-hair

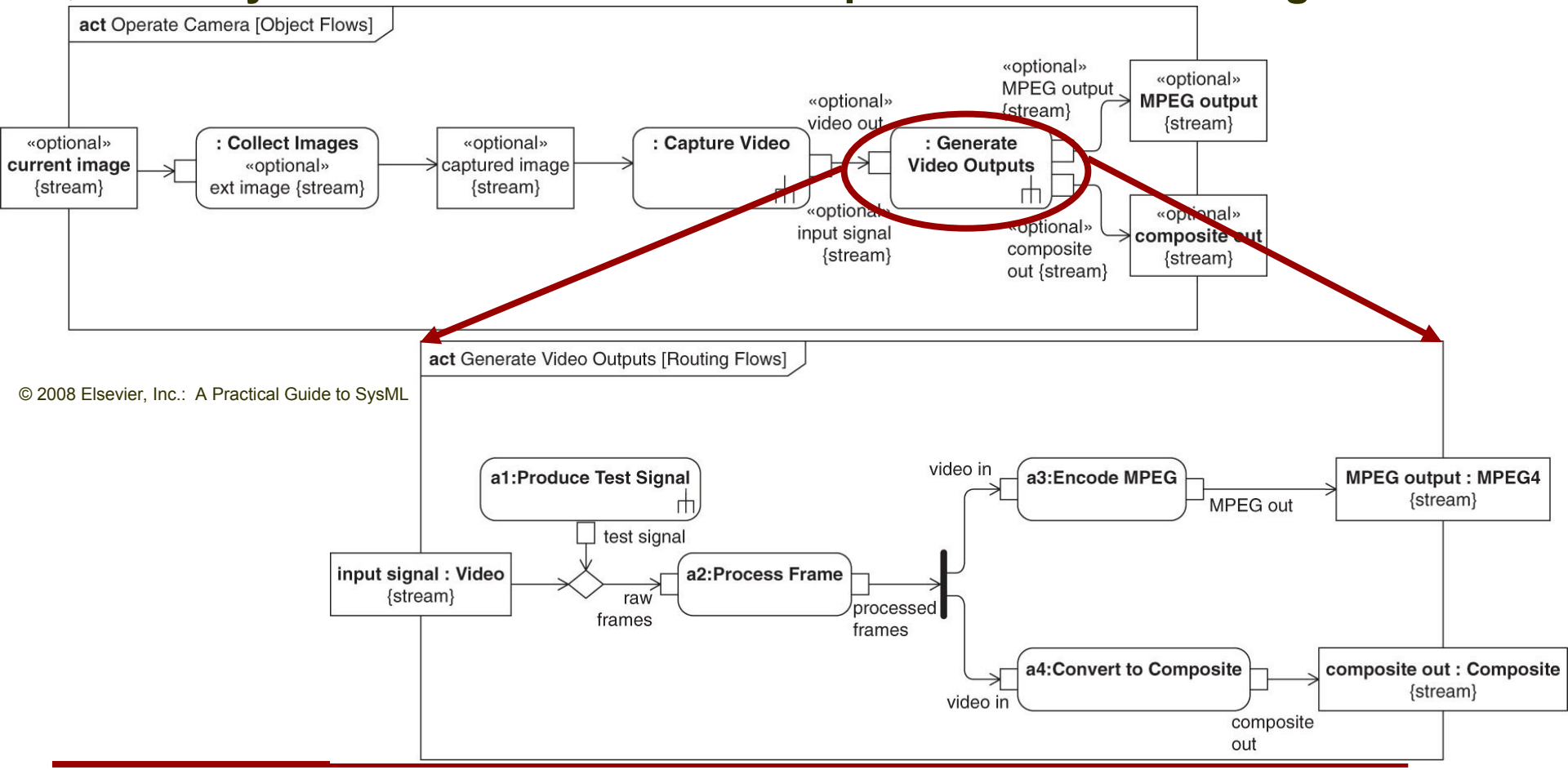


# Actions

- Actions – describe how activities execute
  - Used to model the steps of the activity
  - Accept inputs and create outputs (depicted by 'pins')
  - Call Actions – represent activities that can be further decomposed into other actions
    - Allows for hierarchical modeling of activities

# Call Behavior Actions

- Pins match Parameters in number and type
- Rake symbol denotes details are depicted on another diagram

