Unified Modeling Language

By
Dr. Tabbasum Naz

Why is UML important?

- Construction trade
- Architects design buildings
- Builders use the designs to create buildings
- The more complicated the building, the more critical the communication between architect and builder
- Blueprints are the standard graphical language that both architects and builders must learn as part of their trade
- Writing software is like constructing a building
- The more complicated the underlying system, the more critical the communication among everyone involved in creating and deploying the software

Why is UML important? (Contd)

- The UML gives everyone from business analyst to designer to programmer a common vocabulary to talk about software design
- The UML is applicable to object-oriented problem solving
- A **model** is an abstraction of the underlying problem
- The **domain** is the actual world from which the problem comes
- Models consist of **objects** that interact by sending each other **messages**
- Objects have things they know (**attributes**) and things they can do (**behaviors** or **operations**). The values of an object's attributes determine its **state**
- **Classes** are the "blueprints" for objects. A class wraps attributes (data) and behaviors (methods or functions) into a single distinct entity.
- Objects are **instances** of classes.

Kinds of Modeling Diagrams

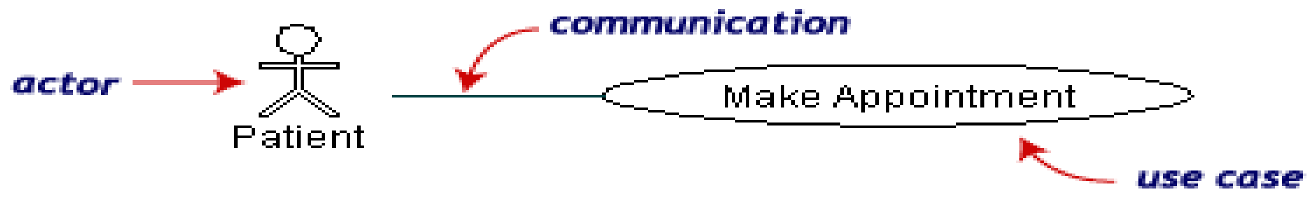
- **Use case diagrams**
- **Class diagrams**
- Object diagrams
- Sequence diagrams
- Collaboration diagrams
- **Statechart diagrams**
- **Activity diagrams**
- Component diagrams
- Deployment diagrams

Use case diagrams

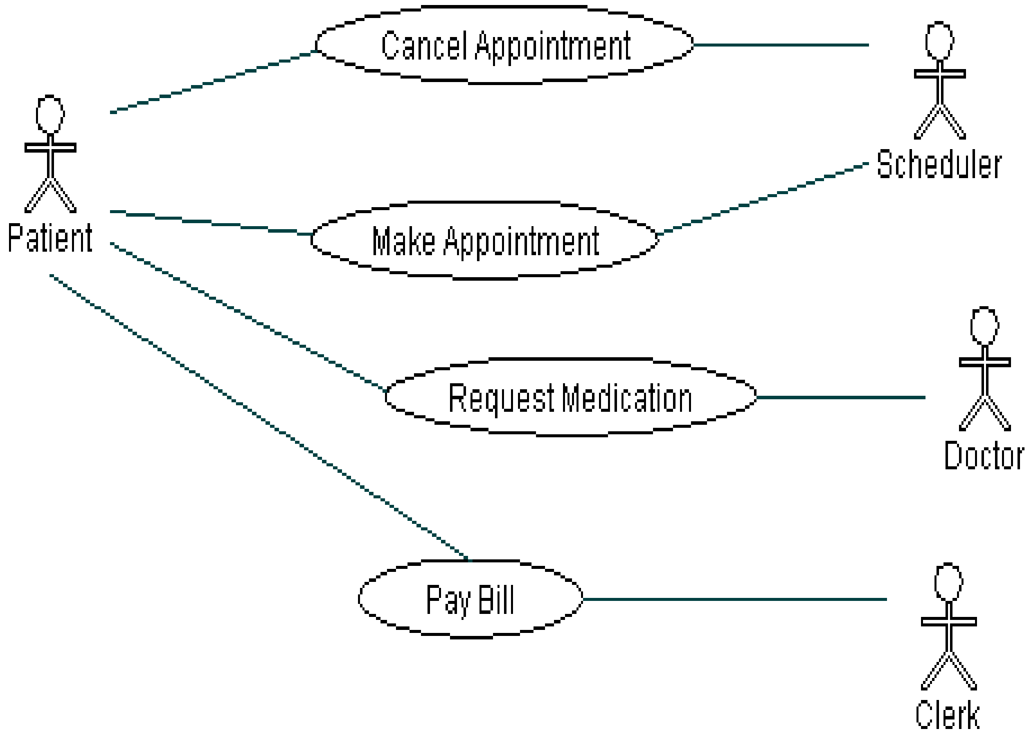
- **Use case diagrams** describe what a system does from the standpoint of an external observer.
- The emphasis is on *what* a system does rather than *how.*
- Use case diagrams are closely connected to scenarios.
- A **scenario** is an example of what happens when someone interacts with the system.
- Medical Clinic Scenario: "A patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "
- A **use case** is a summary of scenarios for a single task or goal.
- An **actor** is who or what initiates the events involved in that task. Actors are simply roles that people or objects play.

