UML – Unified Modelling Language

Ohjelmankehityspr., versionhallinta ja testaus – Chapter 6

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Deepak K.C. ; deepak.kc@kamk.fi

What is UML?

- General purpose visual modeling language for systems
- Most often associated with modeling object oriented software systems
- UML diagrams
  - Human readable and easily rendered by computers
- A process rather than a development method to make a successful system
- UML diagrams for : business users, developers, common people and anyone who wants to understand the system

UML structure

- The structure consists of
  - Building blocks
    - Basic UML modeling elements, relationships & diagrams
  - Common mechanisms
    - UML ways to achieve specific goals
  - Architecture
    - The view of system architecture

UML building blocks

- UML consists of three building blocks
  - Things
    - Most important building blocks of UML
    - Things can be structural, behavioral, grouping or annotational
  - Relationships
    - Shows how elements are associated with each other
    - Association also describes the functionality of an application
  - Diagrams
    - The ultimate output
    - Elements and relationships are used to make a complete UML diagram
    - Most important part of the entire process

Things

- Structural things
- Behavioral things
- Grouping things
- Annotational things

Structural things

- Nouns of a UML model like class, interface, collaboration, use case, active class
- Define the static part of the model
- Represent physical and conceptual elements
  - Class : set of objects having similar responsibilities
  - Interface: A set of operations specifying the responsibility of a class
  - Collaboration: Defines interaction between elements
  - Use case: A set of actions performed to achieve a specific goal
  - Component: describes physical part of a system
  - Node: physical element that exists at run time

Behavioral things

- Verbs of a UML model
  - Such as interactions, activities, state machines
- Dynamic parts of UML models
  - Interaction
    - A behavior that consists of a group of messages exchanged among elements to accomplish a specific task
  - State machine
    - It is useful when the state of an object is important
    - Defines the sequence of states an object goes through in response to events
    - Events are external factors that cause the change in state

Grouping things

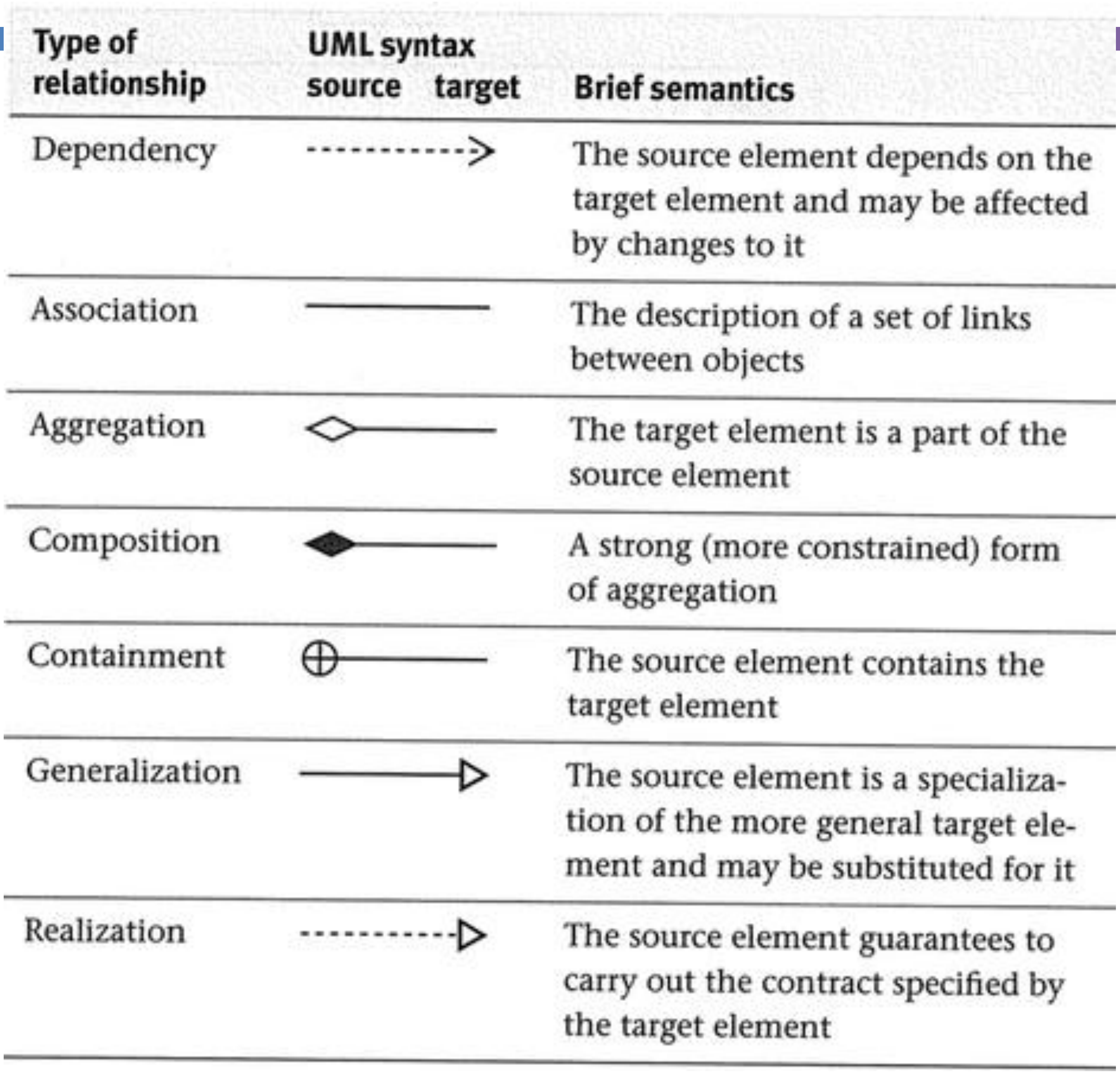
- The package that is used to group semantically related modeling elements into cohesive units
- Package: to gather thing available into structural and behavioral things

