UML Activity Diagrams

Idaho State University | Computer Science

Isaac Griffith

CS 3321
Department of Computer Science
Idaho State University

Outline

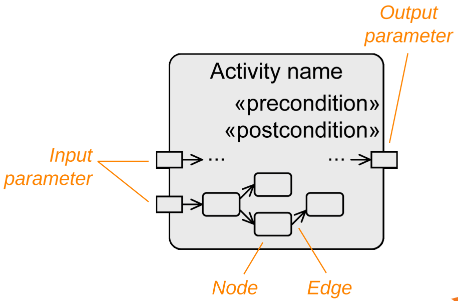
- UML Activity Diagrams

Introduction

- Focus of activity diagram: **procedural processing aspects**
- Flow-oriented language concepts
- Based on
  - languages for defining business processes
  - established concepts for describing concurrent communicating processes (token concept as found in petri nets)
- Concepts and notation variants cover **broad area of applications**
  - Modeling of object-oriented and non-object-oriented systems

Activity

- Specification of user-defined behavior at different levels of granularity
- Examples:
  - Definition of the behavior of an operation in the form of individual instructions
  - Modeling the course of actions of a use case
  - Modeling the functions of a business process
- An activity is a directed graph
  - Nodes: actions and activities
  - Edges: for control and object flow
- Control flow and object flow define the execution
- Optional:
  - Parameter
  - Pre- and postconditions



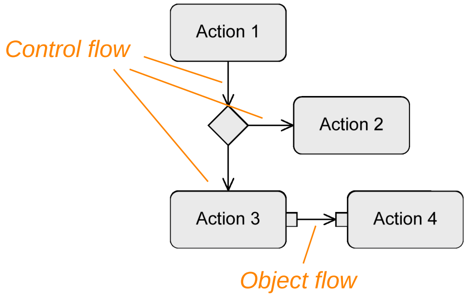
Action

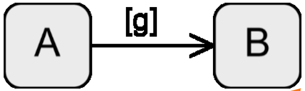
- **Basic element** to specify user-defined behavior
- **Atomic** but can be aborted
- No specific rules for the description of an action
  - Definition in natural language or in any programming language
- Process input values to produce output values
- Special notation for predefined types of actions, most importantly
  - Event-based actions
  - Call behavior actions

Edges

- Connect activities and actions to one another
- Express the execution order

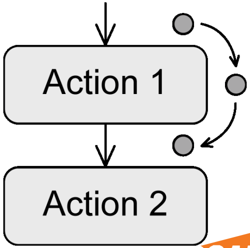
- Types
  - Control flow edges
    - Define the order between nodes
  - Object flow edges
    - Used to exchange data or objects
    - Express a data/causal dependency between nodes
- Guard (condition)
  - Control and object flow only continue if guards in square brackets evaluate to true.





Token

- **Virtual coordination mechanism** that describes the execution exactly
  - No physical component of the diagram
  - Mechanism that grants the execution permission to actions
- If an action receives a token, the action can be executed
- When the action has completed, it passes the token to a subsequent action and the execution of this action is triggered
- Guards can prevent the passing of a token
  - Tokens are stored in previous node
- Control token and object token
  - **Control token:** "execution permission" for a node
  - **Object token:** transport data + "execution permission"



Activity start and end

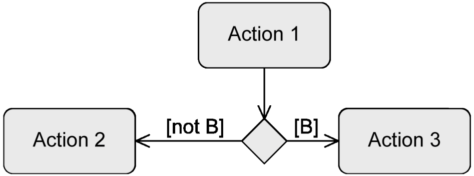
- Initial Node ●
  - Starts the execution of an activity
  - Provides tokens at all outgoing edges
  - Keeps tokens until the successive nodes accept them
  - Multiple initial nodes to model concurrency
- Activity final node ◉
  - Ends all flows of an activity
  - First token that reaches the activity final node terminates the entire activity
    - Concurrent subpaths included
  - Other control and object tokens are deleted
    - Exception: object tokens that are already present at the output parameters of the activity

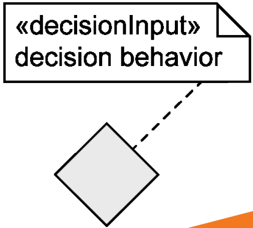
Flow end

- Flow final node ⊗
  - Ends one execution path of an activity
  - All other tokens of the activity remain unaffected

Alternative Paths - Decision Node

- To define alternative branches
- “Switch point” for tokens
- Outgoing edges have guards
  - Syntax: [Boolean expression]
  - Token takes **one** branch
  - Guards must be mutually exclusive
  - Predefined: [else]
- Decision behavior
  - Specify behavior that is necessary for the evaluation of the guards
  - Execution must not have side effects