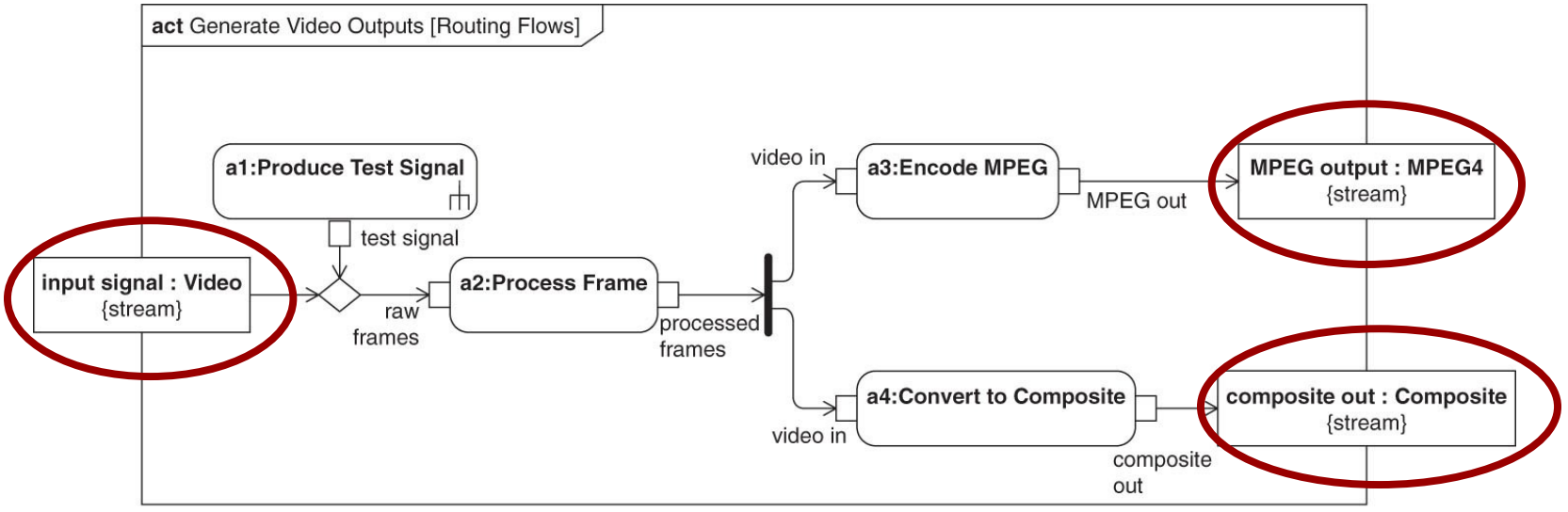


# Tokens

- Conceptual entity that can trigger the firing of actions
- Tokens are placed on the input pins of an action, processed by the action, and then placed on the output pins for other actions to accept
- Pins have multiplicity, that describe the minimum and maximum number of tokens that the action consumes or produces in any one execution
- Actions will begin execution when tokens are available on all of its required input pins and control inputs

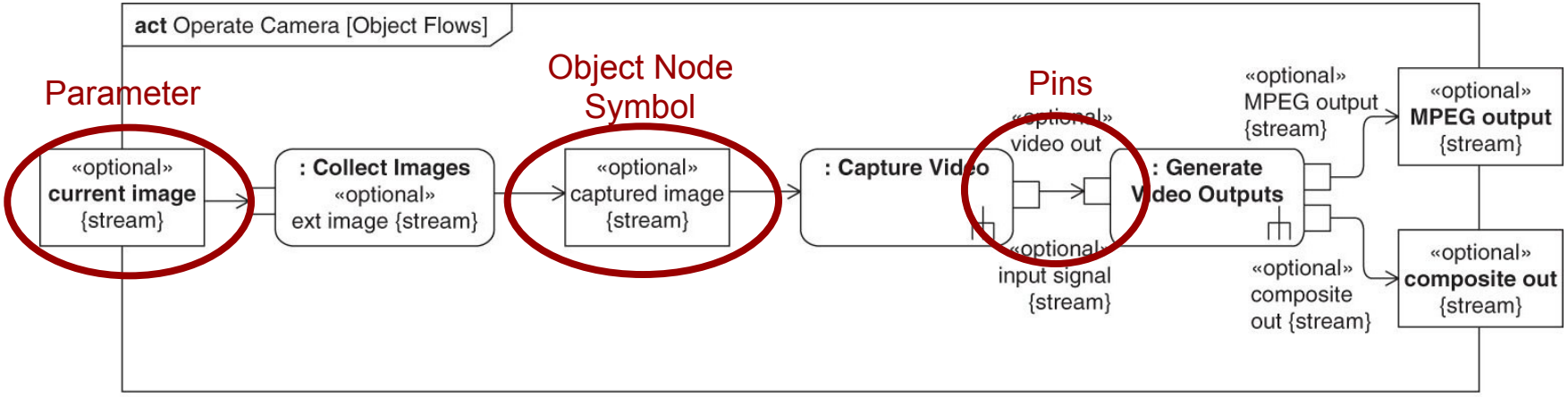
# Input and Output Parameters

- Activities can have multiple inputs and outputs called parameters
- Depicted as nodes on the frame of the activity diagram
- Parameters can be streaming or nonstreaming
  - Nonstreaming – only accept tokens when execution begins or ends
  - Streaming – can accept or produce tokens while executing