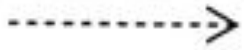
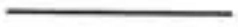
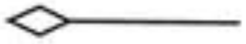
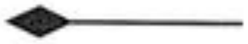
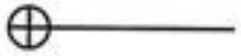
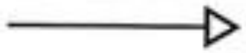
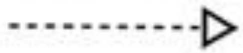
Annotational things

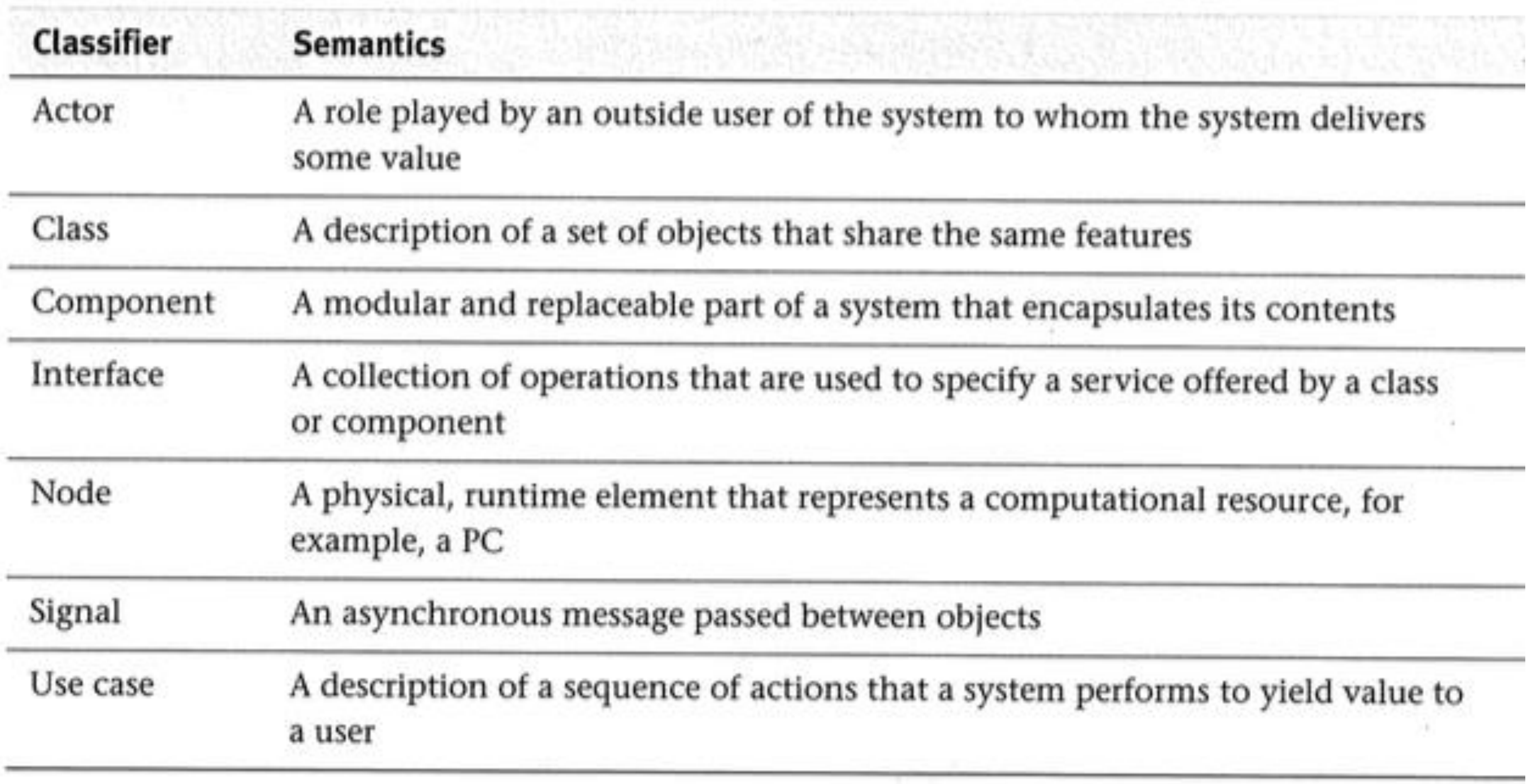
- Mechanism to capture descriptions, comments and remarks of UML model element
- Note is used to render constraints, comments of an UML element
- Very much like a yellow sticky note