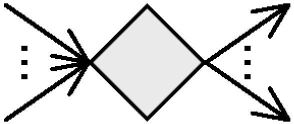




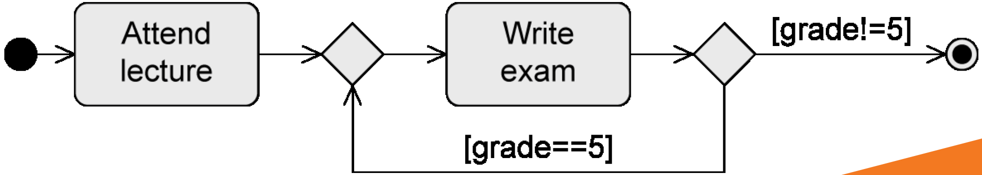
Idaho State University Computer Science

Alternative Paths - Merge Node

- To bring **alternative** subpaths together
- Passes token to the next node
- Combined decision and merge node

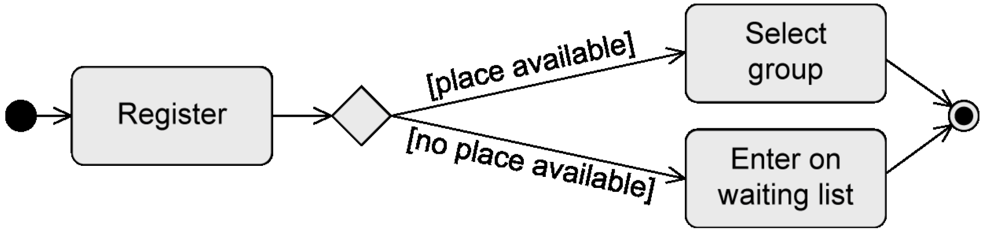


- Decision and merge nodes can also be used to model loops:



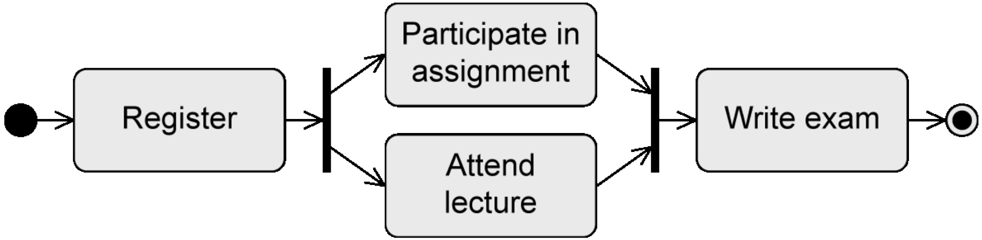
ROAR

Example: Alternative Paths



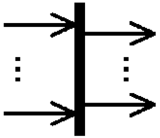
Concurrent Paths - Parellization

- To split path into concurrent subpaths
- Duplicates token for all outgoing edges
- Example:

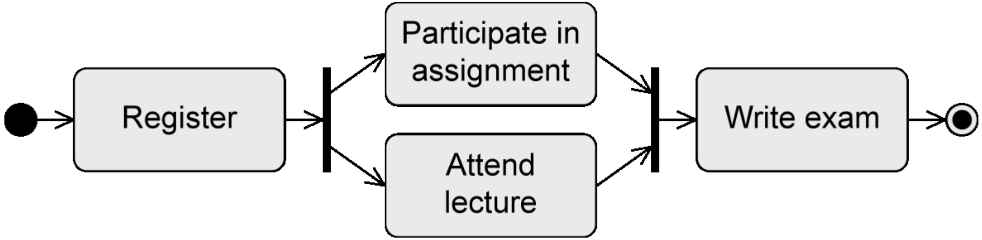


Concurrent Paths - Synchronization

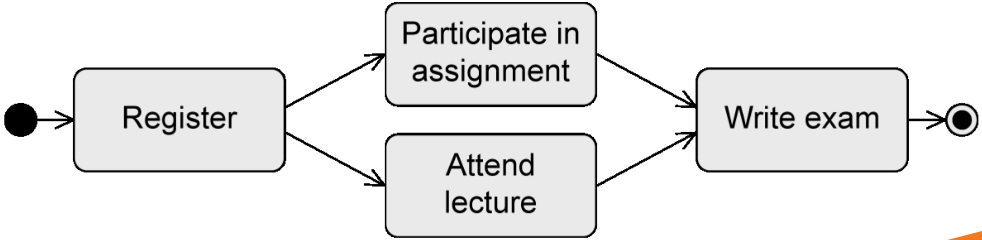
- To merge concurrent subpaths
- Token processing
  - Waits until tokens are present at all incoming edges
  - Merges all control tokens into one token and passes it one
  - Passes on all object tokens
- Combined parallelization and synchronization node:



Example: Equivalent Control Flow

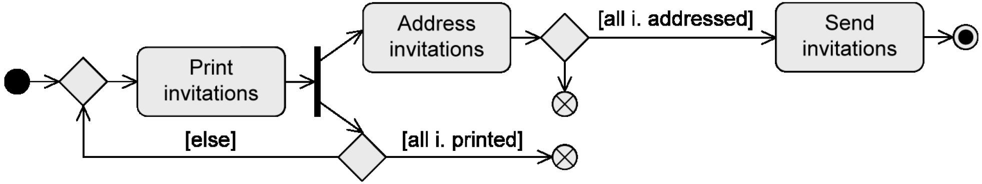


… equivalent to …

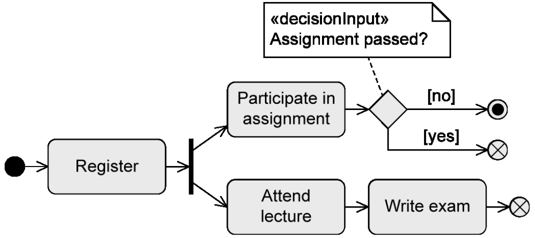


Example: Create and Send Invitations

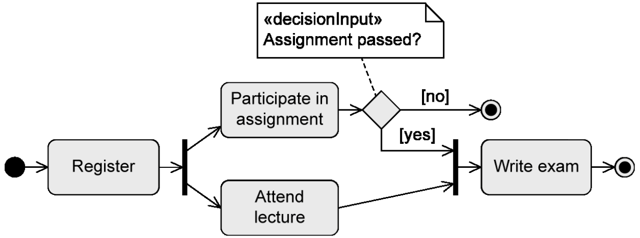
- While invitations are printed, already printed invitations are addressed.
- When all invitations are addressed, then the invitations are sent.



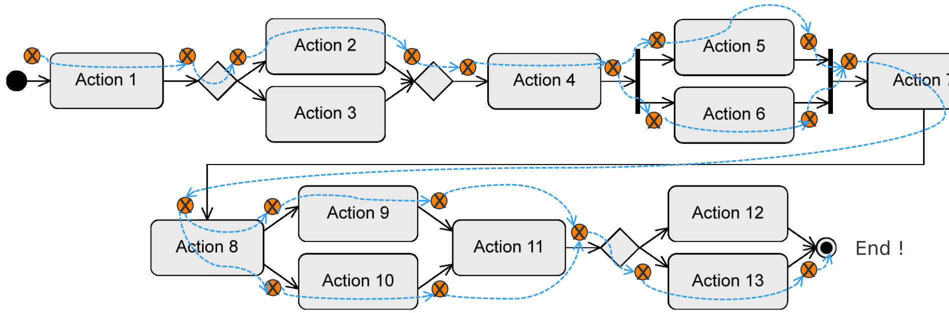
Example: Conduct Lecture



NOT equivalent … why?



Example: Token (Control Flow)



… all outgoing edges of all initial nodes are assigned a token…

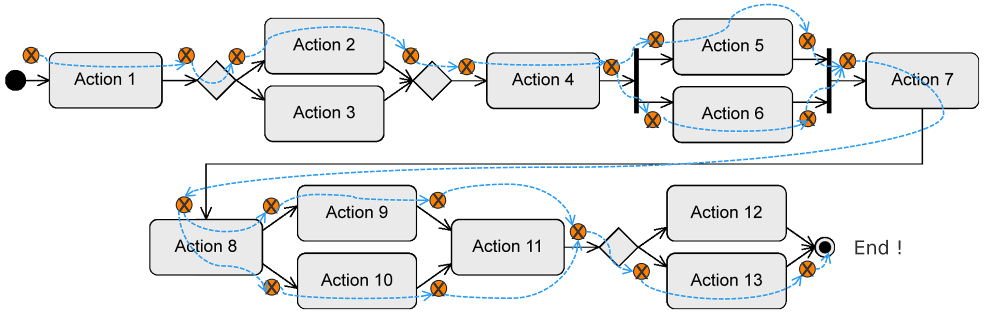
… if all incoming edges of an action have a token, the action is activated and is ready for execution

… before the execution, the action consumes one token from every incoming edge; after the execution, the action passes one token to every outgoing edge

… a decision node passes the token to one outgoing edge (depending on the result of the evaluation of the guard)

… a merge node individually passes each token it gets to its outgoing edge

Example: Token (Control Flow)



