# CS 4320 / 7320 Software Engineering

## Design:
## UML and Data Models

# Design: UML and Data Models

Requirements:
- Use Cases and Use Case Diagrams

Design:
- Entity-Relationship Modeling Diagrams
- Class Diagrams
- Activity Diagrams
- Sequence Diagrams
- State Machines

# Use Case Diagrams

Gives a top down perspective of the system

Shows and describes the functional requirements of the system

- *Does not* show the steps to each function or the code
- *Does not* give an order to the system

Describes the typical interactions between the actors and the system

# Use Case Diagrams : Basic Symbols

Actors – key players to the system

Use Cases – key functions to the system

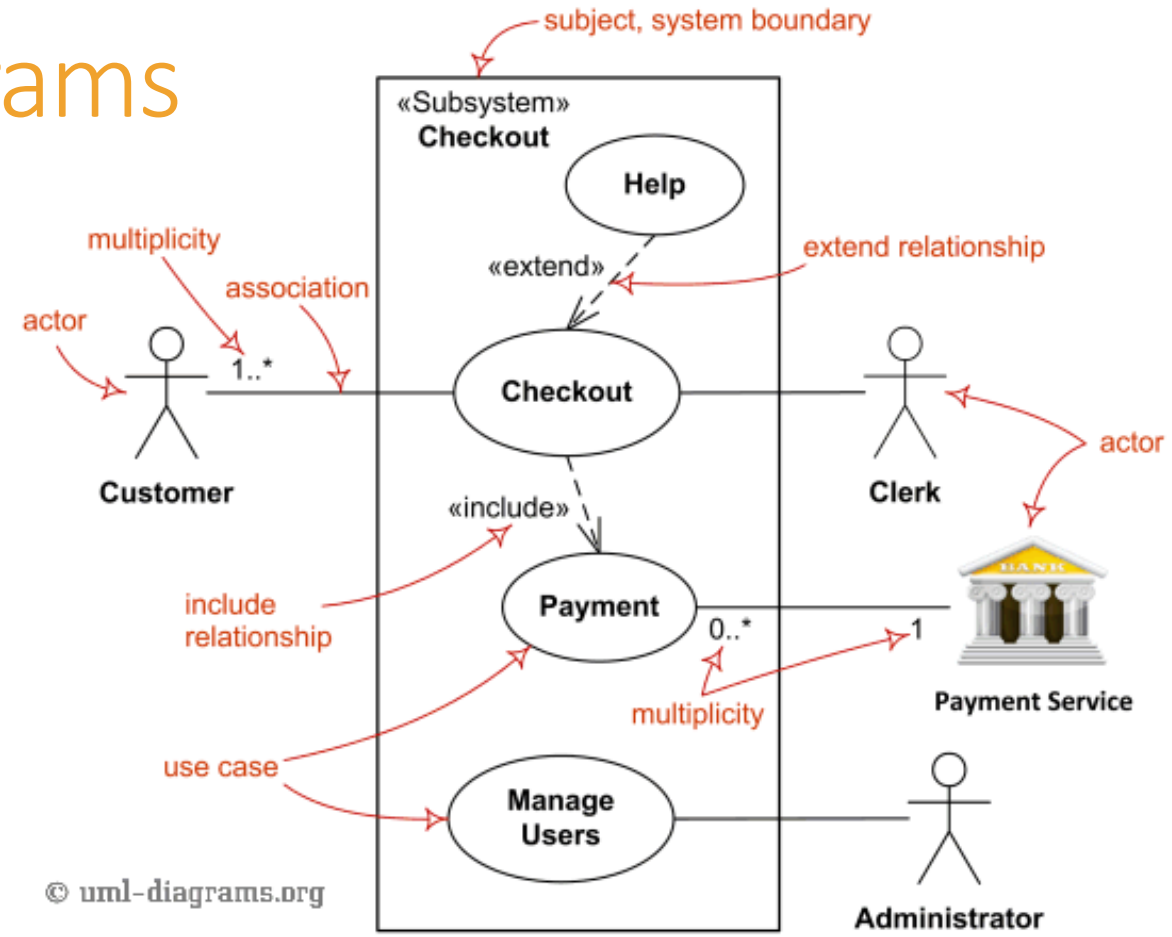
Included Functions – required functions

Extended Functions – optional functions

Use Case Descriptions

- *Overall description of the system*
- *Possible situations of the system*

# Use Case Diagrams



# Use Case Descriptions: Typical Elements

Title (active verb phrase, states main goal)

Description

Triggers

Actors

Preconditions

Main Success Scenario (Goals)

Failed End Condition

Extensions

Steps of Execution (Requirements)

# Use Case Example

Writing Effective use Cases by Alistair Cockburn
ISBN-13: 978-0201702255
ISBN-10: 0201702258
Publisher: Addison-Wesley Professional; (October 15, 2000)

From the pre-publication draft online:
http://alistair.cockburn.us/get/2465

**USE CASE 1: BUY STOCKS OVER THE WEB**

Primary Actor: Purchaser
Scope: Personal Advisors / Finance package ("PAF")
Level: User goal
Stakeholders and Interests:
Purchaser - wants to buy stocks, get them added to the PAF portfolio automatically.
Stock agency - wants full purchase information.
Precondition: User already has PAF open.
Minimal guarantee: sufficient logging information that PAF can detect that something went wrong and can ask the user to provide details.
Success guarantee: remote web site has acknowledged the purchase, the logs and the user's portfolio are updated.
Main success scenario:
1. User selects to buy stocks over the web.
2. PAF gets name of web site to use (E*Trade, Schwabb, etc.) from user.
3. PAF opens web connection to the site, retaining control.
4. User browses and buys stock from the web site.
5. PAF intercepts responses from the web site, and updates the user's portfolio.
6. PAF shows the user the new portfolio standing.
Extensions:
2a. User wants a web site PAF does not support:
    2a1. System gets new suggestion from user, with option to cancel use case.
3a. Web failure of any sort during setup:
    3a1. System reports failure to user with advice, backs up to previous step.
    3a2. User either backs out of this use case, or tries again.
4a. Computer crashes or gets switched off during purchase transaction:
    4a1. (what do we do here?)
4b. Web site does not acknowledge purchase, but puts it on delay:
    4b1. PAF logs the delay, sets a timer to ask the user about the outcome.
    4b2. (see use case *Update questioned purchase*)
5a. Web site does not return the needed information from the purchase:
    5a1. PAF logs the lack of information, has the user *Update questioned purchase*.

# Data Modeling

What are the data entities of the system?

What are the attributes of each entity?

What are the constraints on the attributes of the entities?

What can be used to uniquely identify entities in the system?

How are the different entities related?