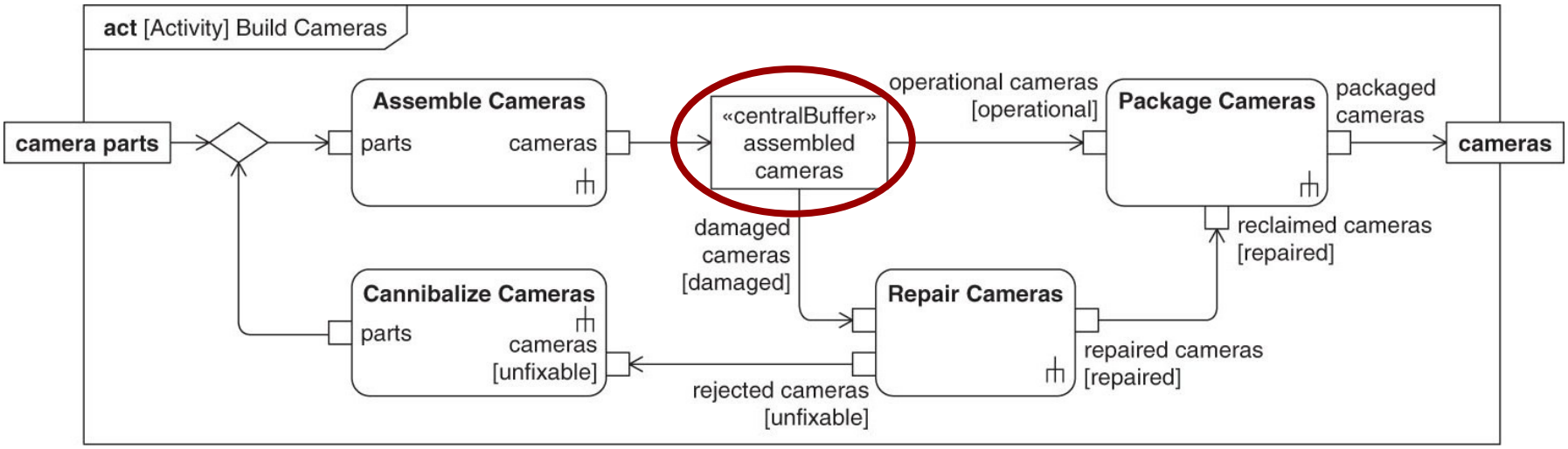
# Object Flows

- Used to route tokens from one object node to another
  - Object Node: Parameters and/or pins
- May represent information and/or physical items (e.g. water)
- Depicted by a solid arrow between the source and destination
- Object Node Symbols can be used in lieu of Pins



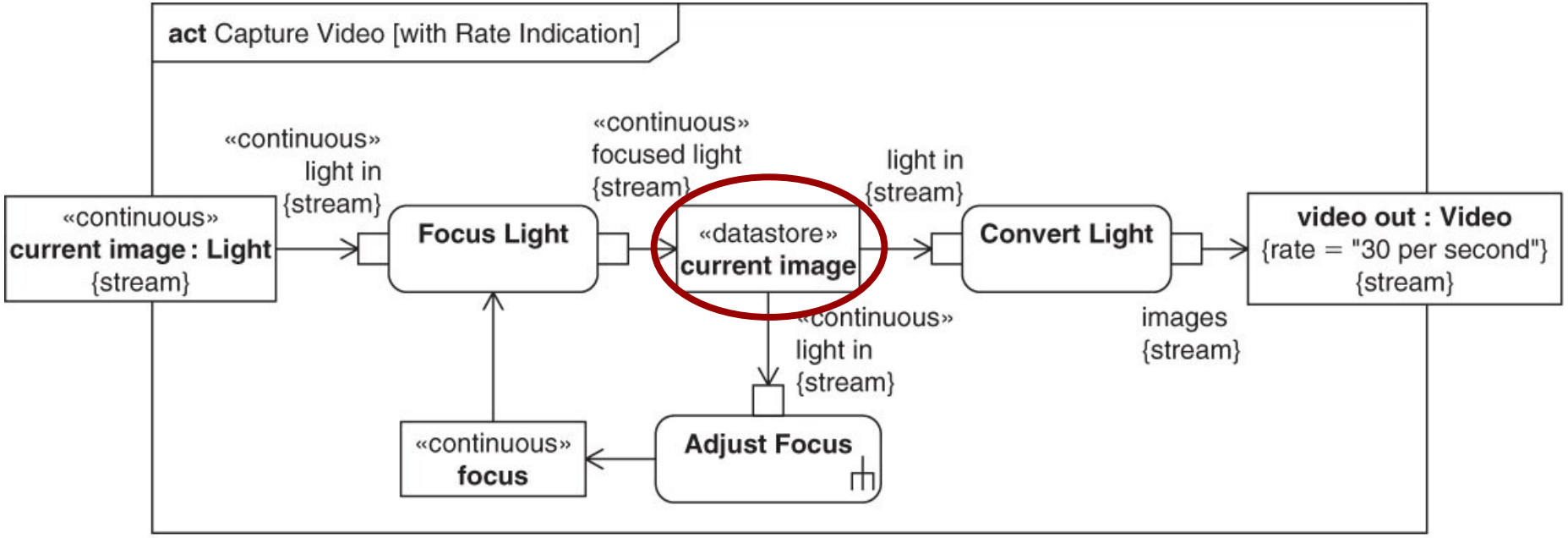
# Central Buffer Nodes

- Central Buffer Nodes
  - Used as a place to store objects
  - Needed, for example, when there are multiple consumers for a single stream of objects (see below)



