Activity Diagrams

Massimo Felici

IF-3.46   0131 650 5899

mfelici@staffmail.ed.ac.uk

Activity Diagrams

- Activity Diagrams consist of **activities**, **states** and **transitions** between activities and states

- Activity Diagrams describe
  - how activities are coordinated to provide a service.
  - the events needed to achieve some operation
  - how the events in a single use case relate to one another
  - how a collection of use cases coordinate to create a workflow for an organization

Activity Diagrams' Rationale

- Model business workflows

- Identify candidate use cases, through the examination of business workflows

- Identify pre- and post-conditions for use cases

- Model workflow between/within use cases

- Model complex workflows in operations on objects

- Model in detail complex activities in a high level activity diagram

Activity Diagram Basics

- ## Activities and Actions

- ## Transitions and Activity Edges

- ## Tokens and Activity Nodes

- ## Control Nodes

  - Initial and Final Nodes
  - Forks and Joins
  - Decision and Merge Points

- ## States

- ## Swimlanes

Activities and Actions

- An Activity is the process being modeled

- Activities are the vertices of the diagram

- 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
- There are four ways in which an action can be triggered
  - **On Entry** - as soon as the activity starts
  - **Do** - during lifetime of the activity
  - **On Event** - in response to an event
  - **On Exit** - just before the activity completes

Transitions or Activity Edges

- A **Transition** is the movement from one activity to another, the change from one state to another, or the movement between a state and an activity in either direction

- **Transitions**: unlabelled arrows from one activity to the next.

- **Transitions** take place when one activity is complete and the next can commence

Activity Edges

- The flow of an activity is shown using arrowed lines called **edges** or paths

- **Control-flow Transitions** indicate the order of action states

- **Object-flow Transitions** indicate that an action state inputs or outputs an object

Tokens

- Conceptually, UML models information moving along an edge as a **token** (e.g., real data, an object or focus of control)
- Each edge may have
  - a **weight** associated with it that indicates how many tokens must be available before the tokens are presented to the target action
  - a **guard condition**

Activity Nodes

- UML 2.0 defines several types of activity nodes to model different types of information flow
  - **Parameters nodes**
  - **Object nodes**
  - (input or output) **Pins** – special notation for object nodes; Exception pins, value pins

Initial and Final Nodes

- An **initial node** is the starting point for an activity

- Two types of final nodes: activity final and flow final

- **Activity final nodes** terminate the entire activity

- **Flow final nodes** terminate a path through an activity, but not the entire activity

- It is possible to have multiple initial nodes and final nodes

Forks

- A transition can be split into multiple paths and multiple paths combined into a single transitions by using a **synchronization bar**
- A synchronization may have many in-arcs from activities and a number of out-arcs to activities
- A **fork** is where the paths split
- On an occurrence of the transition all the activities with arcs from the transition are initiated
- A fork node splits the current flow through an activity into multiple concurrent flows

Joins

- A **join** is where the paths meet
- The bar represents synchronization of the completion of those activities with arcs into the transition
- A join synchronizes multiple flows of an activity back to a single flow of execution

Decision and Merge Points

- A **Decision Point** shows where the exit transition from a state or activity may branch in alternative directions depending on a **condition**

- A Decision involves selecting one control-flow transition out of many control-flow transitions based on a condition

- Each branched edge contains a **guard condition**

- **Guard Expressions** (inside []) label the transitions coming out of a branch

- A merge brings together alternate flows into a single output flow – note that it does not synchronize multiple concurrent flows

States

- A state in an activity diagram is a point where some event needs to take place before activity can continue

- Activities and States are similar
  - States carry out actions as activities do
  - Activities need to complete their actions before exiting
  - States are used to imply waiting, not doing

- It is possible to show an object changing states as it flows through an activity

Start and End States

- The Start state is the entry point to a flow.
- There can be several End states. Multiple End states can be used to indicated different follow-on processes from a particular process
- Start and End states can have actions too
- **Mal-formed diagrams**: it is possible to form ill-formed diagrams that require multiple activations of activities or can allow deadlock

Swimlanes

- **Swimlanes** (or **activity partitions**) indicate where activities take place.
- **Swimlanes** can also be used to identify areas at the technology level where activities are carried out
- **Swimlanes** allow the partition an activity diagram so that parts of it appear in the swimlane relevant to that element in the partition

Sending and Receiving Signals

- In activity diagrams, signals represent interactions with **external participants**
- Signals are messages that can be sent or received
- A receive signal has the effect of waking up an action in your activity diagram
- Send signals are signals sent to external participants