Relationship

- Important aspect of building block of UML
- Shows how elements are associated with each other
- Association also describes the functionality of an application



Type of relationship	UML syntax source target	Brief semantics
Dependency		The source element depends on the target element and may be affected by changes to it
Association		The description of a set of links between objects
Aggregation		The target element is a part of the source element
Composition		A strong (more constrained) form of aggregation
Containment		The source element contains the target element
Generalization		The source element is a specialization of the more general target element and may be substituted for it
Realization		The source element guarantees to carry out the contract specified by the target element



Classifier	Semantics
Actor	A role played by an outside user of the system to whom the system delivers some value
Class	A description of a set of objects that share the same features
Component	A modular and replaceable part of a system that encapsulates its contents
Interface	A collection of operations that are used to specify a service offered by a class or component
Node	A physical, runtime element that represents a computational resource, for example, a PC
Signal	An asynchronous message passed between objects
Use case	A description of a sequence of actions that a system performs to yield value to a user

UML Diagrams

- Class Diagram
- Object Diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

UML architecture

- Visualizing a system from different perspectives
  - Design
    - Consists of classes, interfaces & collaboration
    - Class Diagrams & Object Diagram
  - Implementation
    - Defines how a complete system is made by assembling different components
    - Component diagram
  - Process
    - Defines the flow of the system
    - Elements used in Design also support this perspective
  - Deployment
    - Represents physical nodes of the system that forms the hardware
    - Deployment diagram

Modeling types

- Structural modeling
  - Static features of a system
    - Classes diagram
    - Objects diagram
    - Deployment diagrams
    - Package diagrams
    - Composite structure diagram
    - Component diagram

- Behavioral modeling
  - Describes the interaction in the system
  - Dynamic natures of the system
    - Activity diagrams
    - Interaction diagrams
    - Use case diagrams

Architectural modeling

- Represents the overall framework of the system
- Contains both structural & behavioral elements of the system
- Blue print of the entire system
- Package diagram is used for architectural modeling

UML standard diagrams

- Structural diagrams
  - Class diagram
  - Object diagram
  - Component diagram
  - Deployment diagram

- Behavioral diagrams
  - Use case diagram
  - Sequence diagram
  - Collaboration diagram
  - Statechart diagram
  - Activity diagram