Use case diagrams (Contd)

- **Make Appointment** use case for the medical clinic.
- The actor is a **Patient.**
- The connection between actor and use case is a **communication association** (or **communication** for short).



Use case diagrams (Contd)



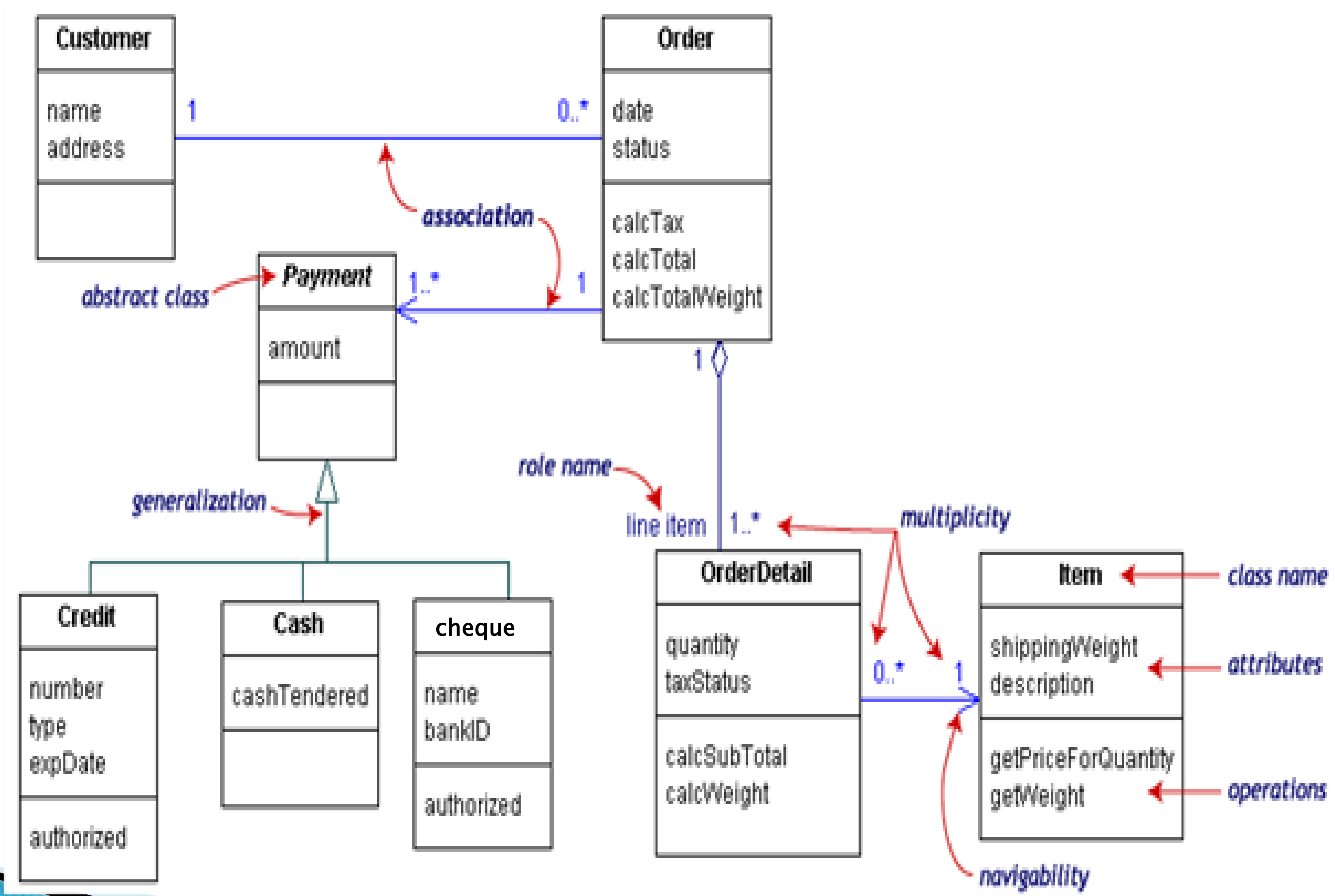
Use case diagrams (Contd)

