COMP1023 Software Engineering

Spring Semester 2019-2020

Dr. Radu Muschevici

School of Computer Science, University of Nottingham, Malaysia

Lecture 6

2020-03-11

Topics covered

So far we learnt about **UML Diagrams** for modelling…

System Behaviour (specifically: Interactions)

- Use case diagrams (& use case bodies)
- Sequence diagrams

Topics covered

Today we will look at modelling…

System Behaviour

- Use case diagrams (& use case bodies)
- Sequence diagrams
- Activity diagrams

Topics covered

Next week we will look at modelling…

System Behaviour

- Use case diagrams (& use case bodies)
- Sequence diagrams
- Activity diagrams

System Structure

- Class diagrams

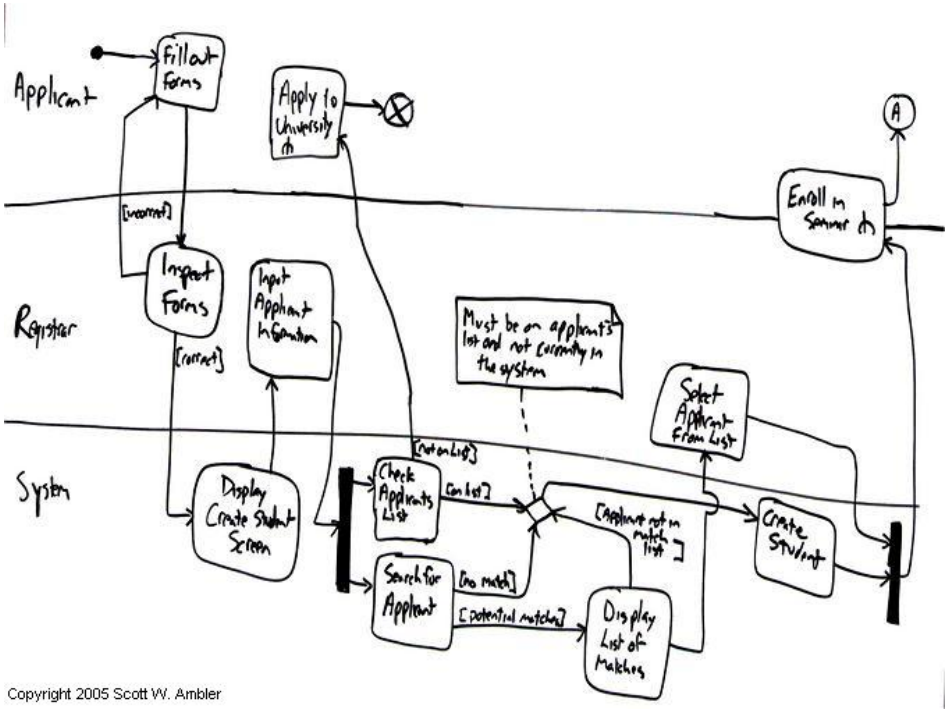
Activity Diagrams

- Capture the **dynamic behavior** of a system.
- Model the **flow of control**, i.e., the order in which actions are executed.
- Emphasise the **sequence** and **conditions** of the flow for **coordinating** activities.
- Show **sequential** and **parallel** activities in a process.

Activity Diagrams

Useful for modelling:

- Business processes
- Workflows
- Data Flows
- Complex algorithms
- Steps in a use case diagram (alternative to tabular use case body)
- Understanding of workflow across many use cases



Activity Diagrams compared to Sequence Diagrams

- Sequence diagrams show the flow of **messages** from one **object** (or actor) to another.
- Activity diagrams show the flow of **control** from one **action** to another.