# Data Store Nodes

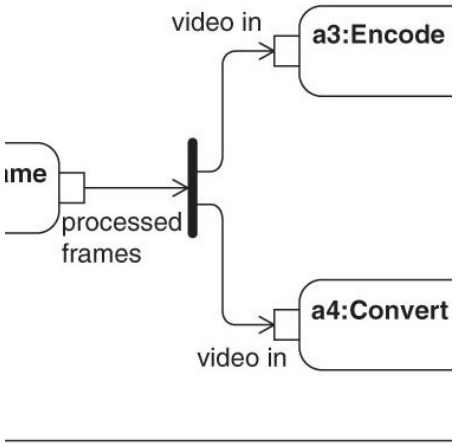
- Data Store Nodes
  - Similar to a Central Buffer, but provides a copy of the object instead of the original



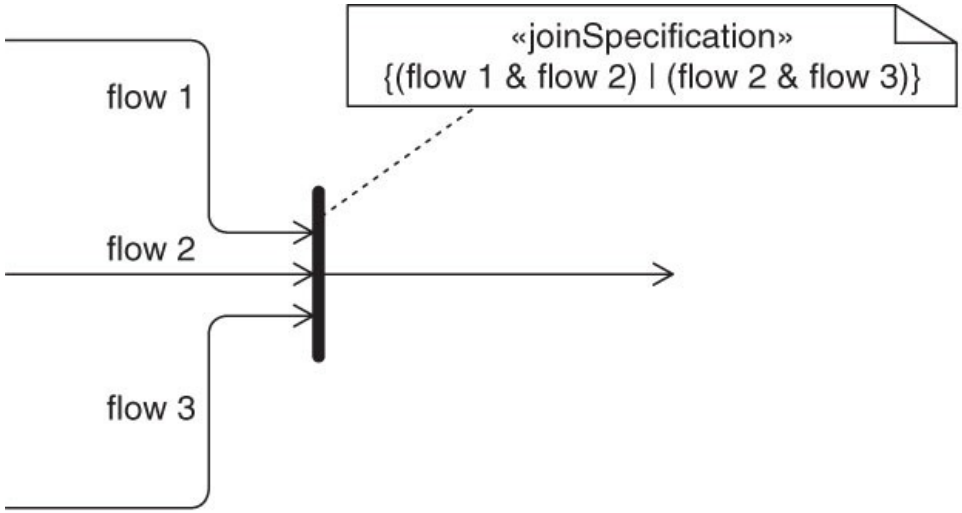
# Fork Nodes and Join Nodes

- Fork Node – one input flow, multiple output flows
  - Output flows are independent and concurrent
- Join Node – multiple input flows, one output flow
  - Output occurs, only when all input tokens are available (default)
- Join Specification may override default

Fork Node



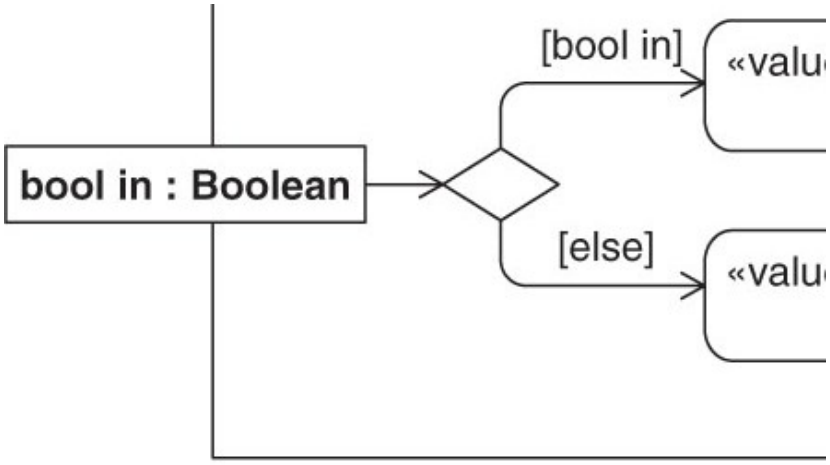

Join Node with Join Specification



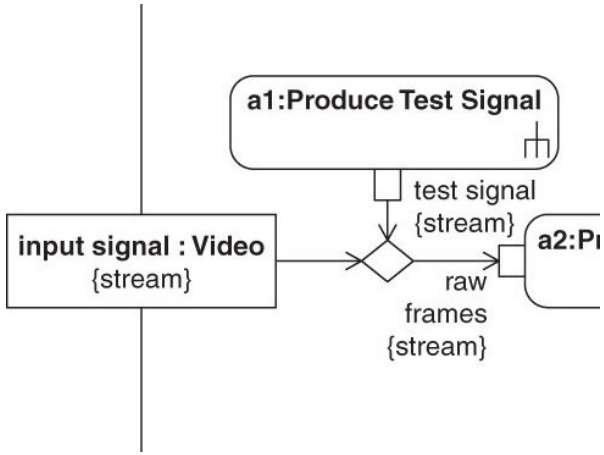
# Decision Nodes and Merge Nodes

- Decision Nodes – one input, multiple output paths
  - Only one output path is valid, based on 'guard' conditions
  - Guards must be mutually exclusive
- Merge Node – multiple inputs, one output flow
  - Output flow is triggered upon arrival of any of the input flows