- Use case diagrams are helpful in three areas.
- **determining features (requirements)**. New use cases often generate new requirements as the system is analyzed and the design takes shape.
- **communicating with clients**. Their notational simplicity makes use case diagrams a good way for developers to communicate with clients.
- **generating test cases**. The collection of scenarios for a use case may suggest a suite of test cases for those scenarios.

Class Diagrams

- **A Class diagram** gives an overview of a system by showing its classes and the relationships among them.
- Class diagrams are static -- they display what interacts but not what happens when they do interact.

Class Diagrams (Contd)



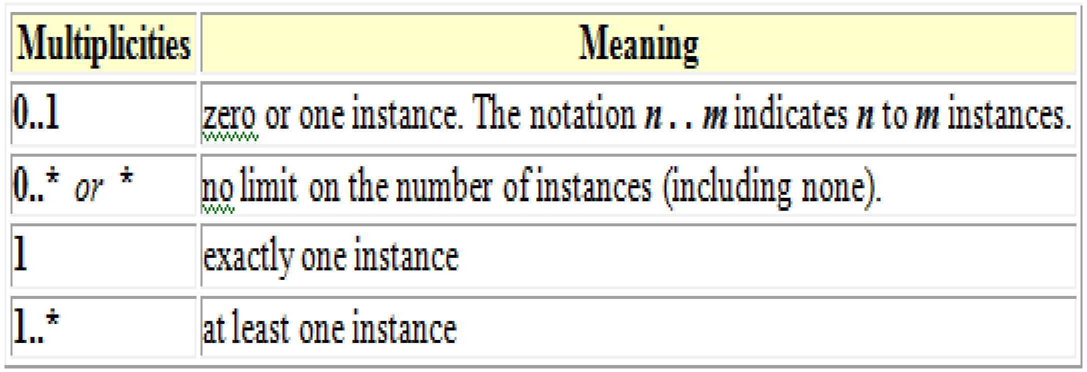
Class Diagram

- Our class diagram has three kinds of relationships.
- **association** -- a relationship between instances of the two classes. There is an association between two classes if an instance of one class must know about the other in order to perform its work. In a diagram, an association is a link connecting two classes.
- **aggregation** -- an association in which one class belongs to a collection. An aggregation has a <u>diamond</u> end pointing to the part containing the whole. In our diagram, **Order** has a collection of **OrderDetails**.
- **generalization** -- an inheritance link indicating one class is a superclass of the other. A generalization has a <u>triangle</u> pointing to the superclass. ***Payment*** is a superclass of **Cash**, **Cheque**, and **Credit**.

Class Diagram (Contd)

The **multiplicity** of an association end is the number of possible instances of the class associated with a single instance of the other end

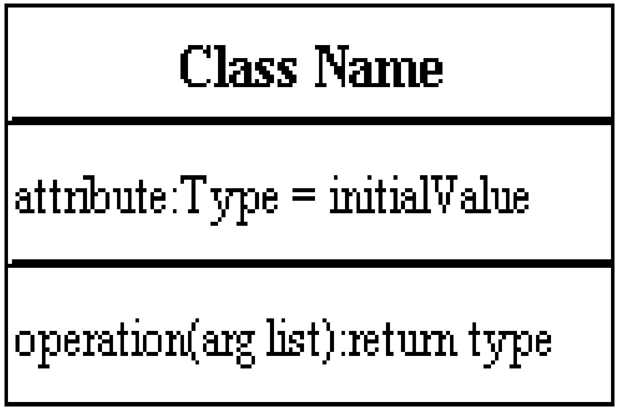
In our example, there can be only one **Customer** for each **Order**, but a **Customer** can have any number of **Orders**.