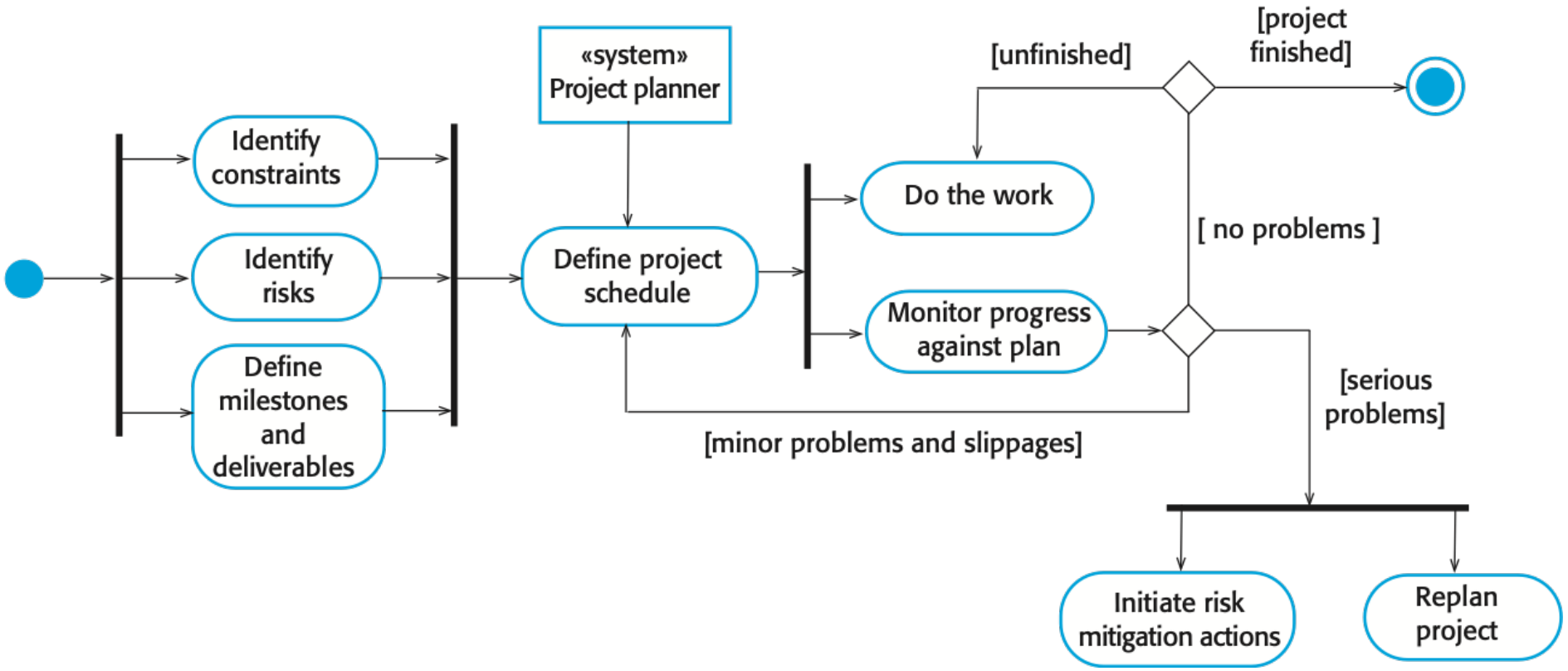
- Ultimately, each type of diagram helps understanding the system from a certain point of view. Used **together**, they help to achieve a comprehensive and detailed understanding of the system.

What is an “Activity”?

Activity vs. Actor or Object

- Activities are tasks (sequences of actions) that **are performed**. They have a starting point and an endpoint in time.
- Objects/actors just **are**. They can **perform** actions (& activities).
- Verbs vs. nouns, e.g. “dispense cash” vs. “cash dispenser”
- Diagrams that focus on actors and objects (**Use case**, **Sequence**) emphasise responsibilities (**who** does what).
- Diagrams that focus on actions (**Activity**) emphasise the work itself (**what** is being done).

Example: The Project Planning Process[1]



[1] from: Sommerville, Software Engineering, 10th Ed., 2015

Basic Activity Diagram Notation

Do

Action A single step in an activity. There is automatic transition upon its completion.

Edge A transition models control flow.

Diamond A control node to branch or merge the control flow.

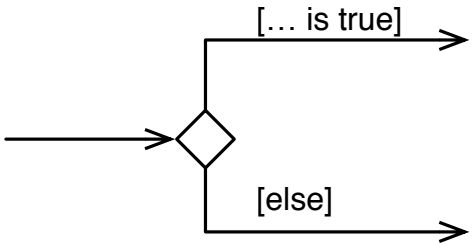
Bar A control node to fork a control flow or to join (synchronise) multiple control flows.

Start Node at which flow starts when the activity is invoked.

Stop Node that stops all flows in an activity.

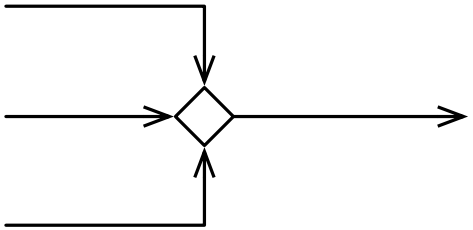
Branching and Merging Control Flows

Branch (Decision)



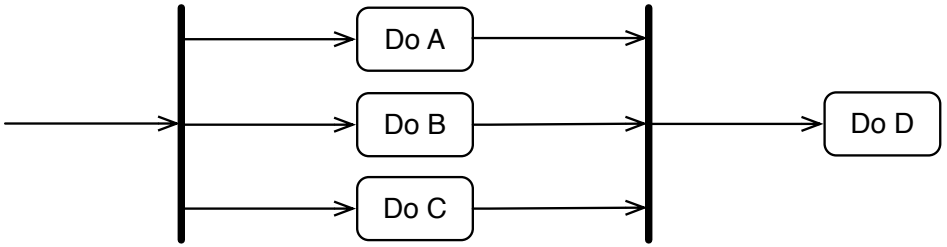
- Models conditional behaviour (decisions), showing the possible **alternative** flows.
- Has one incoming edge and any number of outgoing edges.
- Which of the outgoing edges is actually traversed depends on the evaluation of the guards on the outgoing edges.

Merge



- Merges multiple incoming **alternate** flows into a single outgoing flow.
- Has any number of incoming edges and one outgoing edge.

Forking and Joining Control Flows



Fork

- Is used to split incoming flow into multiple **concurrent** flows (control is transferred simultaneously to **all** actions connected by the outgoing edges).
- Has one incoming edge and multiple outgoing edges.