Decision Node

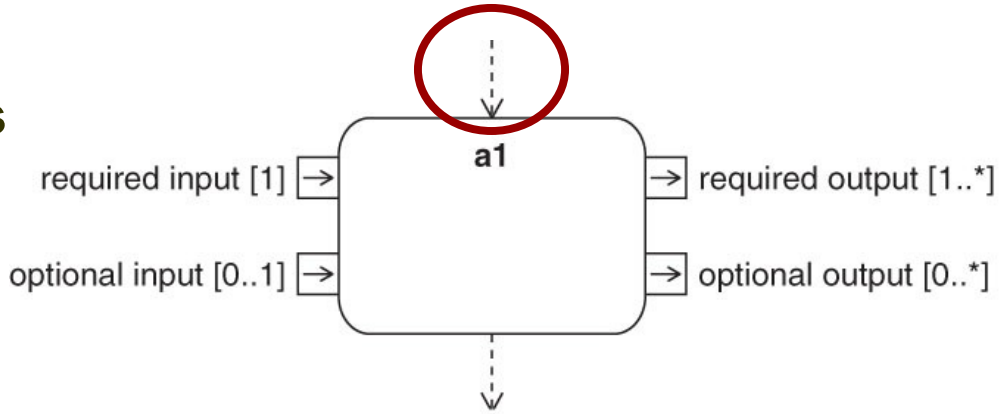



Merge Node



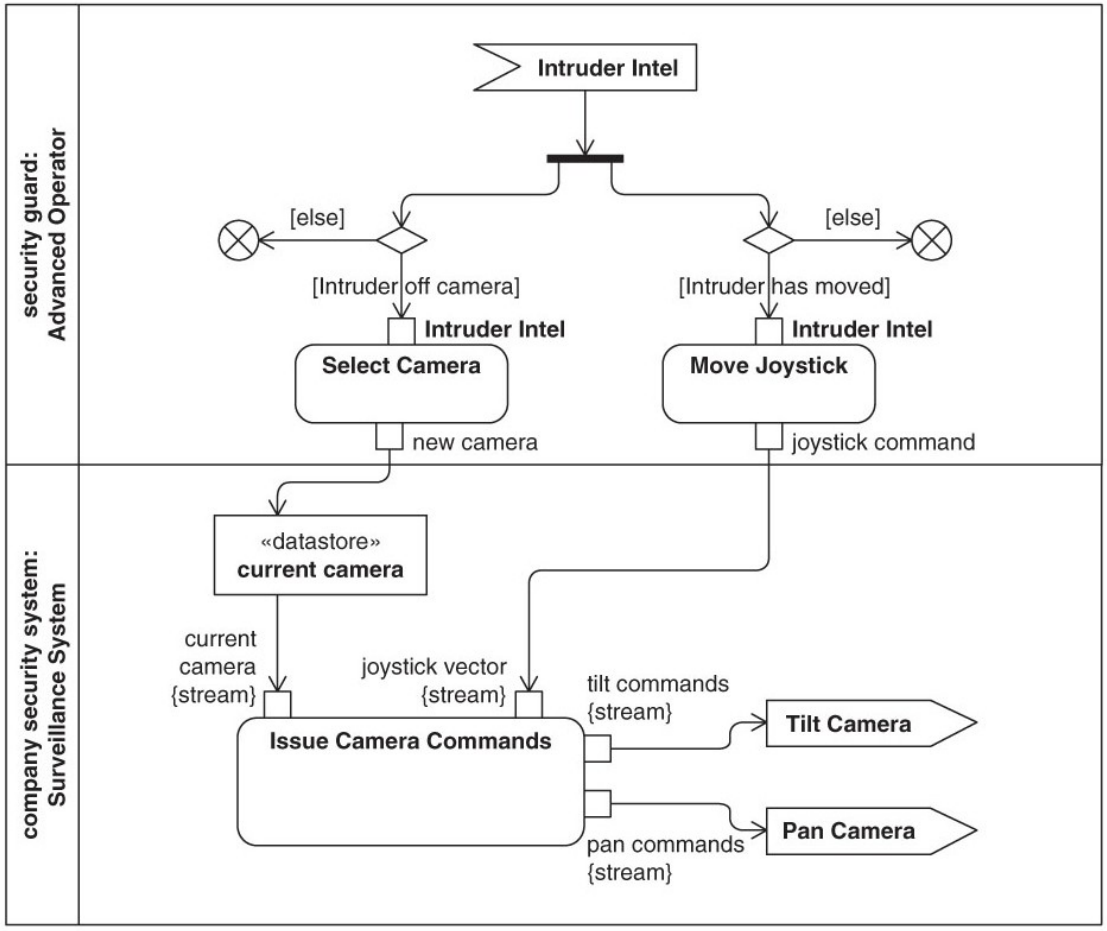
# Control Flow

- Used to show sequence of actions
- Represents a control token
  - An action cannot start until it receives a control token on all input control flows
  - When an action is completed, it places control tokens on all outgoing control flows
- Can be depicted with a dashed arrow, to distinguish it from object flows
- Like object flow, can be used with:
  - Forks and Joins
  - Decision Nodes and Merges



# Partitions (aka Swimlanes)

- Allocates actions to an entity responsible for performing the action
- Can be used to specify functional requirements of an actor, component, or part
- Can be depicted horizontally or vertically