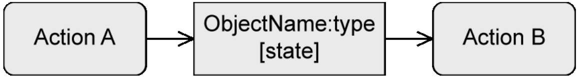
... a parallelization node duplicates an incoming token for all outgoing edges

... a synchronization node waits until all incoming edges have a token, merges them to a single token and passes it to its outgoing edge

... the first token that reaches the activity final node terminates the entire activity

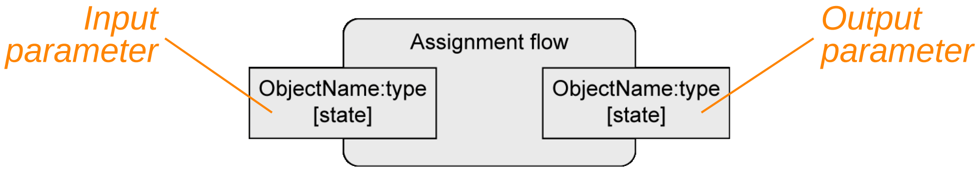
Object Node

- Contains object tokens
- Represents the exchange of data/objects
- Is the source and target of an object flow edge
- Optional information: type, state

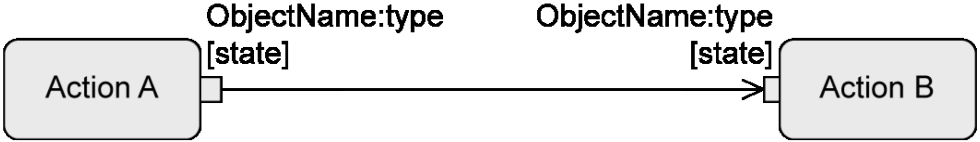


Object Node

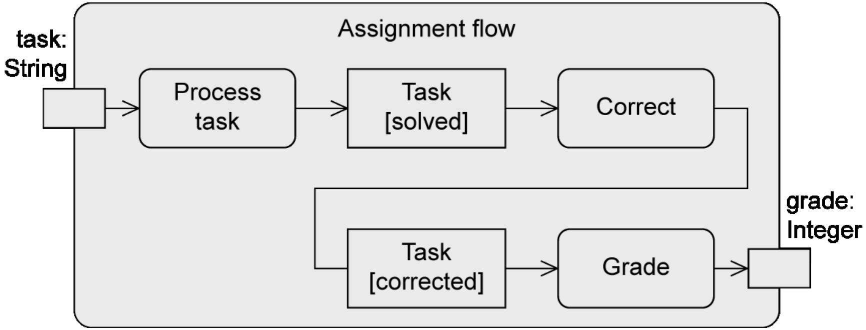
- Notation variant: object node as parameter
  - For activities