Join

- Is used to **synchronize** incoming concurrent flows (**all** actions connected by the incoming edges must finish before control is transferred to the action connected to the outgoing edge).
- Has multiple incoming edges and one outgoing edge.

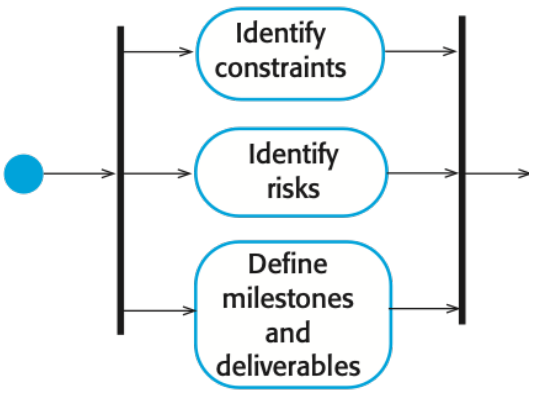
Full Activity Diagram Notation

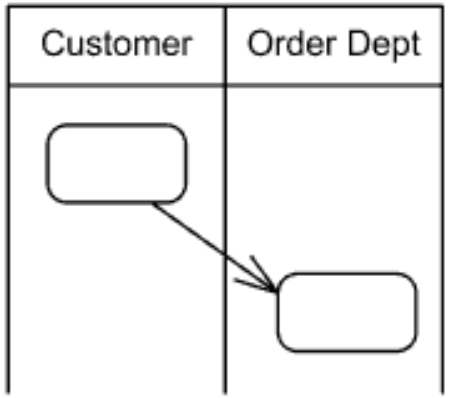
- Not all elements of activity diagrams are presented in this lecture. See: https://www.uml-diagrams.org/activity-diagrams.html for a comprehensive list.

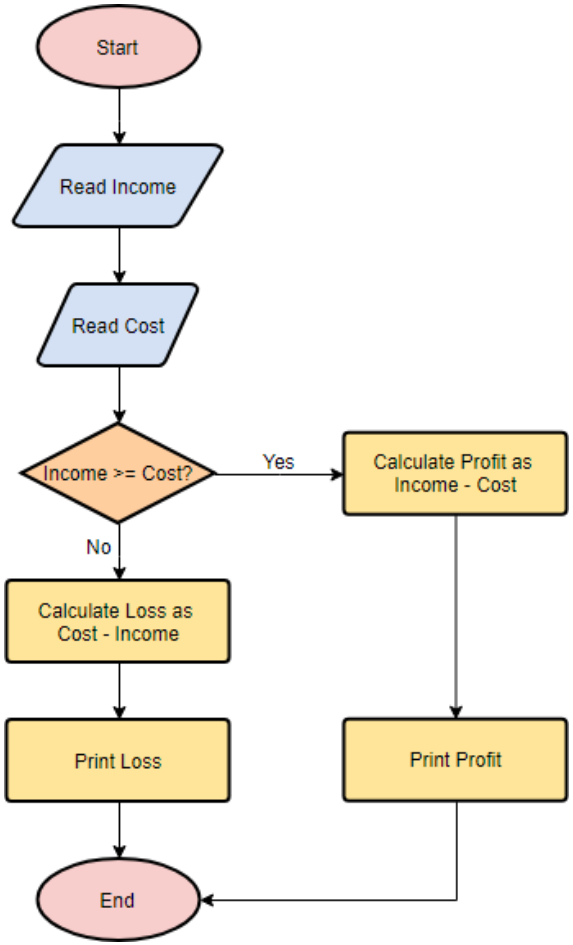
Activity Diagrams vs. Flowcharts (1)

You might already know flowcharts…

- Activity diagrams can model **concurrent activities**. Flowcharts are purely sequential.
- Activity diagrams can **group (partition) activities** by context or concerns.
- Activity diagrams have a richer notation, which can produce more precise models.