Advanced Activity Modeling

- **Connectors**
- UML 2.0 provides supports for modeling **Exception Handling**
- It is possible to show that an action, or set of actions, executes over a collection of input data by placing the action in an **expansion region** (<<parallel>>, <<iterative>> or <<stream>>)
- UML 2.0 defines a construct to mode looping in activity diagrams. A **loop node** has three subregions: setup, body and test

- An action is said to be **streaming** if it can produce output while it is processing input
- **Interruptible activity region**
- UML 2.0 introduces a new type of activity node, called the **central buffer node**, that provides a place to specify queuing functionality for data passing between object nodes
- A **data store node** is a special type of central buffer node that copies all data that passes through it

How to construct Activity Diagrams

Activity Diagrams for Business Modeling

1. Finding **business actors** and **use cases**

2. Identifying key **scenarios** of business use cases

3. Combining the scenarios to produce comprehensive workflows described using **activity diagrams**

4. Where appropriate, mapping activities to business areas and recording this using **swimlines**

5. Refining complicated high level activities similarly, **nested activity diagrams**
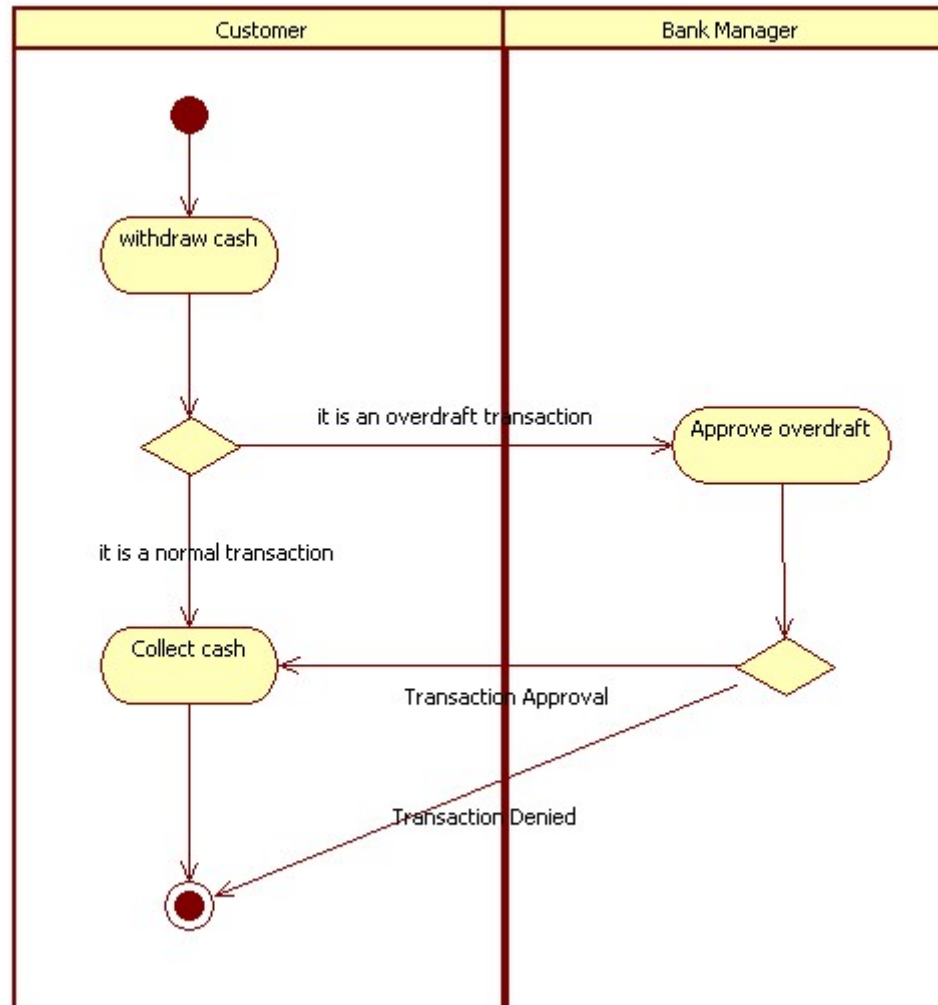
How to construct Activity Diagrams