# Data Modeling

**Entity:** a person, place, or thing about which we want to collect and store multiple instances of data

**Attribute:** data that describes the entity

**Constraints:** specific rules like not null, unique, check constraints (value < 10), default values

**Unique identifiers:** primary or surrogate key (be careful!)

# Data Modeling

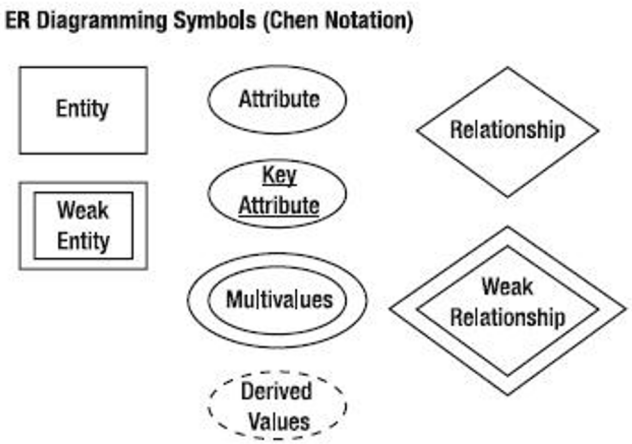
**Relationships:** associations between entities

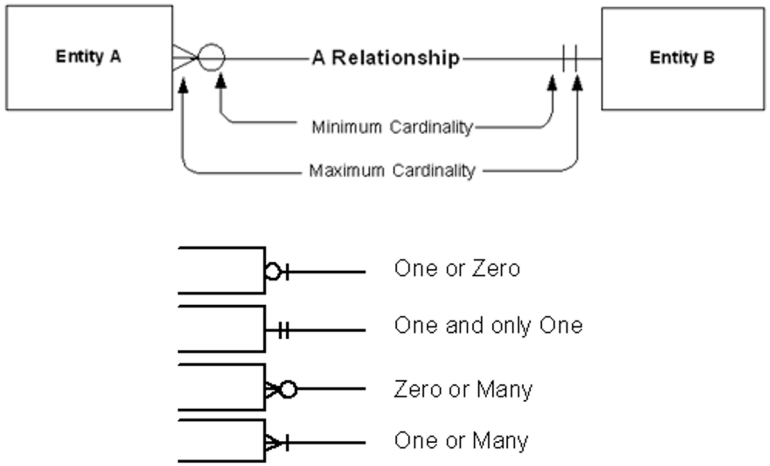
*In logical and physical models, requires foreign keys*

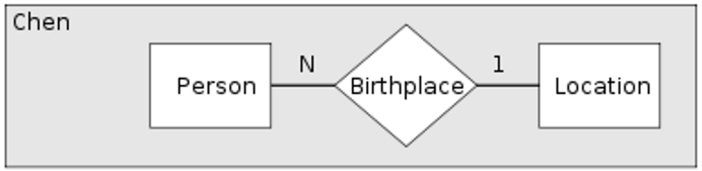
**Cardinality constraints**

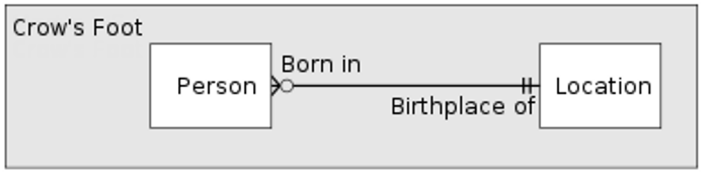
*zero or more; one or more; one and only one; zero or one*

# ERD: Chen vs. Crow's Foot



# Object Oriented Design

## What

- Representing the components of a system as objects, which have a strong correlation to real-world entities
- Modeling the attributes and actions of these entities
- Modeling their inter-relationships and associations

## Why

- Clear approach to capturing the real-world needs in the model
- Straight-forward conversion from model to components

# Object Oriented Design
# What is an Object?

Some entity that models a real-world concept that is either:

- Actor
- Environment
- Entity

Objects are defined by a Class definition

- Attributes / Fields / Members / State
- Actions / Methods

An instantiated Class is an Object

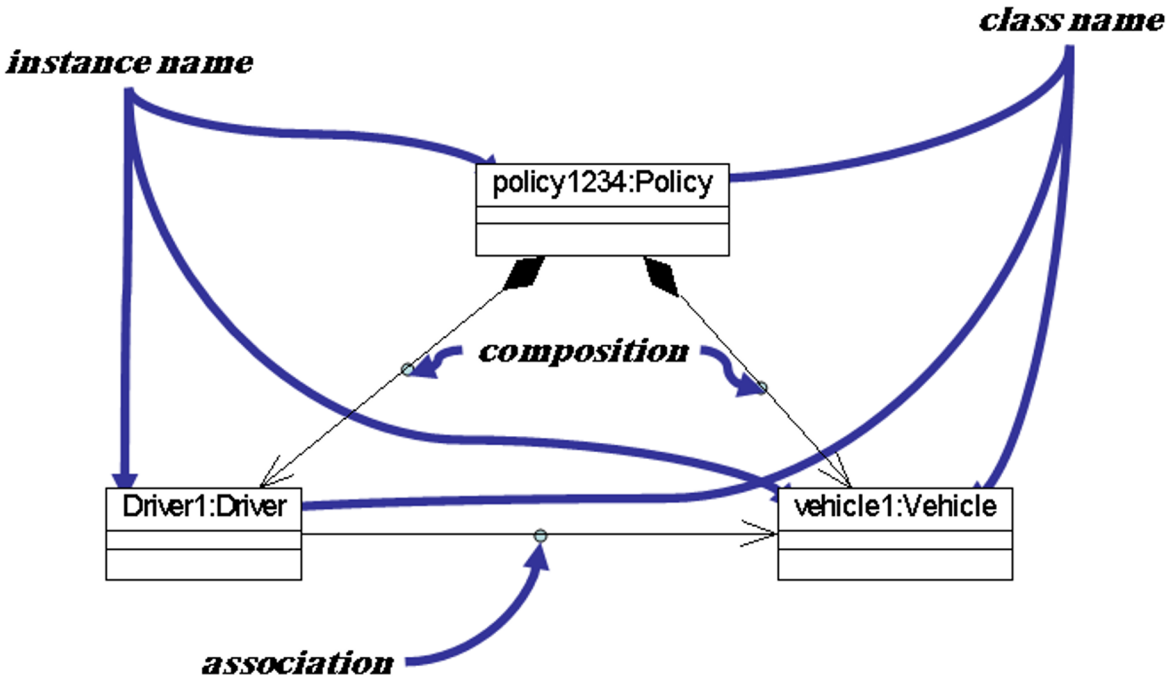
*The class is the blueprint!*

# Object Oriented Design

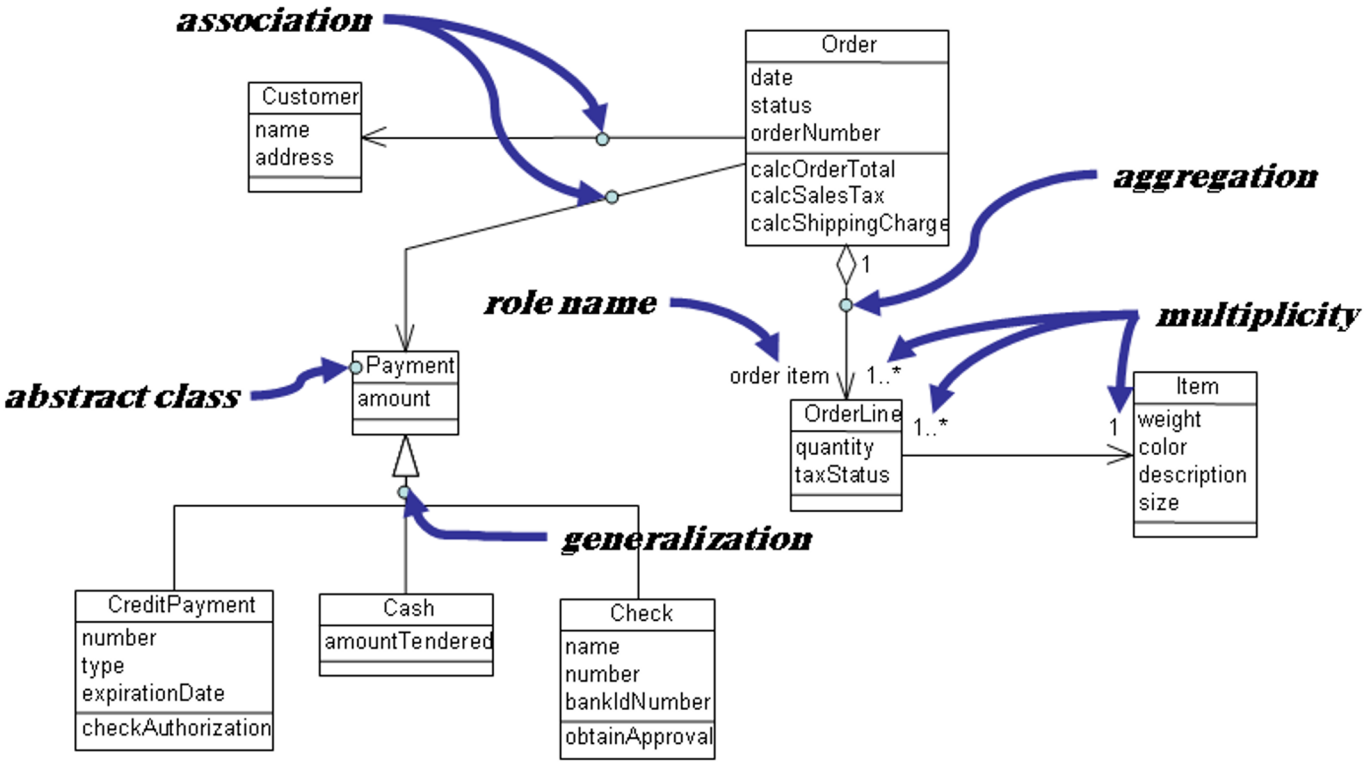
Other topics in OOD:

- *Encapsulation (information hiding)*
- *Abstraction*
- *Inheritance*
- *Polymorphism*

# Object Diagram



# Class Diagram



# UML Class Diagram: HasA

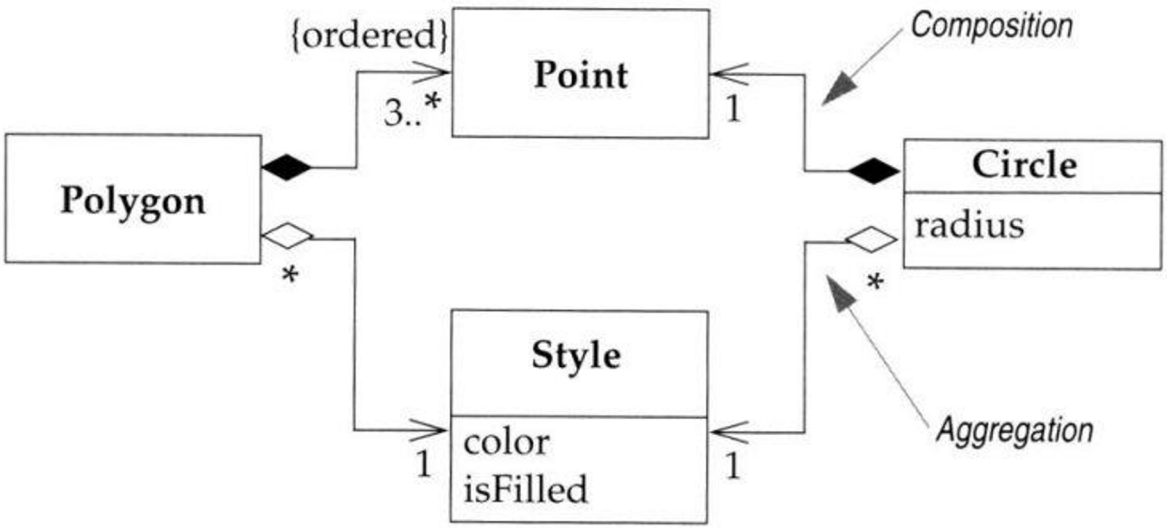
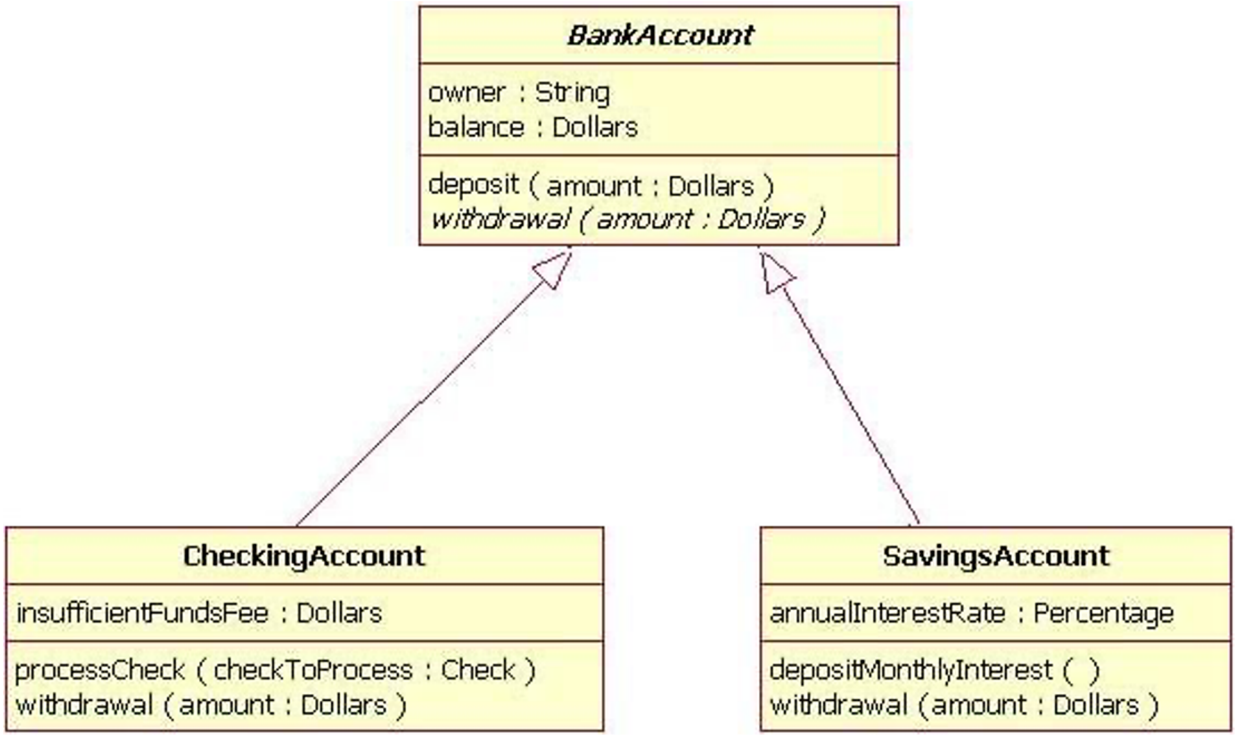