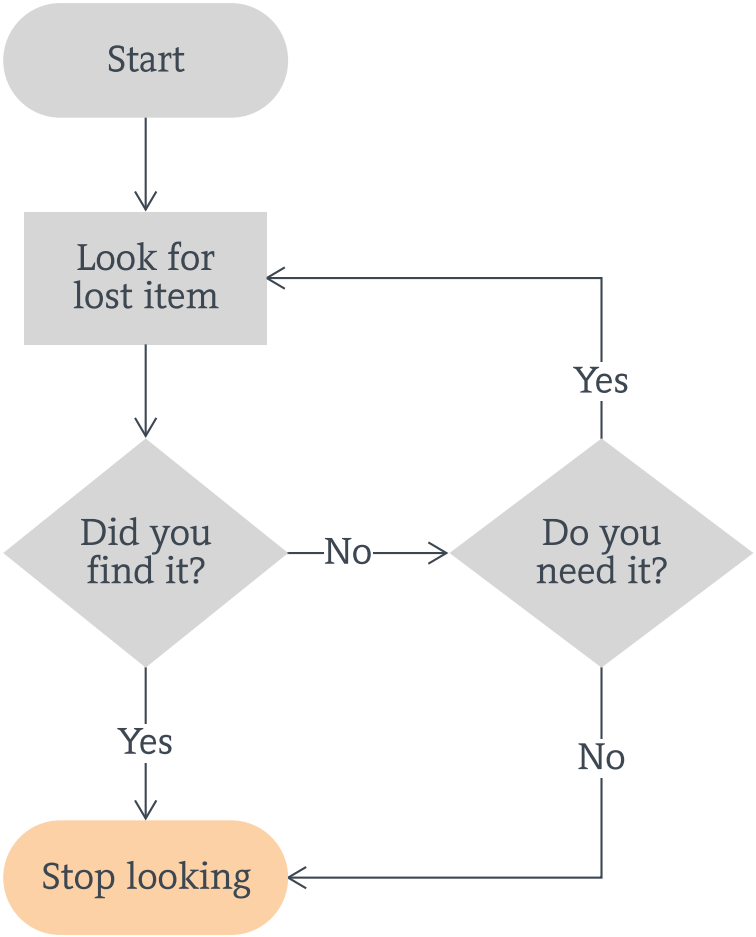


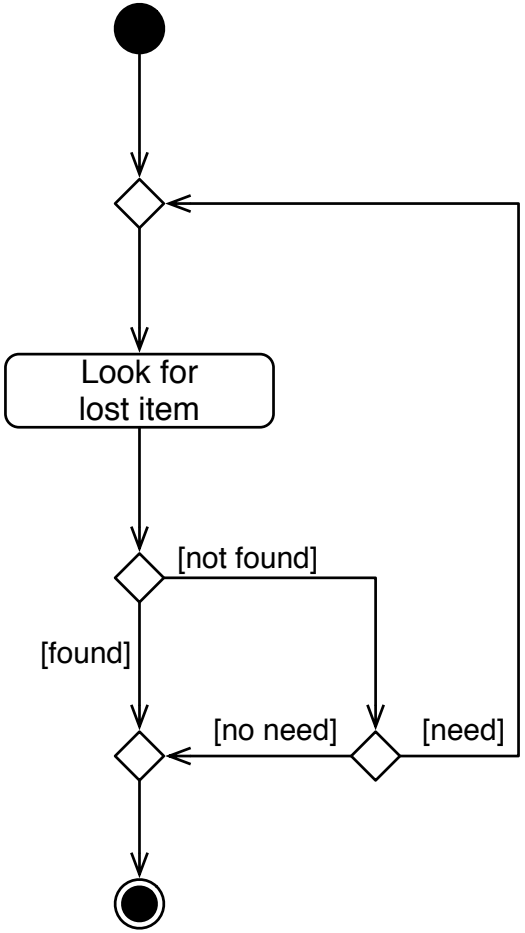




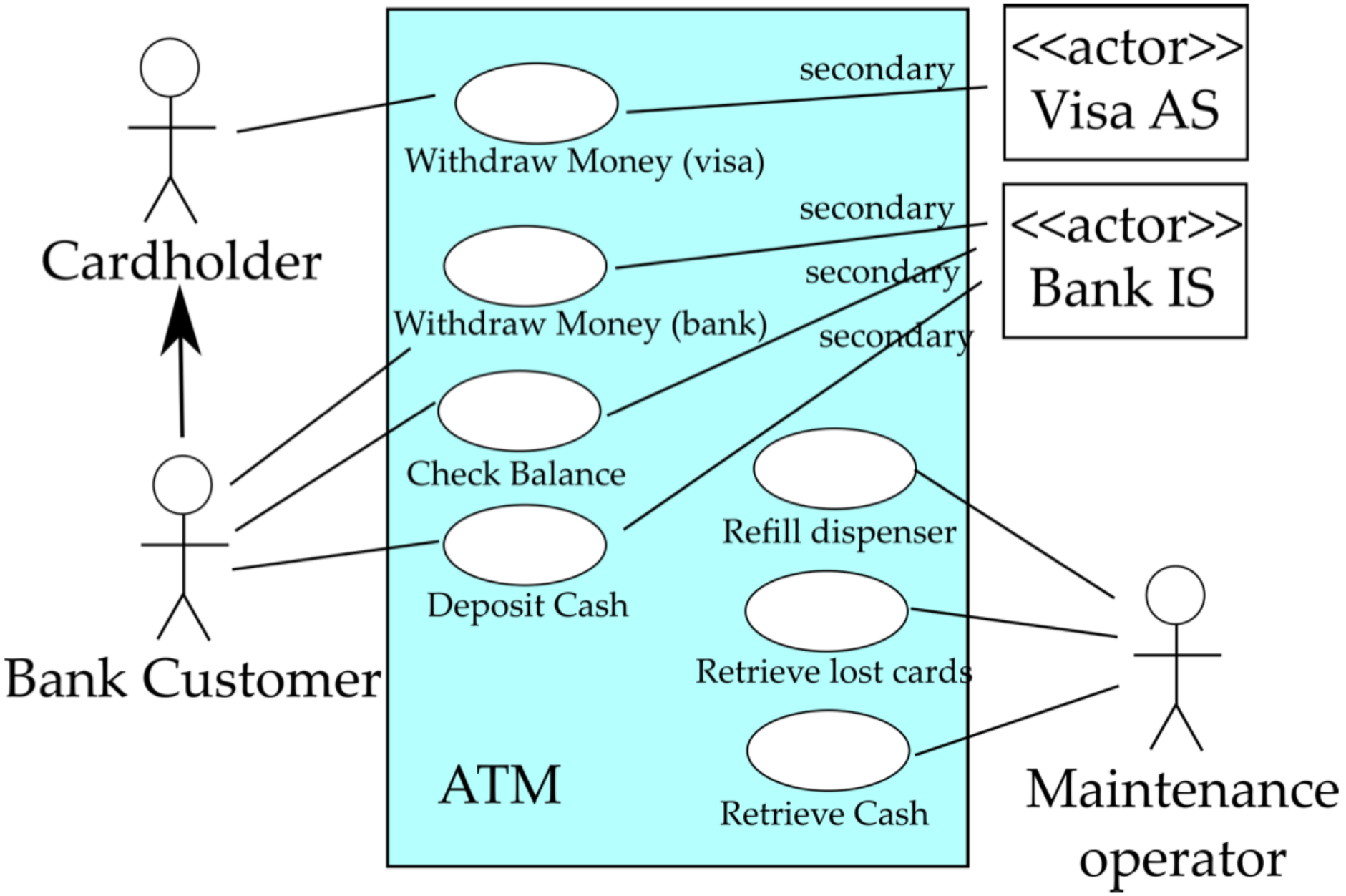
Activity Diagrams vs. Flowcharts (2)