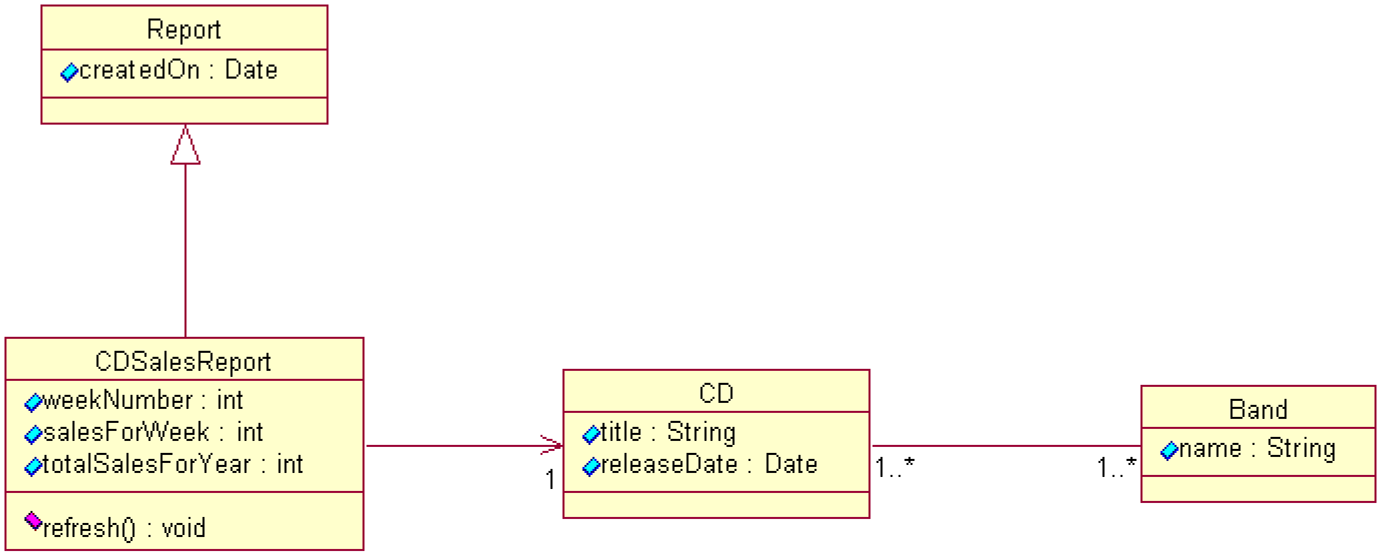
Multiplicities	Meaning
0..1	zero or one instance. The notation ***n* . . *m*** indicates ***n*** to ***m*** instances.
0..* *or* *	no limit on the number of instances (including none).
1	exactly one instance
1..*	at least one instance

Class Diagram (Contd)

- Figure below, illustrates classes with rectangles divided into compartments.
  - Place the name of the class in the first partition (centered, bolded, and capitalized),
  - list the attributes in the second partition,
  - and write operations into the third.


Class Name
attribute:Type = initialValue
operation(arg list):return type

Class Diagram (Contd)