**Figure 6-6:** *Aggregation and Composition*

# UML Class Diagram: Inheritance



# UML Activity Diagrams

Used to describe procedural logic and work flow

Similar to a flowchart

Sometimes used to describe a Use Case

*Shows the procedures needed for each Use Case*

Supports parallel behavior

# UML Activity Diagrams: Symbols

Action

*Initial function to describe*

Fork / Join

*Parallel processes not needed to run in order*

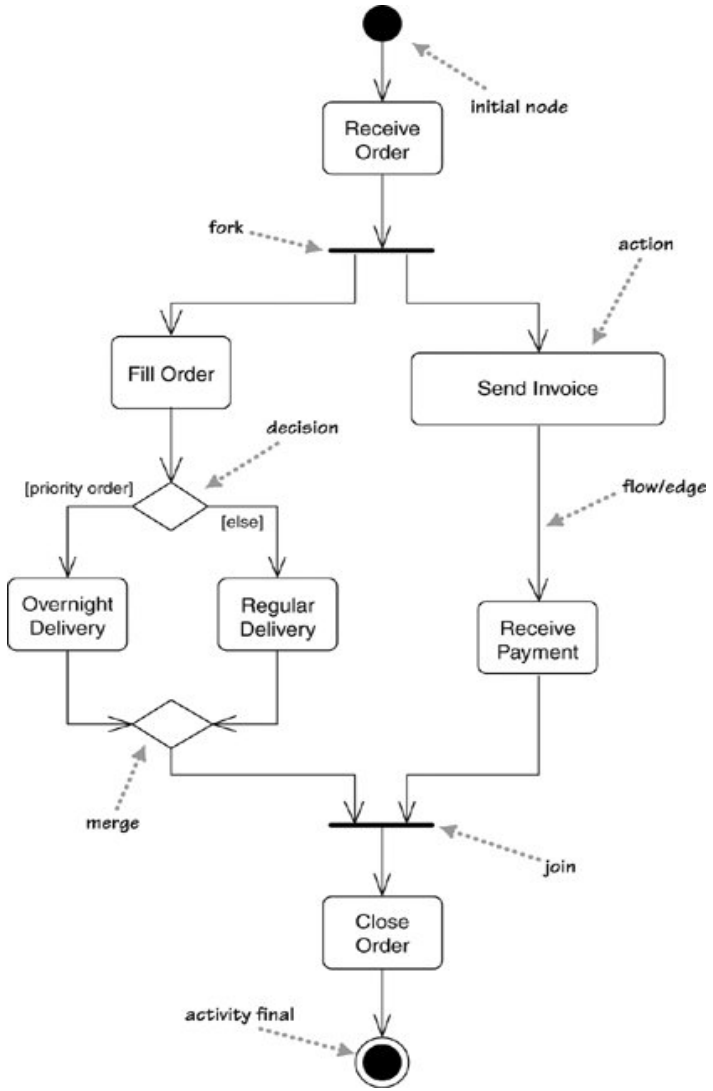
Decision / Merge

*Conditional statements*

*If-then-else statements*

Flow / Edges

# UML Activity Diagram



# UML Sequence Diagram

Interactive Design

- Passes messages between participants (objects)
- Shows the process of a single scenario
- Shows how long each participant is active
- Shows what each participant contributes to the process