- Activity diagrams have **token semantics**: all incoming transitions must fire before an action can start. With flow charts, only one incoming transition needs to fire to start the next action.





Recall the bank ATM example: Use Case Diagram



Recall the bank ATM example: Use Case Body

For the use case **Withdraw money (Visa)**: withdraw money using Visa card.
Actors: Cardholder, Visa AS.

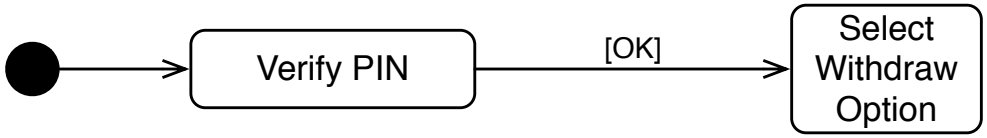
1. Cardholder inserts card	2. Request PIN from Cardholder
3. Cardholder enters PIN	4. Request authorisation from Visa AS
5. Visa AS confirms	6. Show options
7. Cardholder selects “withdraw”	8. Ask Cardholder for desired amount
9. Cardholder enters amount	10. Requests limit from Visa AS
11. Visa AS reports limit	12. Checks if amount below limit
	13. Returns card
14. Cardholder takes card	15. Issues banknotes
16. Cardholder takes banknotes	
Actor actions	**System responsibilities**

Activity Diagram

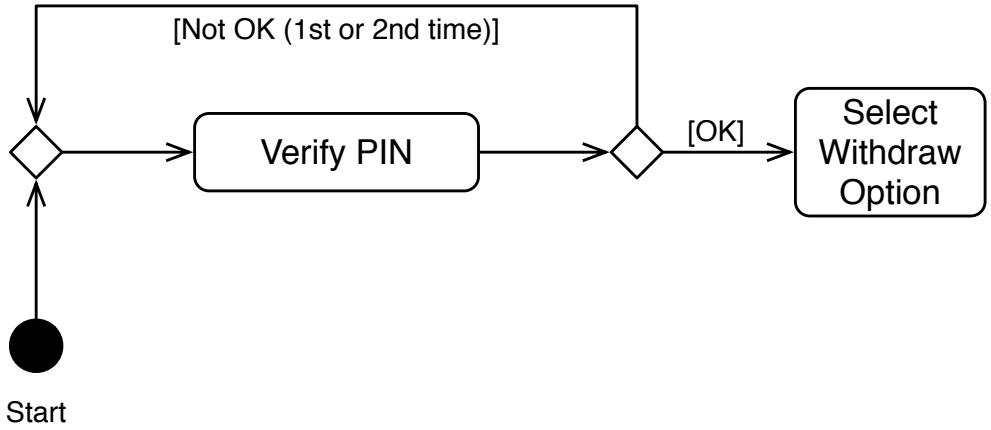
- We draw an activity diagram for a “Withdraw money” use case.