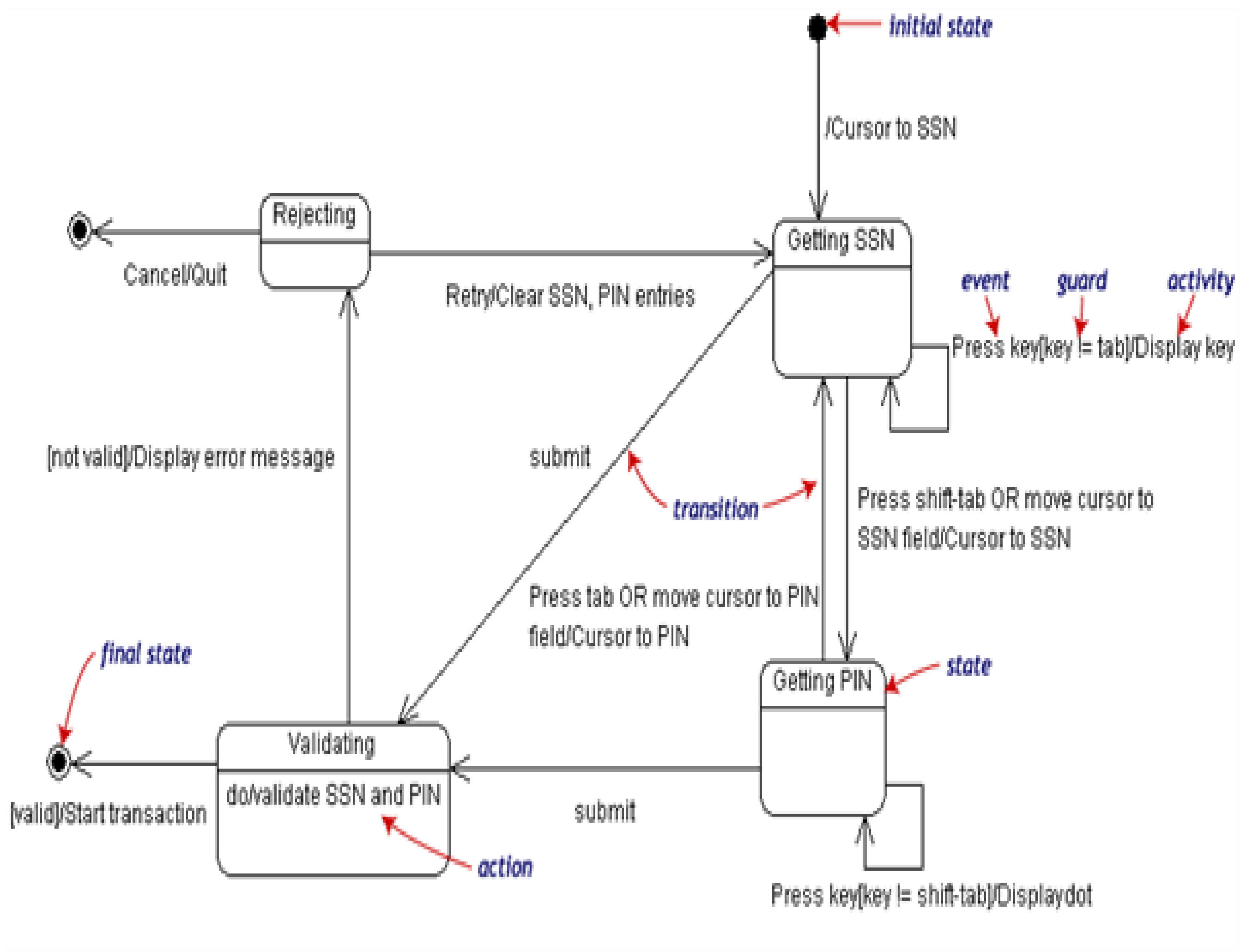
Statechart diagrams

- Objects have behaviors and state.
- The state of an object depends on its current activity or condition.
- A **statechart diagram** shows the possible states of the object and the transitions that cause a change in state.
- Our example diagram models the login part of an online banking system. Logging in consists of entering a valid social security number and personal id number, then submitting the information for validation.
- Logging in can be factored into four non-overlapping states: **Getting SSN**, **Getting PIN**, **Validating**, and **Rejecting**. From each state comes a complete set of **transitions** that determine the subsequent state.

Statechart diagrams (Contd)

- States are rounded rectangles.
- Transitions are arrows from one state to another.
- Events or conditions that trigger transitions are written beside the arrows.
- Our diagram has two self-transition, one on **Getting SSN** and another on **Getting PIN**.
- The initial state (black circle) is a dummy to start the action.
- Final states are also dummy states that terminate the action.

Statechart diagrams (Contd)