# How to Model Activities

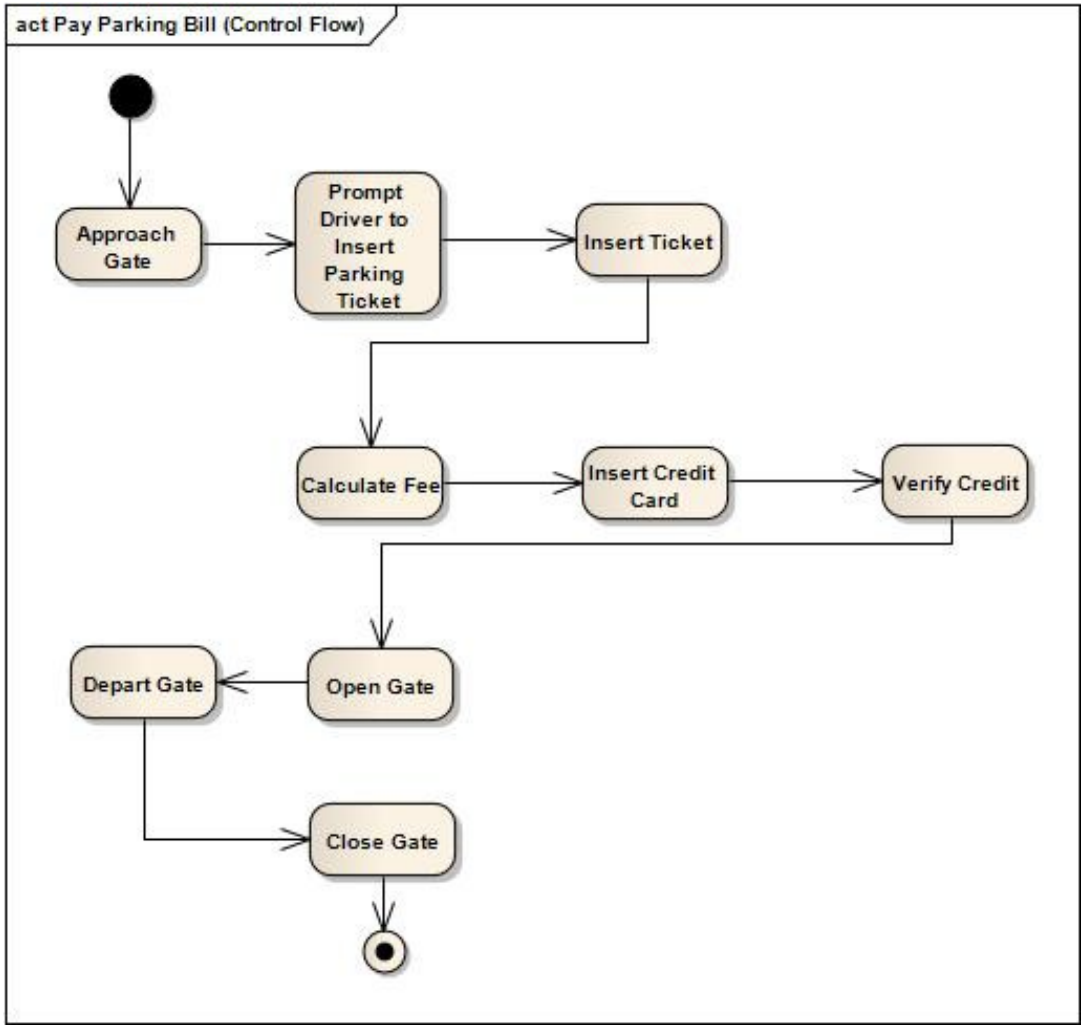
- Decide what to model – Use Cases
- Model the Primary Path
- Model the Alternate and Exception Paths
- Identify Objects that flow between Actions
- Decompose High-Level Activities
- Add Partitions

- Questions to ask along the way:
  - What are the steps in the process?
  - What is the primary path through the activities?
  - What are the alternate and exception paths?
  - What 'flows' between steps?
  - What are the input and output parameters of each activity?
  - Can an activity be decomposed into distinct actions?
  - Who or what performs each step?