Example: Activity Diagram for “Withdraw” Use Case

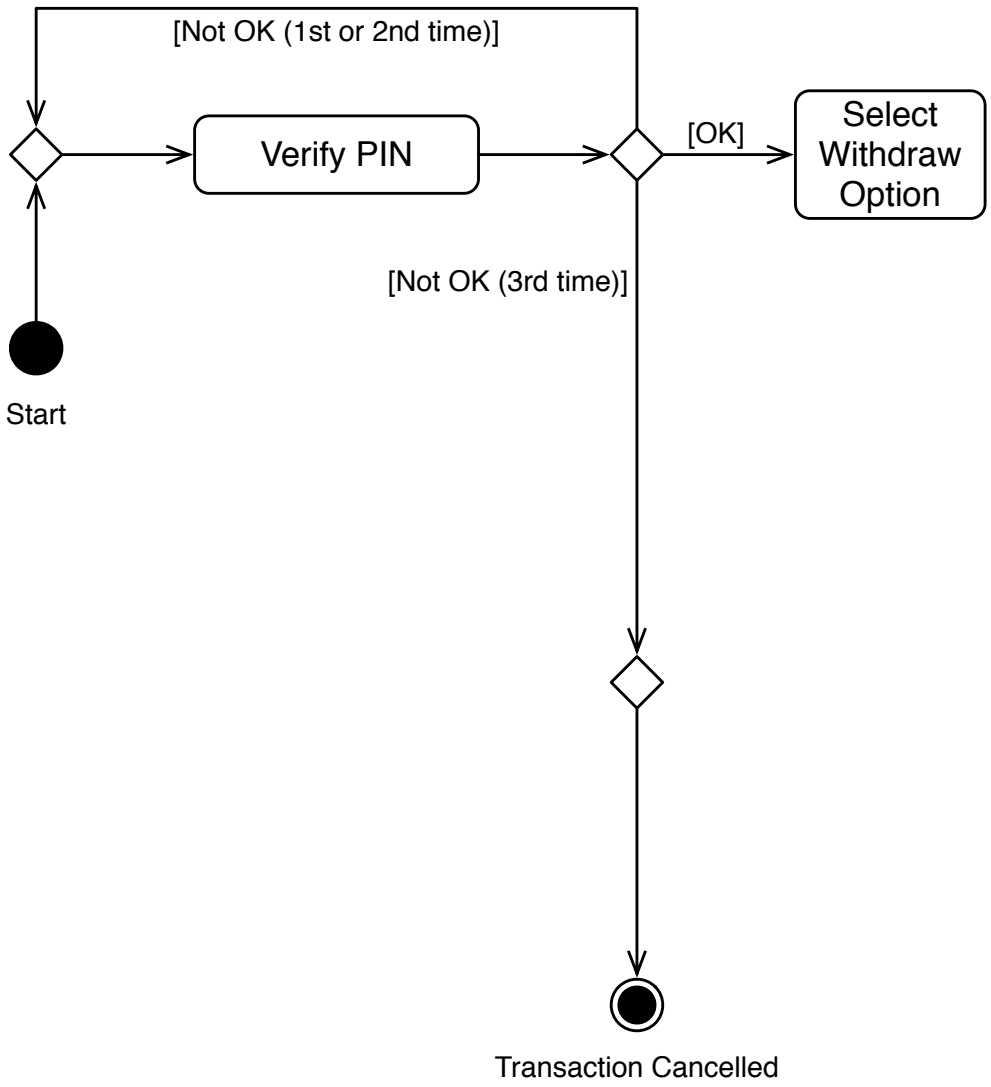
Example: Activity Diagram for “Withdraw” Use Case



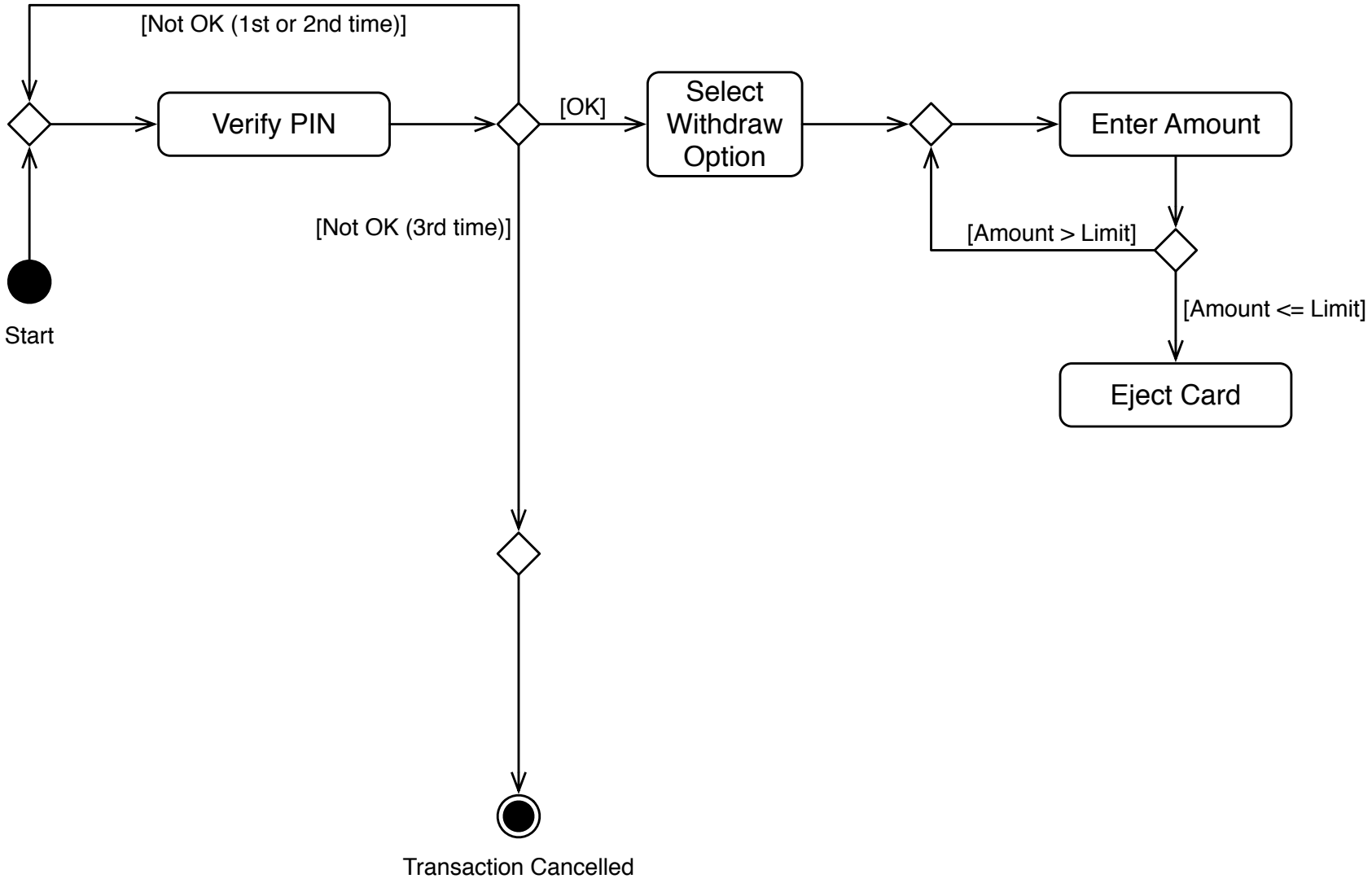
Example: Activity Diagram for “Withdraw” Use Case