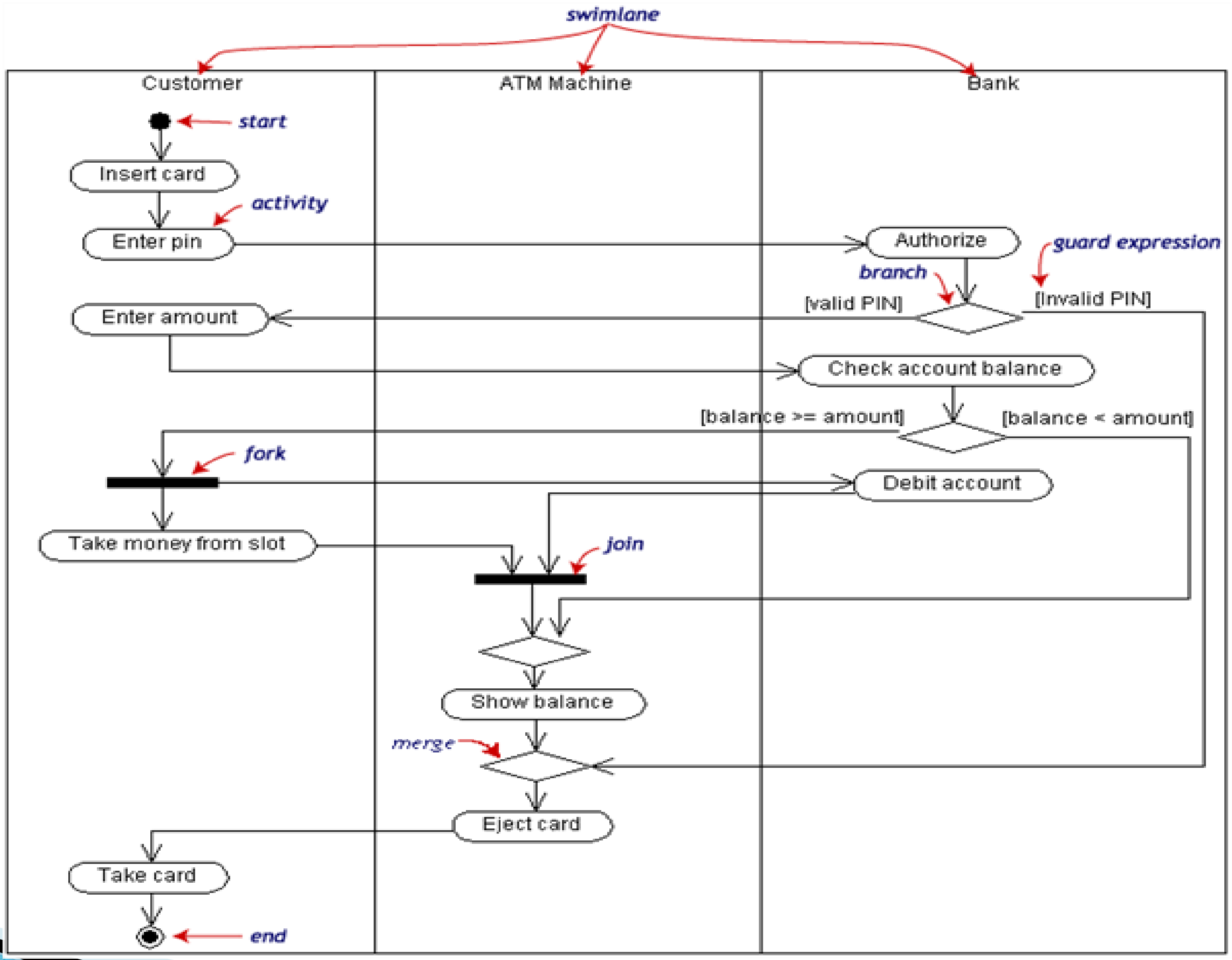


initial state
/Cursor to SSN
Rejecting
Cancel/Quit
Retry/Clear SSN, PIN entries
Getting SSN
event
guard
activity
Press key[key != tab]/Display key
[not valid]/Display error message
submit
transition
Press shift-tab OR move cursor to SSN field/Cursor to SSN
Press tab OR move cursor to PIN field/Cursor to PIN
final state
Getting PIN
state
Validating
do/validate SSN and PIN
[valid]/Start transaction
submit
action
Press key[key != shift-tab]/Displaydot

Activity diagrams (Contd)

- An **activity diagram** is essentially a fancy flowchart.
- Activity diagrams and statechart diagrams are related.
- While a statechart diagram focuses attention on an object undergoing a process (or on a process as an object), an activity diagram focuses on the flow of activities involved in a single process.
- The activity diagram shows the how those activities depend on one another.
- For our example, we used the following process.
- "Withdraw money from a bank account through an ATM."

Activity diagrams (Contd)



Activity diagrams (Contd)

- Activity diagrams can be divided into object **swimlanes** that determine which object is responsible for which activity.
- A single **transition** comes out of each activity, connecting it to the next activity.
- A transition may **branch** into two or more mutually exclusive transitions.
- **Guard expressions** (inside [ ]) label the transitions coming out of a branch. A branch and its subsequent **merge** marking the end of the branch appear in the diagram as <u>hollow diamonds</u>.
- A transition may **fork** into two or more parallel activities. The fork and the subsequent **join** of the threads coming out of the fork appear in the diagram as <u>solid bars</u>.

UML Tools and Modeling Tools

- Visio
- RationalRose
- Visual Studio
  - Many more.

References

- http://www.ibm.com/developerworks/rational/library/769.html
- http://edn.embarcadero.com/article/31863