Use Case Diagrams

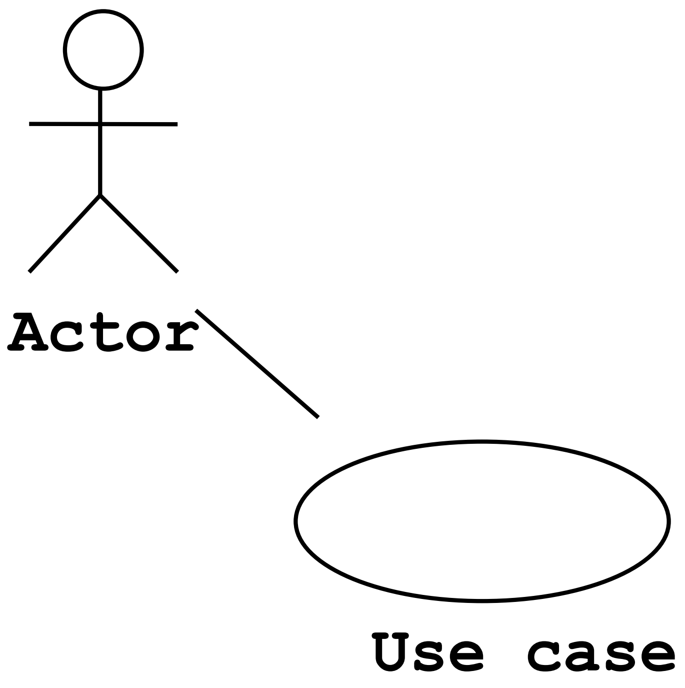
Deepak K.C. ; deepak.kc@kamk.fi ;2015

Use case diagram

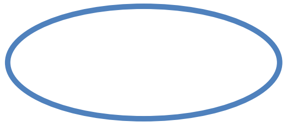
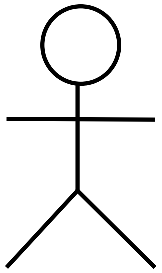
- Set of cases, actors & their relationships
- Explains the outside view of a system
- Identifies internal & external factors influencing the system
- Describes set of actions (use cases) and external users of the system (actors)

Drawing use case diagrams

- Actors
  - Something or someone that interacts with the system
  - Can be human user or internal or external application
- Functionalities
  - Represented as an use case
- Relationship
  - How are use cases related to each actor?

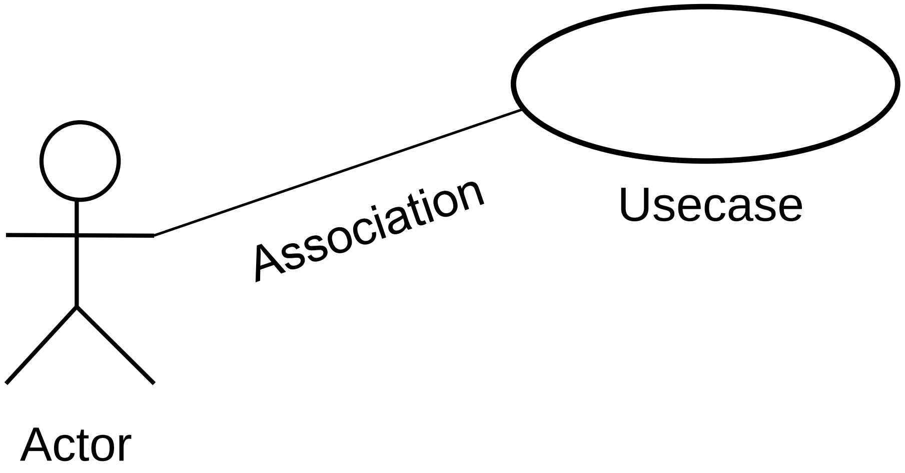


Notations

- Use case 
  - Description of a set of sequences of actions including variants that system performs
  - *"A use case is the specification of a set of actions performed by a system, which yields an observable result that is typically of value for one or more actors or other stakeholders of the system."*
- Actor 
  - Something or someone that interacts with the system
  - Can be human user or internal or external application
  - *"An actor specifies a role played by a user or any other system that interacts with the subject."*