# UML Sequence Diagram: Symbols

Object

Timeline

*How long each object is active*

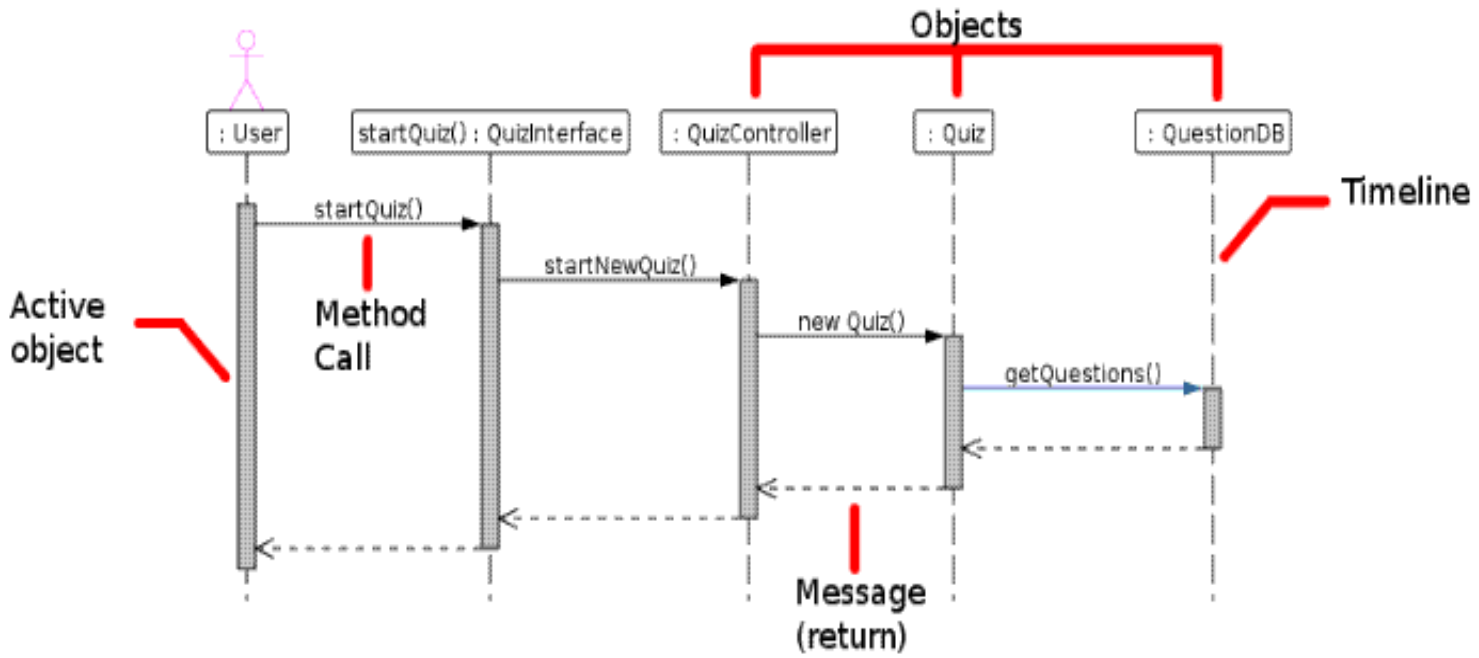
Active Object

Method Call

*Reaching the next object with a call*

Messages

# UML Sequence Diagrams:



# UML Sequence Diagram: Advanced Symbols

Creating and Deleting Objects

*Objects only last while as an active object*

Loops, Conditional Statements