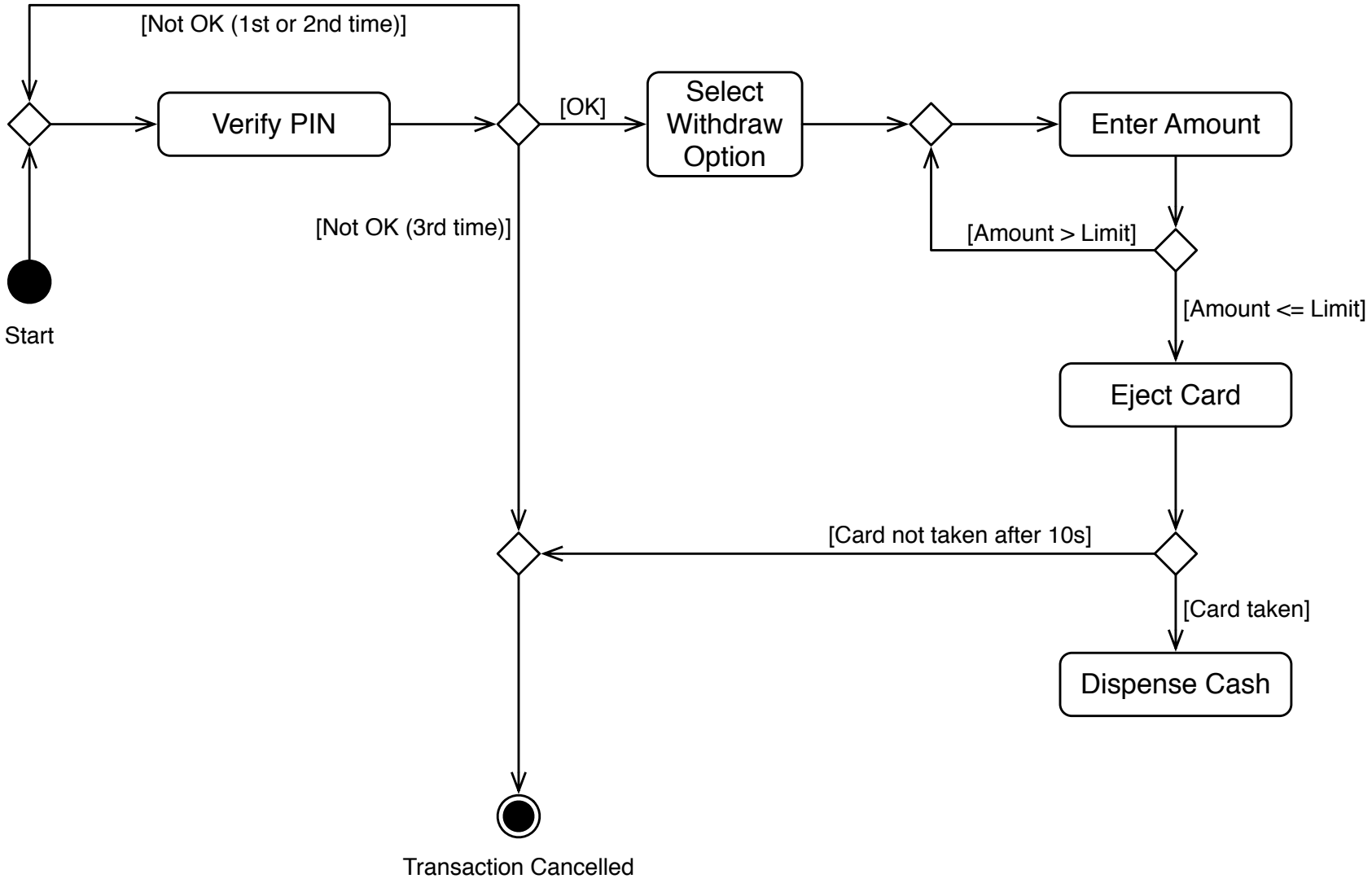
Example: Activity Diagram for “Withdraw” Use Case