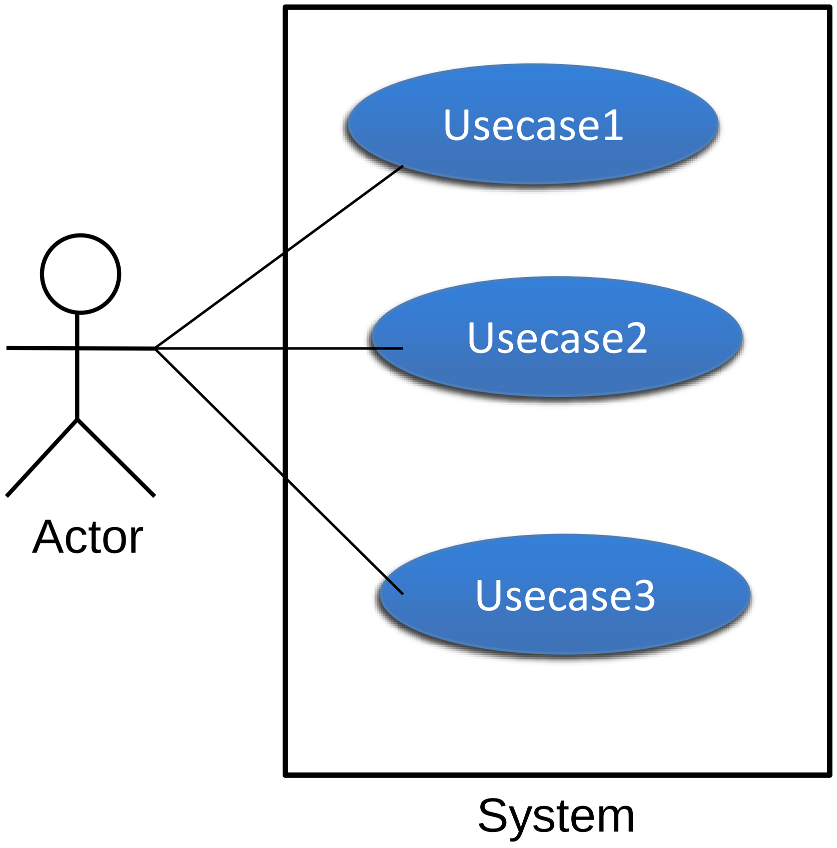
Notations

- Association
  - To indicate that the actor participates in a use case
  - *"An association specifies a semantic relationship that can occur between typed instances. It has at least two ends represented by properties, each of which is connected to the type of the end. More than one end of the association may have the same type."*



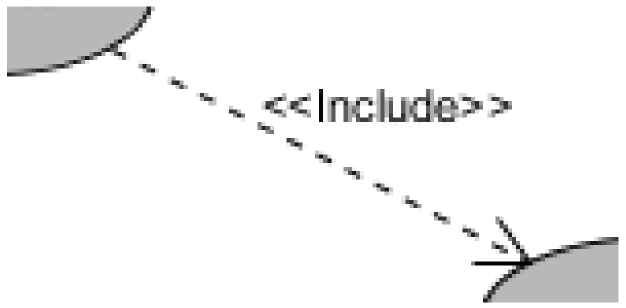
Notations

- System
  - Scope of a system is represented by a rectangular box
  - It also means the system boundary
  - Use case are placed inside the rectangular box
  - *"If a subject (or system boundary) is displayed, the use case ellipse is visually located inside the system boundary rectangle. Note that this does not necessarily mean that the subject classifier owns the contained use cases, but merely that the use case applies to that classifier."*



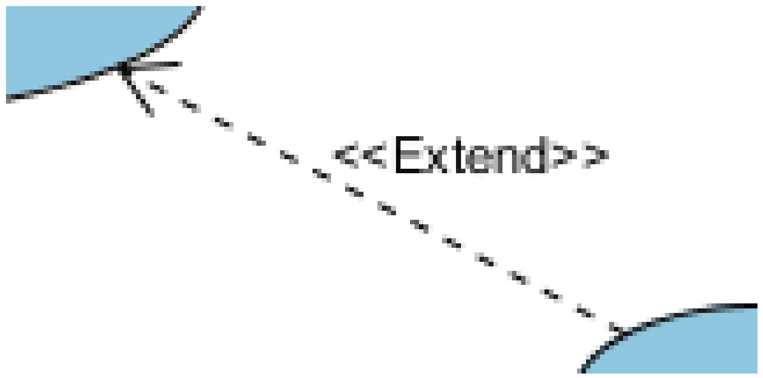
Notations: Include

- A base use case is dependent on the included use case(s)
- To specify that the source use case explicitly incorporates the behavior of another use case
- To simplify large uses cases by dividing into several use cases
- To extract common parts of the behaviors of two or more use cases
- *"An include relationship defines that a use case contains the behavior defined in another use case."*



