Activity Diagrams

Massimo Felici

Room 1402, JCMB, KB

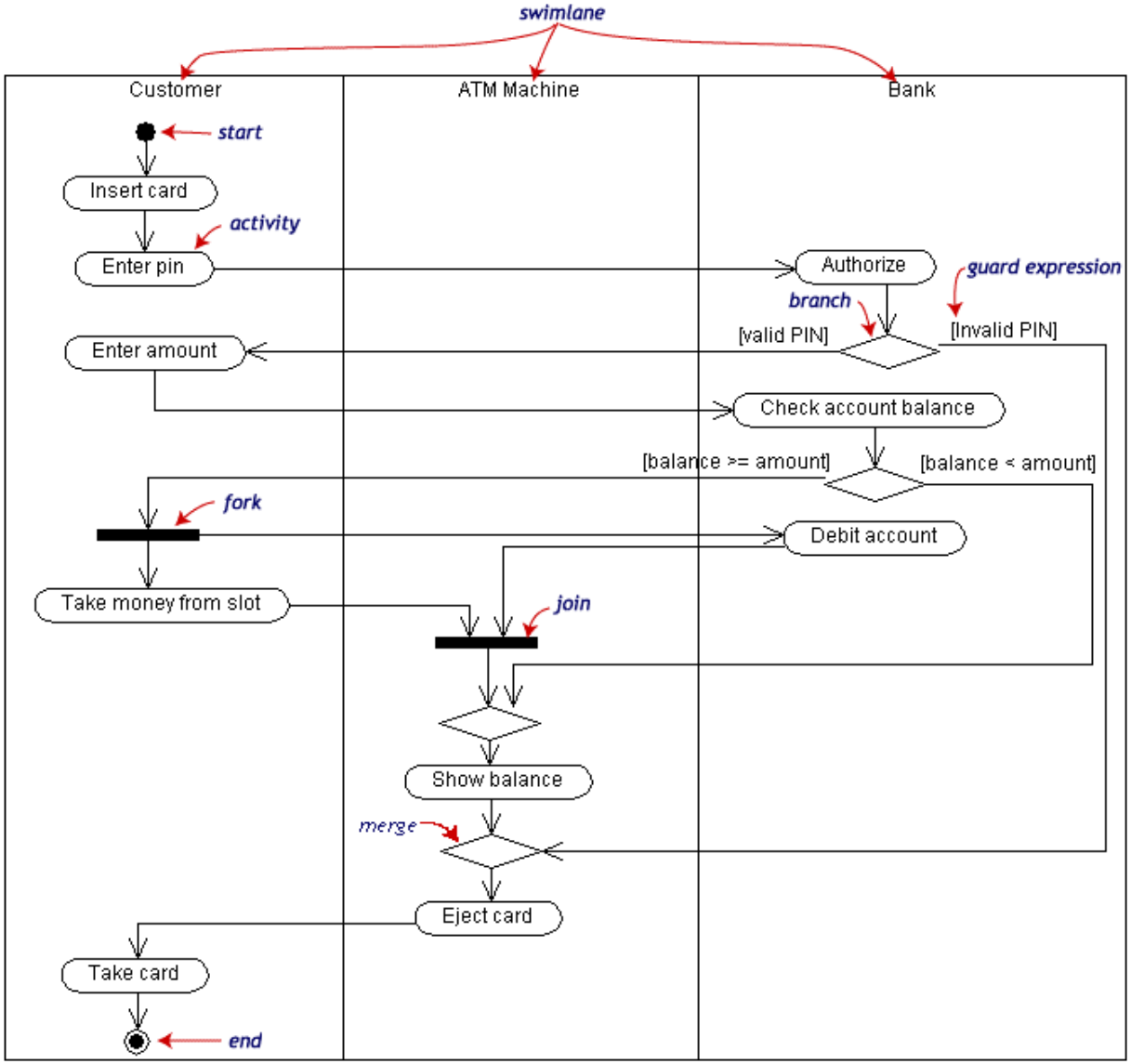
0131 650 5899

mfelici@inf.ed.ac.uk

Activity Diagrams

- **Activity Diagrams** describe
  - how **activities** are **coordinated** to provide a **service**. The service can be at different levels of abstraction.
  - The **events** needed to achieve some operation, particularly where the operation is intended to achieve a number of different things that require **coordination**.
  - How the **events** in a single **use case** relate to one another. In particular, **use cases** where **activities** may overlap and require **coordination**.
  - How a collection of **use cases** **coordinate** to create a **workflow** for an organization.
- Activity Diagrams consist of **activities**, **states** and **transitions** between activities and states

Activity Diagrams at a Glance



Activity Diagrams focus on the **flow** of **activities** involved in a single process. Activity Diagrams show how activities depend on one another.

Activity Diagrams' Purpose

- 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
- States
- Transitions
- Decision Points
- Swimlanes
- Forks and Joins

Activities and Actions

- **Activities** are the vertices of the diagram. This is like a state where the criterion for leaving the state is the completion of the activity.
- An **Activity** is a unit of work that needs to be carried out
  - Any Activity takes **time**
- 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

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
- **Start** and **End** states
  - The Start state is the entry point to a flow. Only one start state is allowed
  - 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

Transitions

- 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
- **Control-flow Transitions** indicate the order of action states
- **Object-flow Transitions** indicate that an action state inputs or outputs an object

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

Swimlines

- **Swimlines** indicate where activities take place.
- **Swimlines** can also be used to identify areas at the technology level where activities are carried out
- **Swimlines** allow the partition an activity diagram so that parts of it appear in the swimline relevant to that element in the partition
- Partitions may be constructed on the basis of:
  - the **class and actor** doing the activity
  - **Partitioning by class and actor** can help to identify new associations that have not been documented in the Class model
  - the **use case** the activity belongs to
  - **Partitioning by use cases** can help document how use cases interact

Forks and Joins

- 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
- The bar represents synchronization of the completion of those activities with arcs into the transition
- A **Join** is where the paths meet
- On an occurrence of the transition all the activities with arcs from the transition are initiated
- A **Fork** is where the paths split

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**

Reading/Activity

- Read Chapter 10, Activity Diagrams, of the Schaum's Outlines UML book
- Read Chapter 11, Statechart Diagrams, of the Schaum's Outlines UML book
- The Java Tutorial provides a quick reference to Java with many working examples.

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