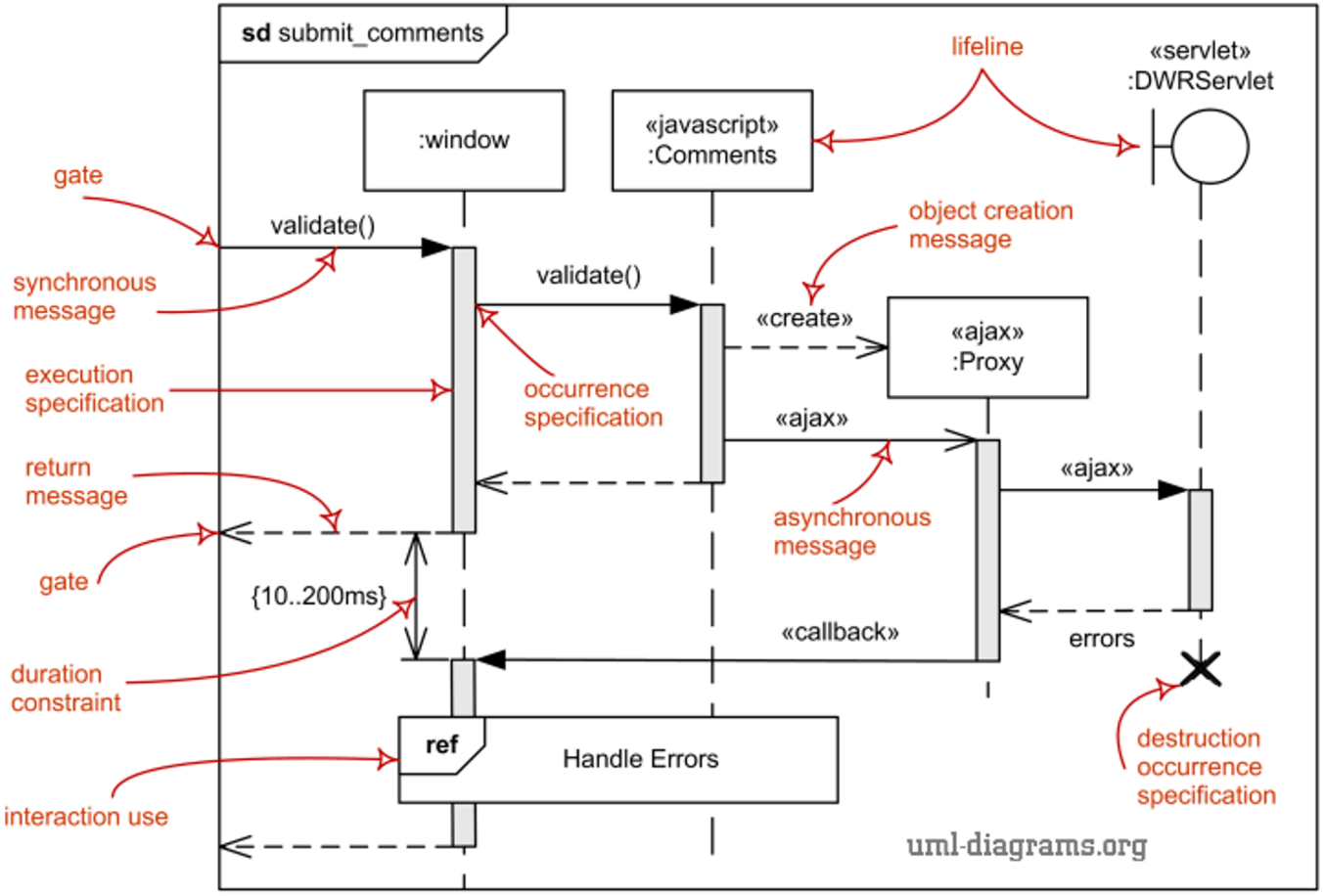
*if – then statements*

Synchronous and Asynchronous Calls

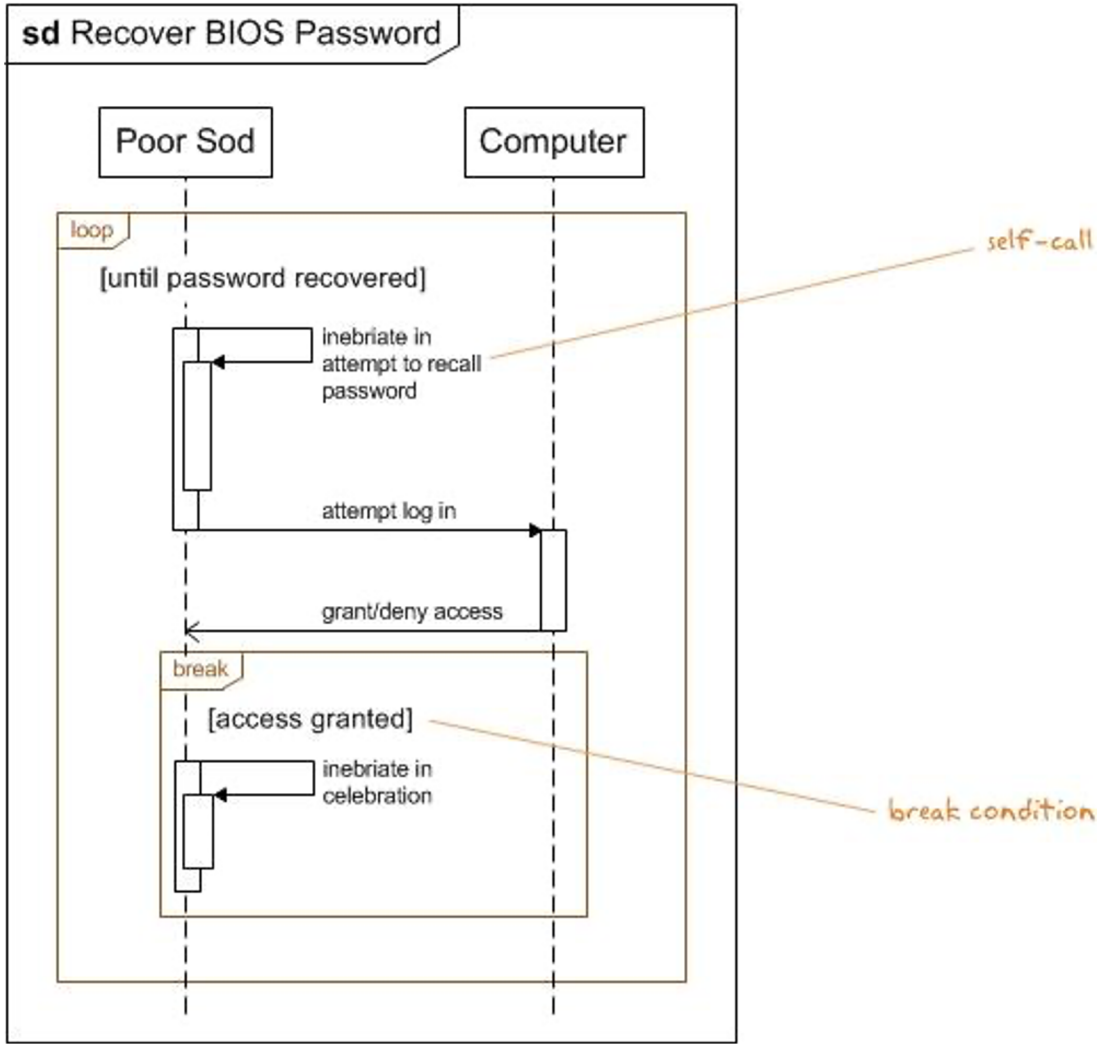
*Sending several messages in parallel*

Common Operators for Interaction Frames

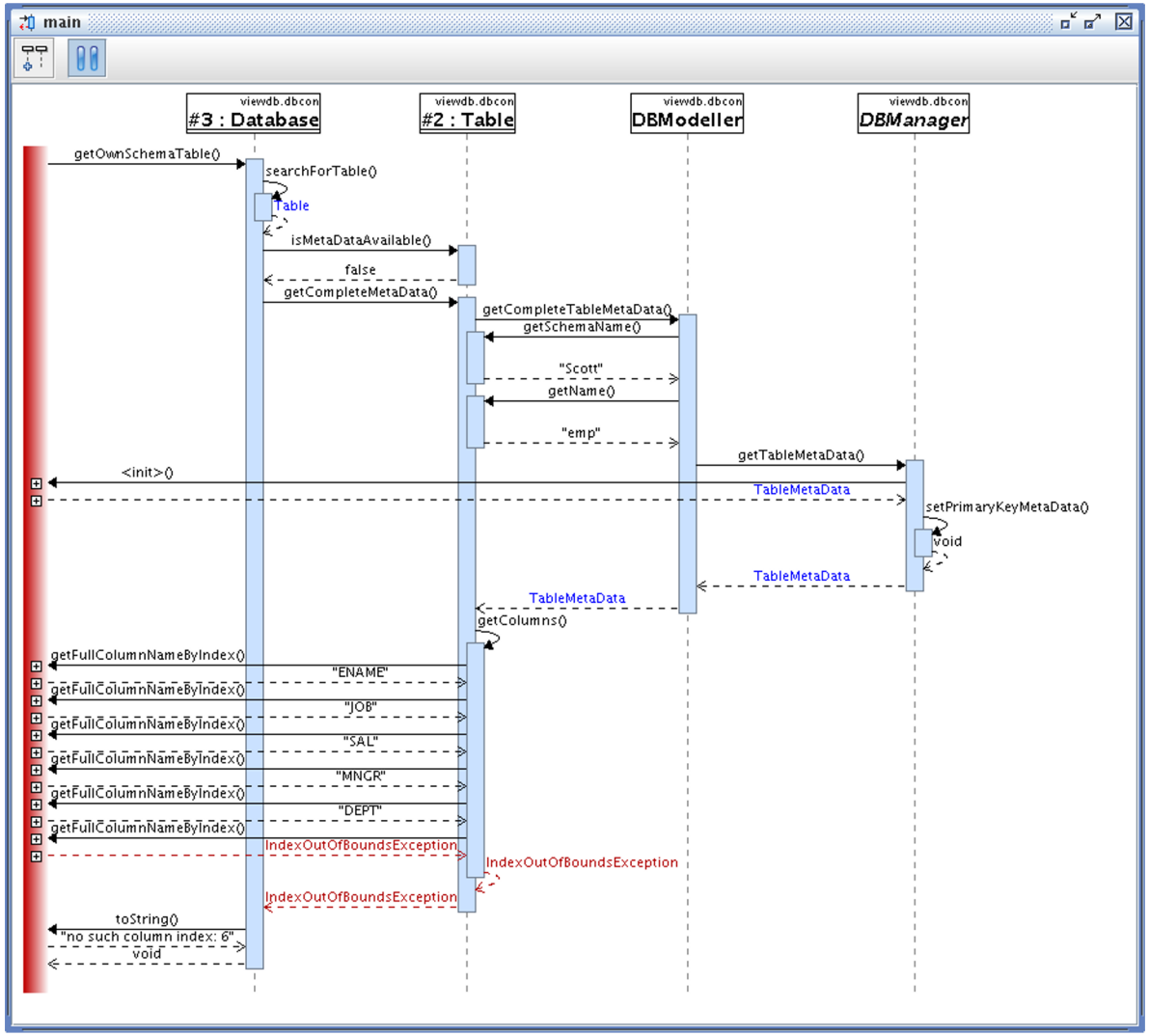
# UML Sequence Diagrams



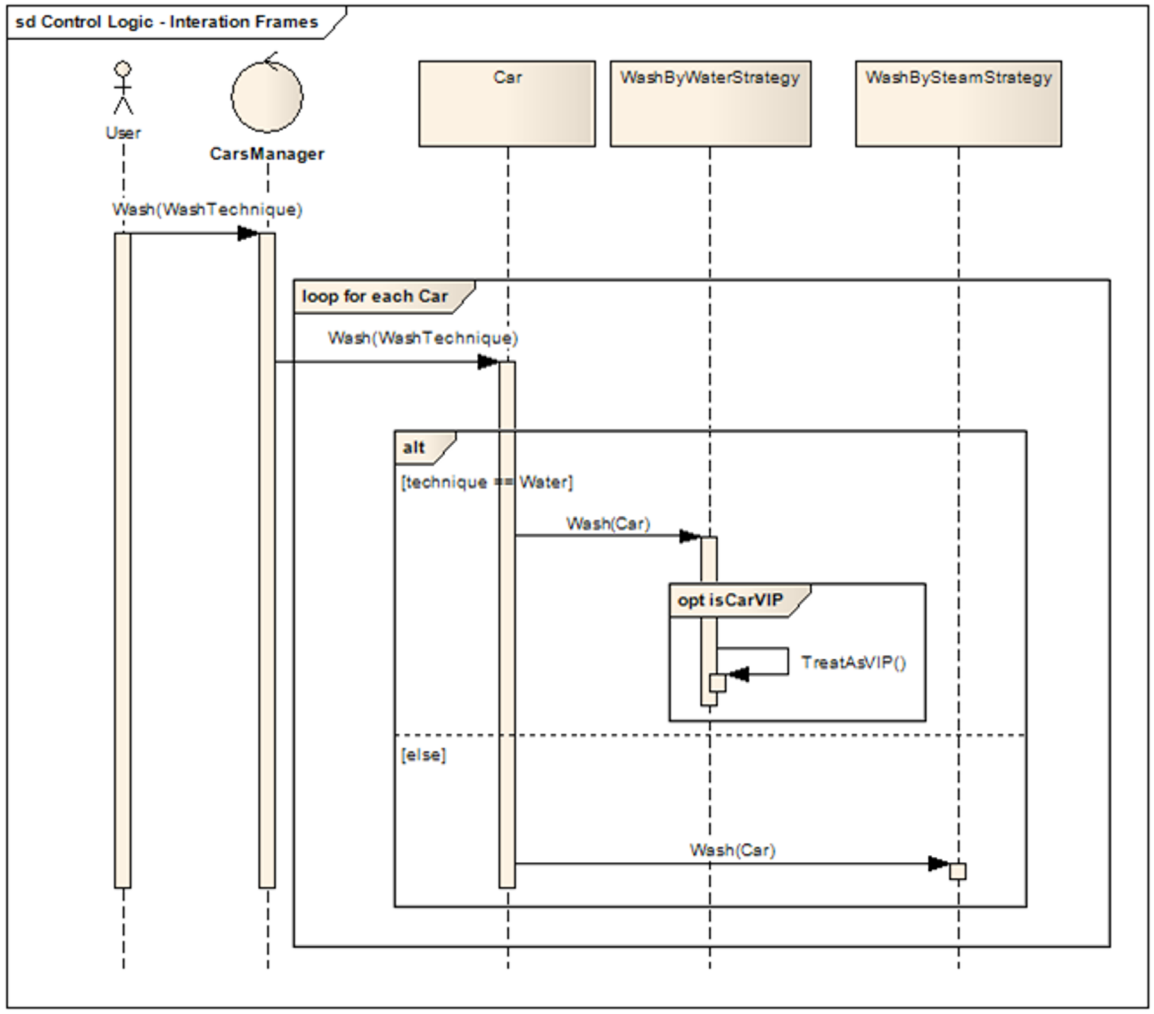
# UML Sequence Diagram: Loops



# UML Sequence Diagram: Threads



# UML Sequence Diagram: Loop/Interaction



# UML State Machines

Another way to describe the behavior of a system

Similar style to the Activity Diagrams

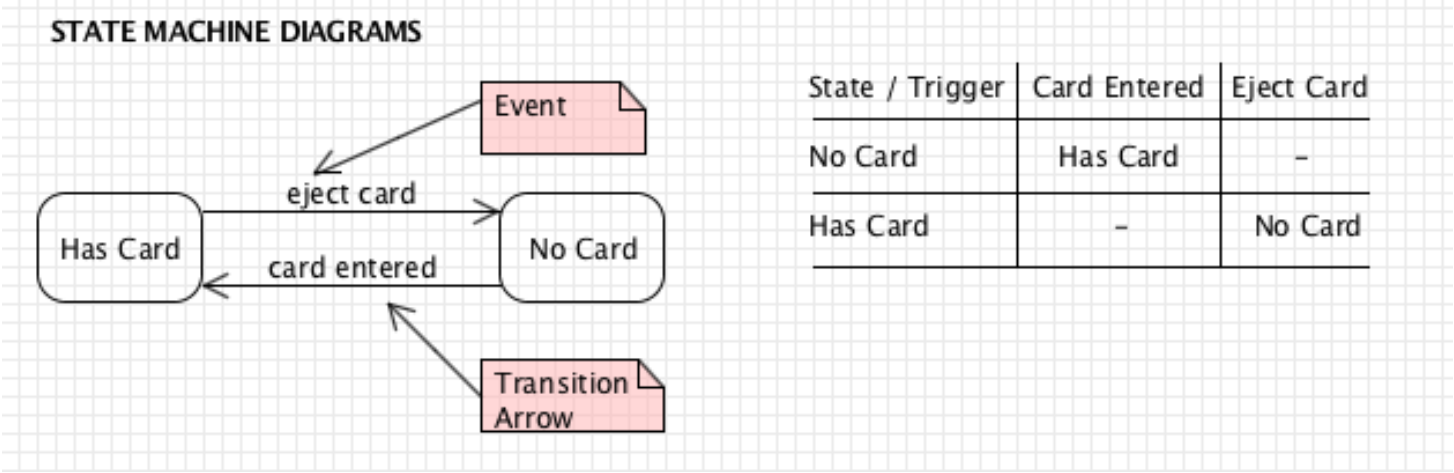
*Closest diagram that represents the code*

In Object-oriented approaches, the state machine diagram shows the lifetime behavior of a single object

State Machines are good at describing the behavior across several use cases.