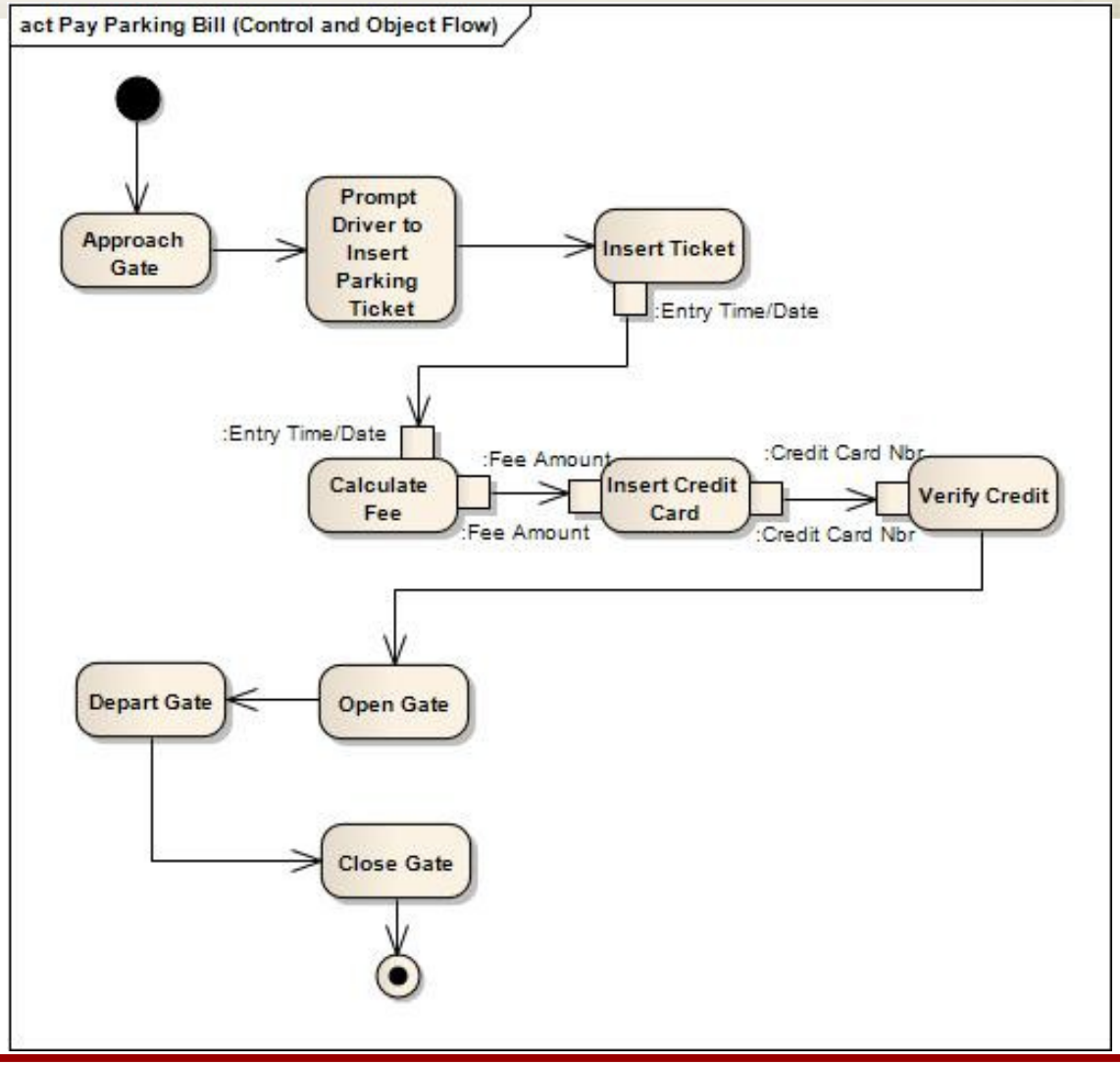
# Activity Modeling for In-Class Project

- Build Activity Model for Parking Garage Gate Project using EA
- Define
  - Actions
  - Primary Path
  - Control Flow
  - Object Flow
  - Partitions

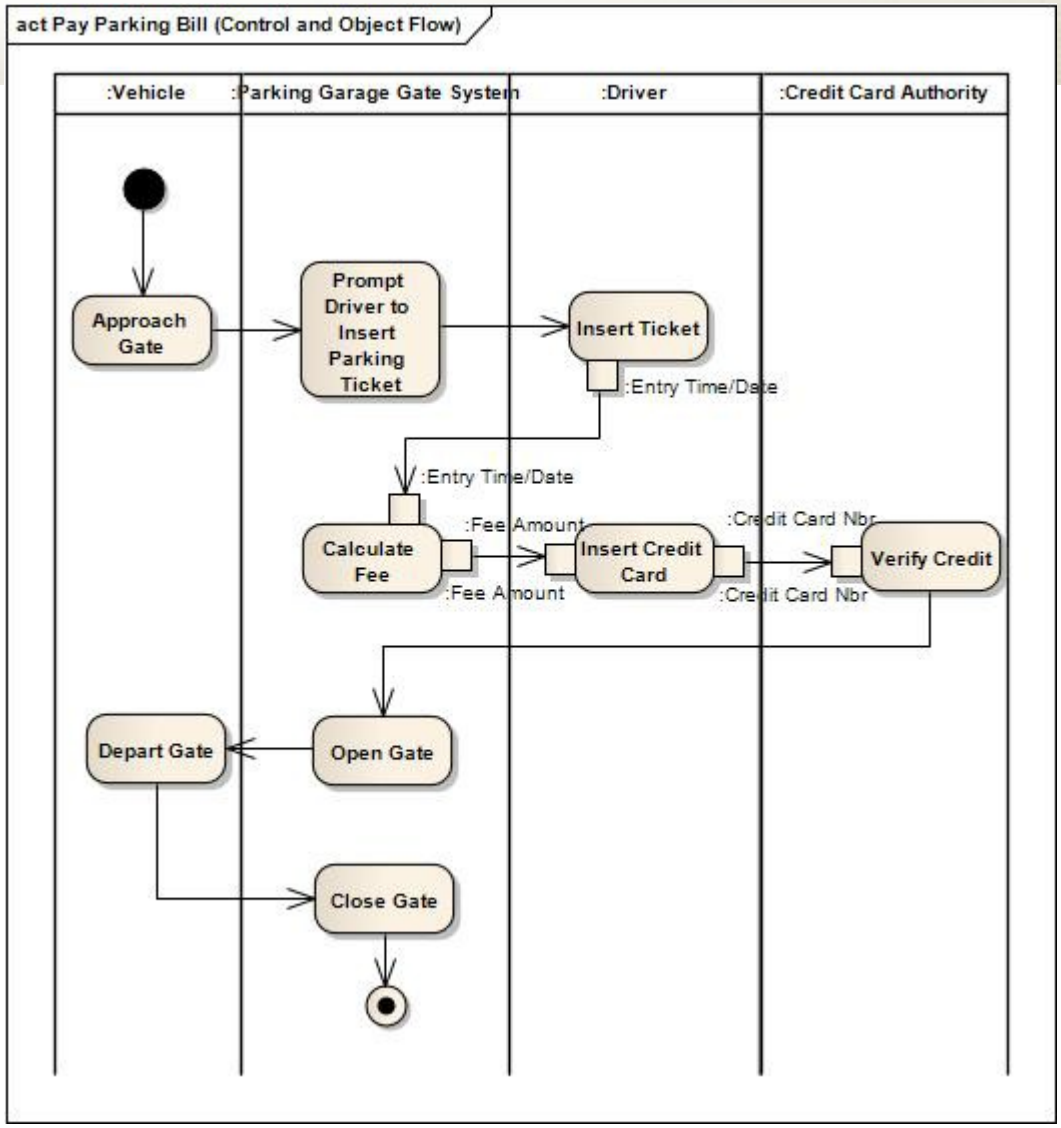
# Activity Model (Primary Path)