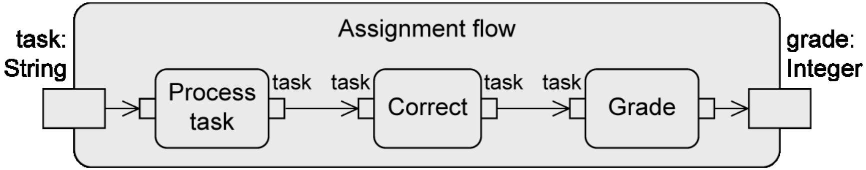


  - For actions



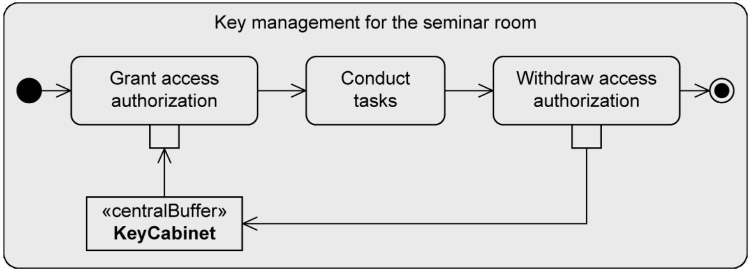
Example: Object Node





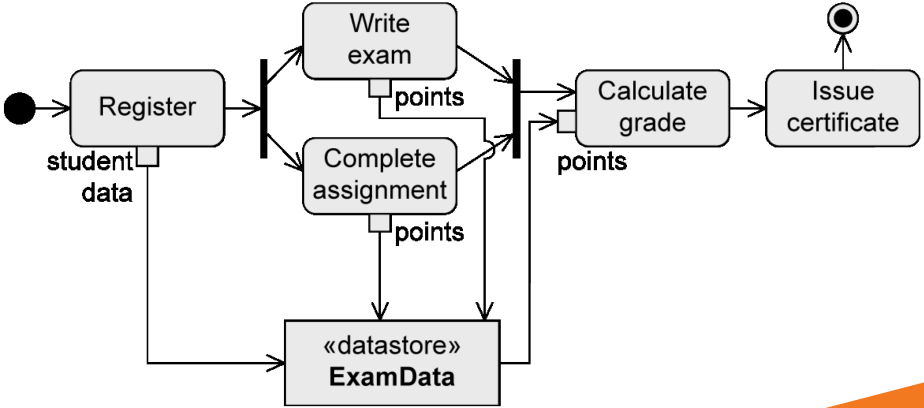
Central Buffer

- 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


Key management for the seminar room
Grant access authorization
Conduct tasks
Withdraw access authorization
«centralBuffer»
KeyCabinet

Data Store

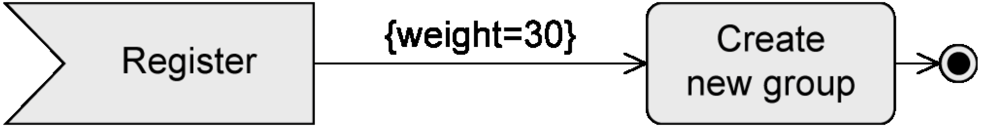
- For saving and passing on object tokens
- Permanent memory
- Saves object tokens permanently, passes copies to other nodes



Idaho State University
Computer Science

Weight of Edges

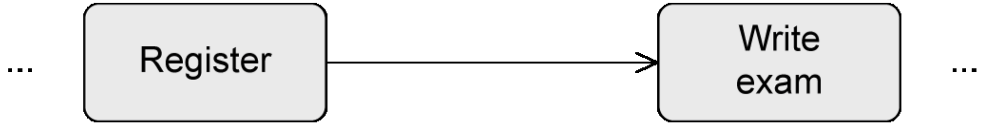
- Minimal number of tokens that must be present for an action to be executed
- Default: `1`
- All tokens present have to be consumed: `0` (also `all` or `*`)



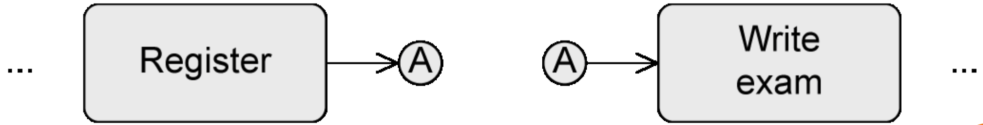
ROAR

Connector

- Used if two consecutive actions are far apart in the diagram
- Without connector:

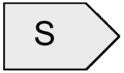


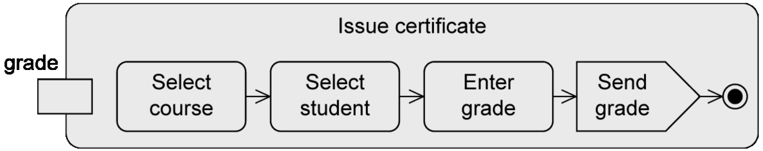
- With connector:



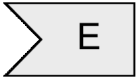
Event-Based Actions

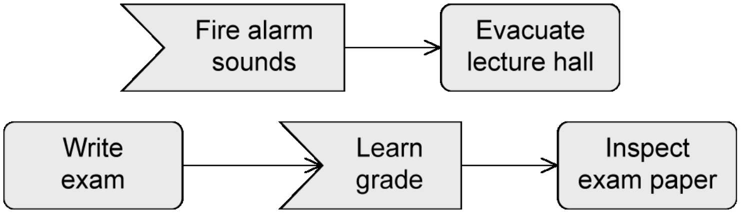
- To send signals
  - Send signal action