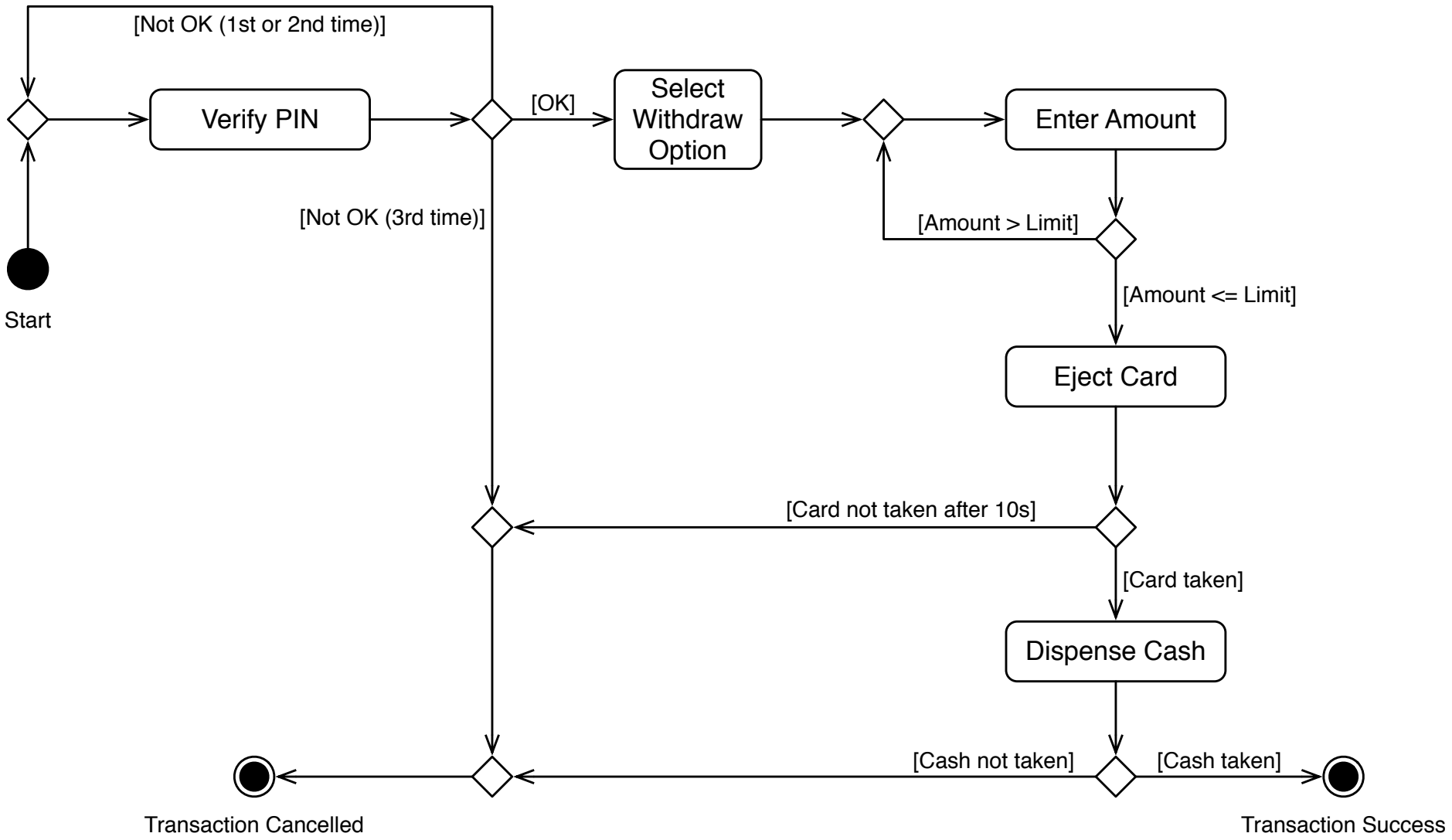
Example: Activity Diagram for “Withdraw” Use Case



Example: Activity Diagram for “Withdraw” Use Case



Example: Activity Diagram for “Withdraw” Use Case



Key Points: Activity Diagrams

- Activity diagrams are a type of behavioural models where the steps of a process are represented as actions connected by the flow of control over time.
- They are able to model sequential and concurrent behaviour.
- Valuable tool for documenting complex processes, usually involving many parties.
- The complexity (level of detail) of a diagram depends on the level of abstraction chosen for the actions. Typically, one would start at a very high level of abstraction, producing a small diagram.
- Later, models can be refined by expanding higher-level (more abstract, general) actions into lower-level (less abstract, more specific) actions.