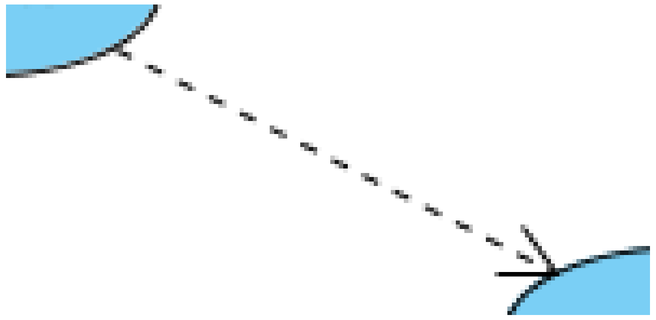
Notations: Exclude

- Extending use case is dependent on the base use case
- Literally extends the behavior described by the base use case
- *“A relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case.”*



Notation: Dependency

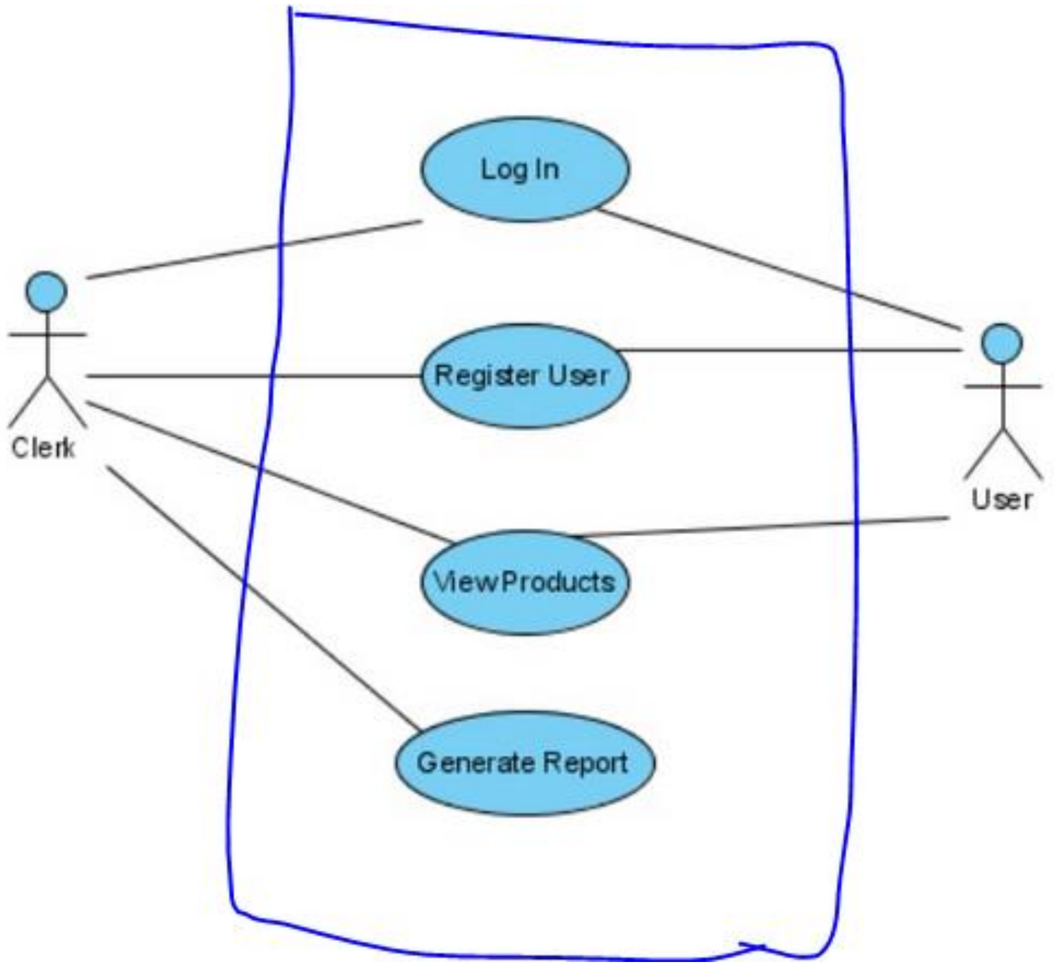
- A model element relies on another model element for implementation
- *"A dependency is a relationship that signifies that a single or a set of model elements requires other model elements for their specification or implementation. This means that the complete semantics of the depending elements is either semantically or structurally dependent on the definition of the supplier element(s)."*