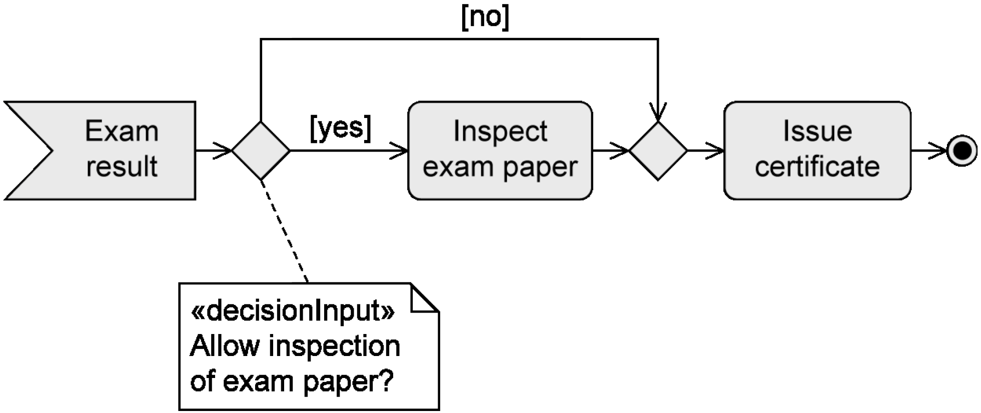


- To accept events
  - Accept event action



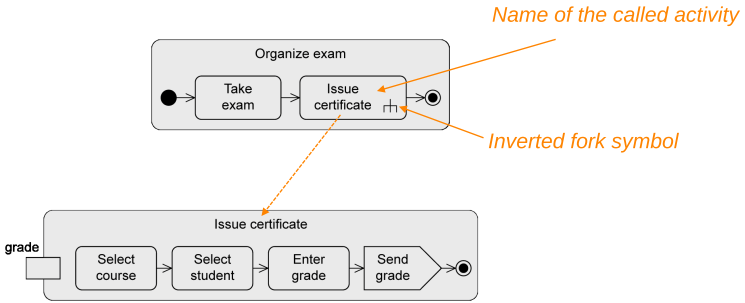


Example: Accept Event Action



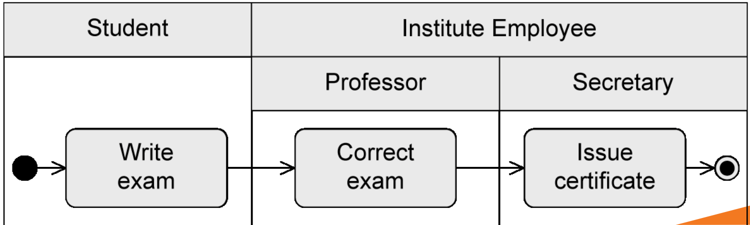
Call Behavior Action

- The execution of an action can call an activity
- Content of the called activity can be modeled elsewhere
- Advantages:
  - Model becomes clearer
  - Reusability

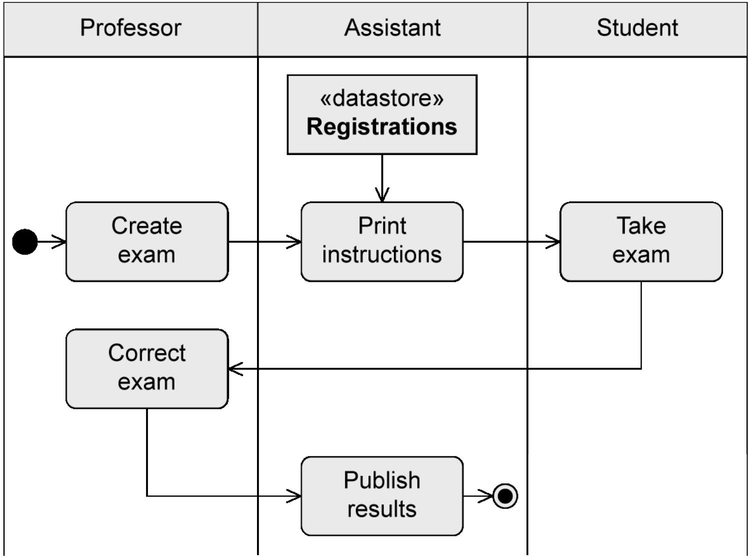


Partition

- “Swimlane”
- Graphically or textual
- Allows the grouping of nodes and edges of an activity due to responsibilities
- Responsibilities reflect organizational units or roles
- Makes the diagram more structured
- Does not change the execution semantics
- Example: partitions `Student` and `Institute Employee` (with subpartitions `Professor` and `Secretary`)