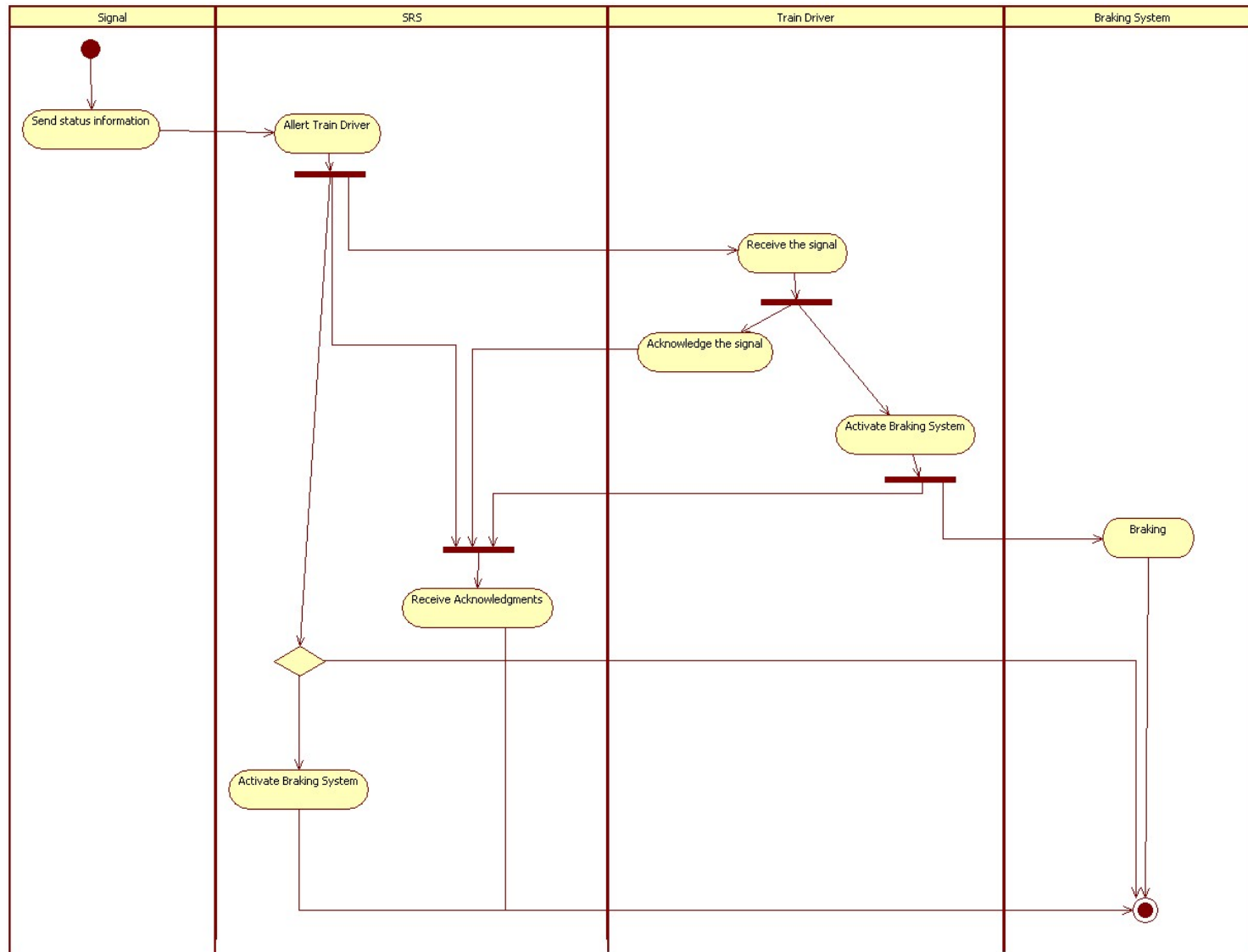*Activity Diagrams for Use Case Modeling*

1. Finding system **Actors, Classes** and **use cases**

2. Identifying key **scenarios** of system use cases

3. Combining the scenarios to produce comprehensive workflows described using **activity diagrams**

4. Where significant object behavior is triggered by a workflow, adding **object flows** to the diagrams

5. Where workflows cross technology boundaries, using **swimlines** to map the activities

6. Refining complicated high level activities similarly, **nested activity diagrams**

An Example of Activity Diagram

Customer	Bank Manager

withdraw cash

it is an overdraft transaction → Approve overdraft

it is a normal transaction

Collect cash

Transaction Approval

Transaction Denied

An Example of Activity Diagram

Readings

- **UML course textbook**
  - Chapter 11 on Activities

Summary

- Activity Diagrams are good for describing synchronization and concurrency between activities

- Activity diagrams are useful for capturing detailed activities, but they can also capture elements of the high level workflow the system is intended to support

- Partitioning can be helpful in investigating responsibilities for interactions and associations between objects and actors