# Activity Model (w/Object Flow)

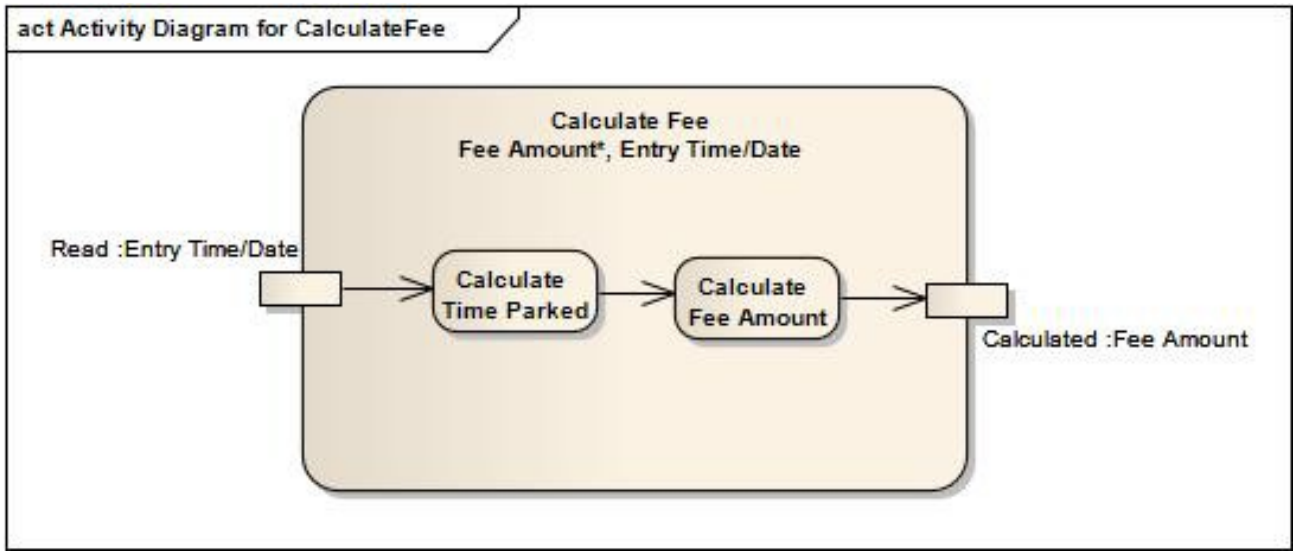


# Activity Model (w/Partitions)



# Decomposition of Calculate Fee

- Example below shows use of Input and Output Parameters for the Calculate Fee Activity
- Hierarchical relationship of Activities and Actions



# Further Analysis Needed

- Model contains primary path
- What's Missing?
  - Alternate Flows
    - Pre-paid Ticket
  - Exception Flows
    - Lost or Damaged Ticket
    - Invalid Credit Card
  - Missing Action
    - Return Credit Card to Driver
  - Candidate Actions for De-composition (Call Behavior Actions)
    - Verify Credit
- Modeling helps uncover areas that need further analysis

# Summary

- Activity Diagrams are used to model behavior that specifies the transformation of inputs to outputs through a controlled sequence of Actions
- Activities are made up of Actions that represent the lowest level of behavior
- Call Actions are used to depict activities that can be further decomposed into other actions
- Activities can have multiple inputs or outputs called parameters
- Parameters can either be streaming or nonstreaming
- Actions consume input tokens and produce output tokens via pins
- Object Flows are used to depict the flow of object tokens from one action to other actions
- Control Flows are used to depict the transfer of control from one action to other actions using control tokens
- Partitions are used to assign responsibility for actions to blocks or parts that the partition represent