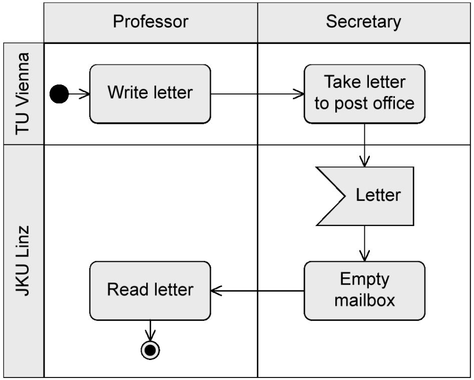


Example: Partitions

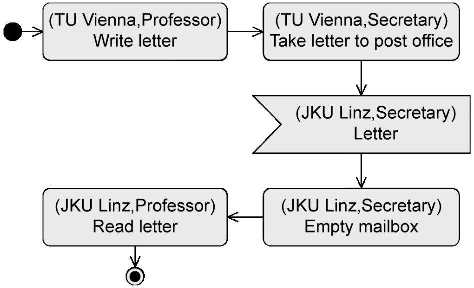


Multidimensional Partitions

- Graphical notation



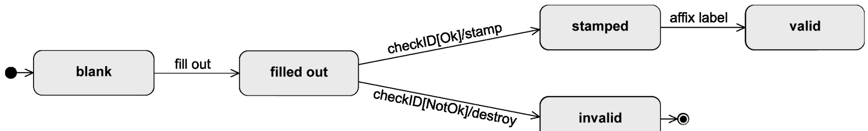
… or alternatively textual notation



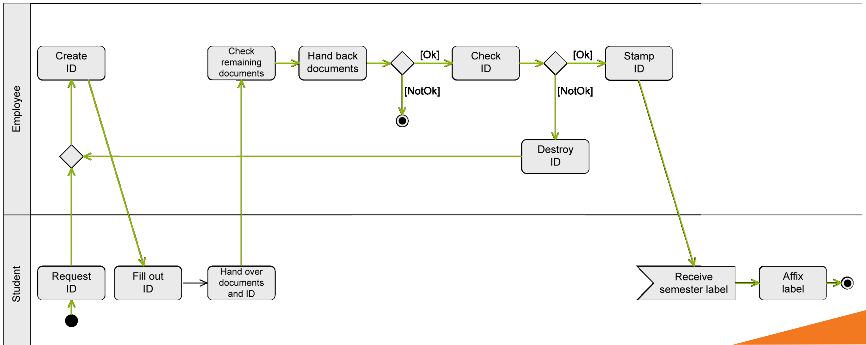
(TU Vienna,Professor) Write letter
(TU Vienna,Secretary) Take letter to post office
(JKU Linz,Secretary) Letter
(JKU Linz,Secretary) Empty mailbox
(JKU Linz,Professor) Read letter

Example

- State machine diagram of Student ID:

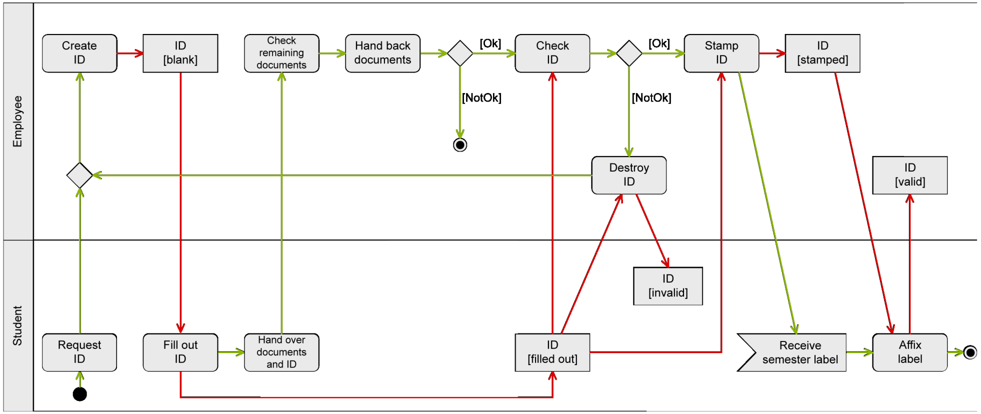


- Activity diagram – control flow:



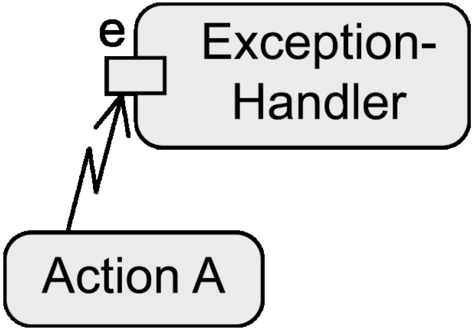
Example

- Control flow (green) and object flow (red) in one activity diagram



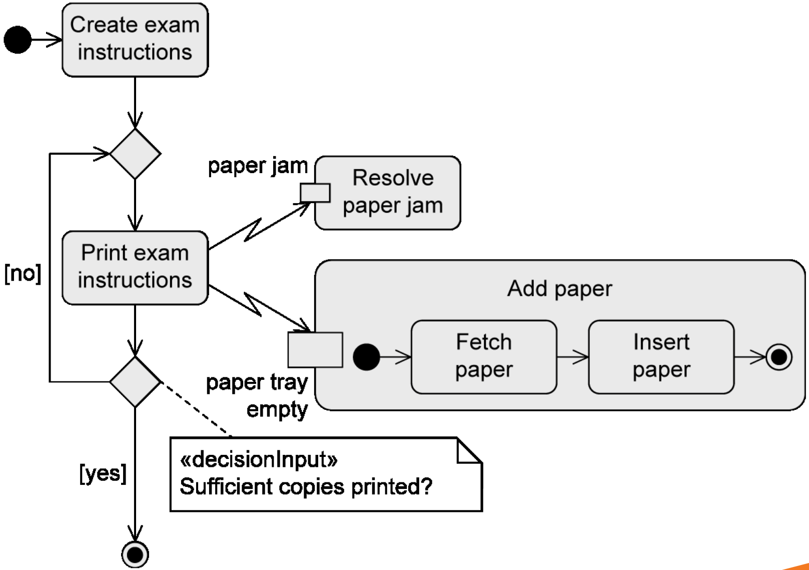
Exception Handling

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



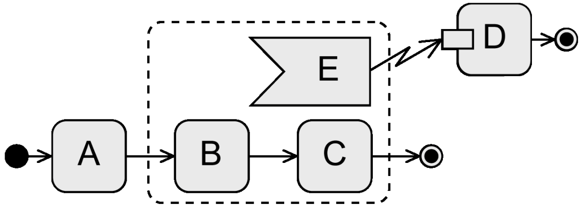
- If the error `e` occurs...
  - All tokens in `Action A` are deleted
  - The exception handler is activated
  - The exception handler is executed instead of `Action A`
  - Execution then continues regularly

Example: Exception Handler



Interruptible Activity Region

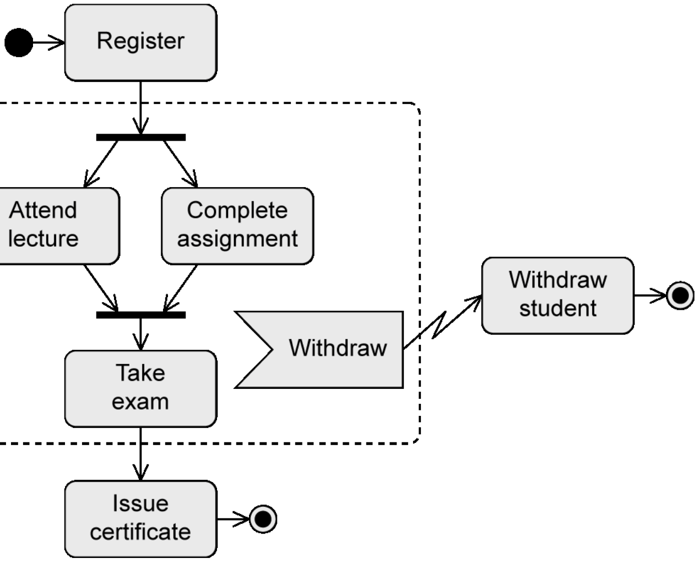
- Defining a group of actions whose execution is to be terminated immediately if a specific event occurs. In that case, some other behavior is executed



- If `E` 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
- No "jumping back" to the regular execution!

Idaho State University Computer Science

Example: Interruptible Activity Region

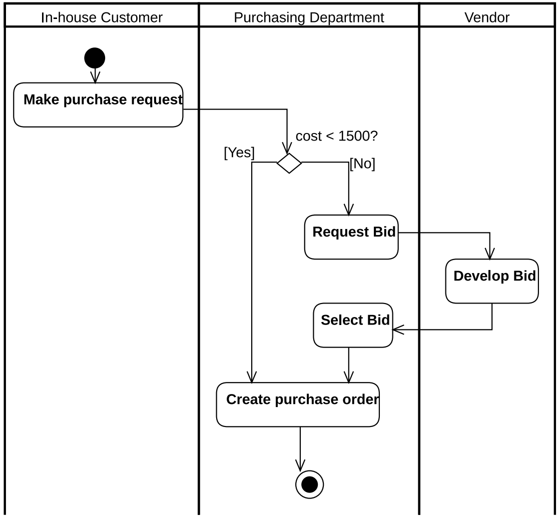


ROAR

Activity Diagram Exercise

The purchasing department handles purchase requests from other departments in the company. People in the company who initiate the original purchase request are the "customers" of the purchasing department. A case worker within the purchasing department receives that request and monitors it until it is ordered and received. Case workers process the requests for purchasing products under $1,500, write a purchase order, and then send it to the approved vendor. Purchase requests over $1,500 must first be sent out for a bid from the vendor that supplies the product. When the bids return, the case worker selects one bid. Then, the case worker writes a purchase order and sends it to the approved vendor.

Activity Diagram Exercise



Are there any questions?