# UML State Machines: Description/Tables

State Machines model the change of states and the events that cause them

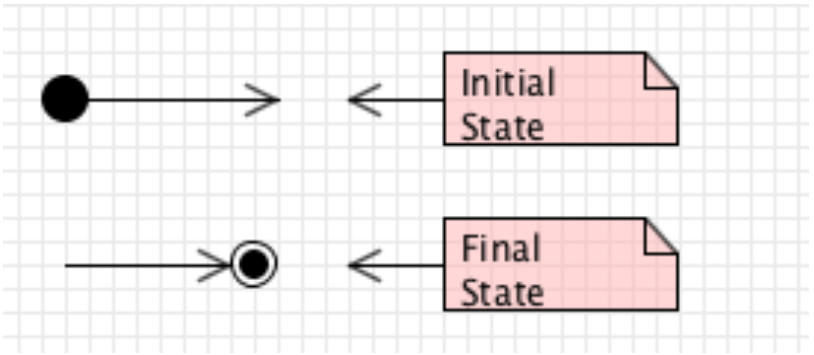



| State / Trigger | Card Entered | Eject Card |
|---|---|---|
| No Card | Has Card | - |
| Has Card | - | No Card |

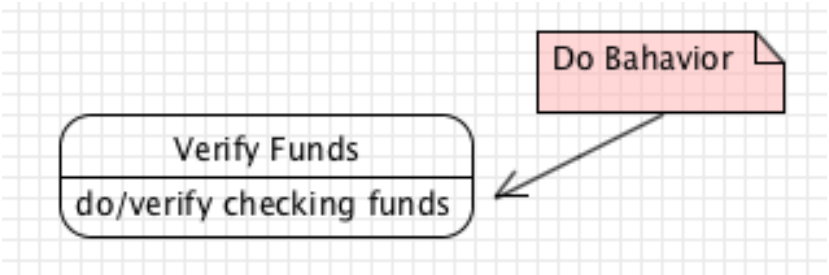
# UML State Machines: Basic Symbols

Initial State – Starting Position

Final State – Ending Position




Do Behavior



# UML State Machines: Basic Symbols

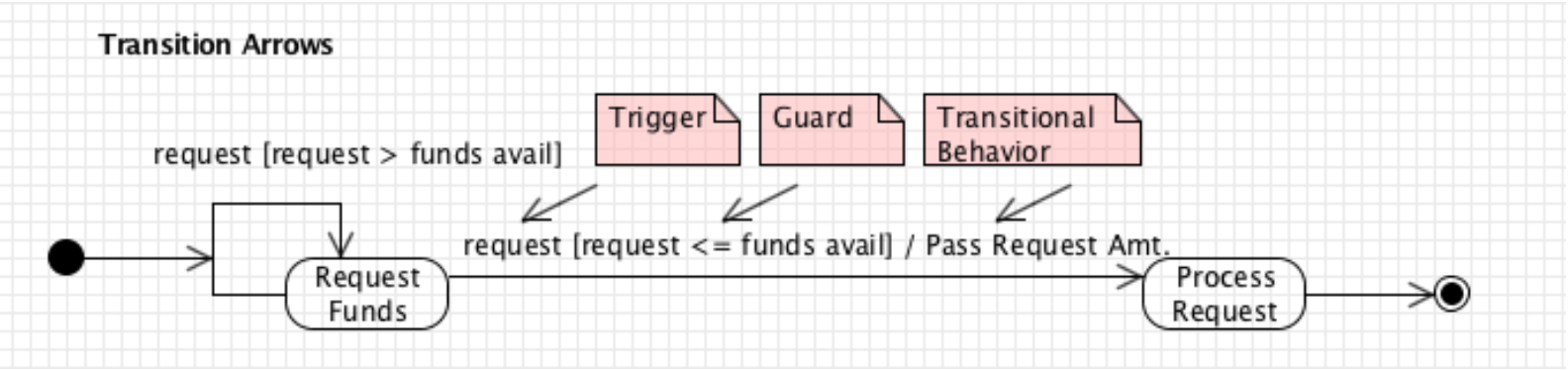
Event / Transition

*Transitions from one state to another*

*Trigger-signature – trigger to change the state*

*[guard] – Boolean Condition for transition to occur*

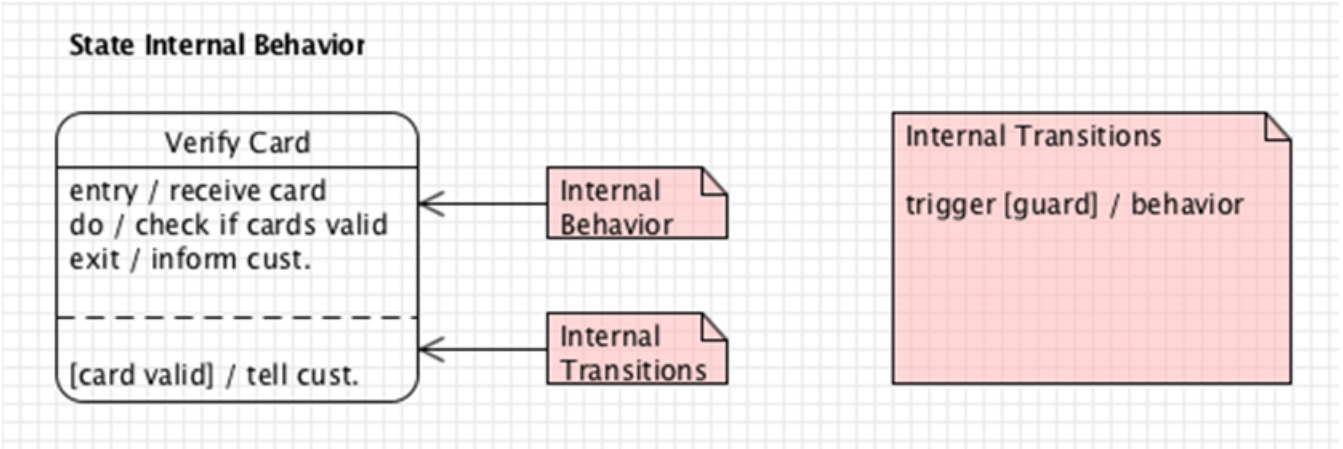
*Activity – Event that happens during transition*



# UML State Machines: Advanced Symbols

State Internal Behavior

- Internal Behavior
  - *Entry Behavior / Exit Behavior*
  - *Internal Events / Do Behaviors*
- Internal Transitions
  - *Transition Label*



# UML State Machines: Advanced Symbols

Concurrent States

*Fork / Join*

*Concurrent Boundary*

Choice Pseudostates
(Boolean decisions)