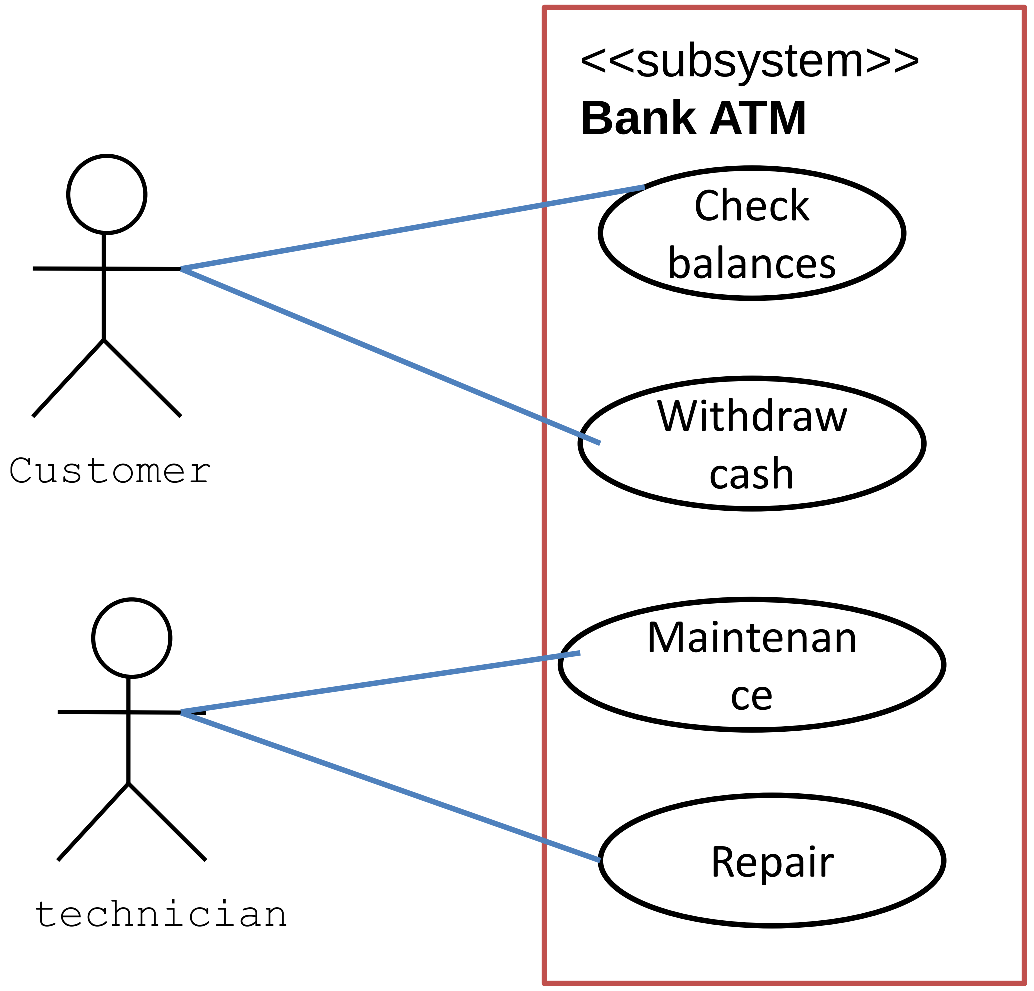
Guidelines for use case diagrams

- Begin use cases with a verb like register new user, generate report
- Use case name should be very distinct and properly identifies the functionalities performed
- Use suitable name for actors
- Show dependencies and relationships explicitly in the diagram
- Do not include all types of relationships as the main purpose is to identify requirements
- Use note whenever need to clarify important points
- You can avoid include and exclude instead focus on showing user centered functionality

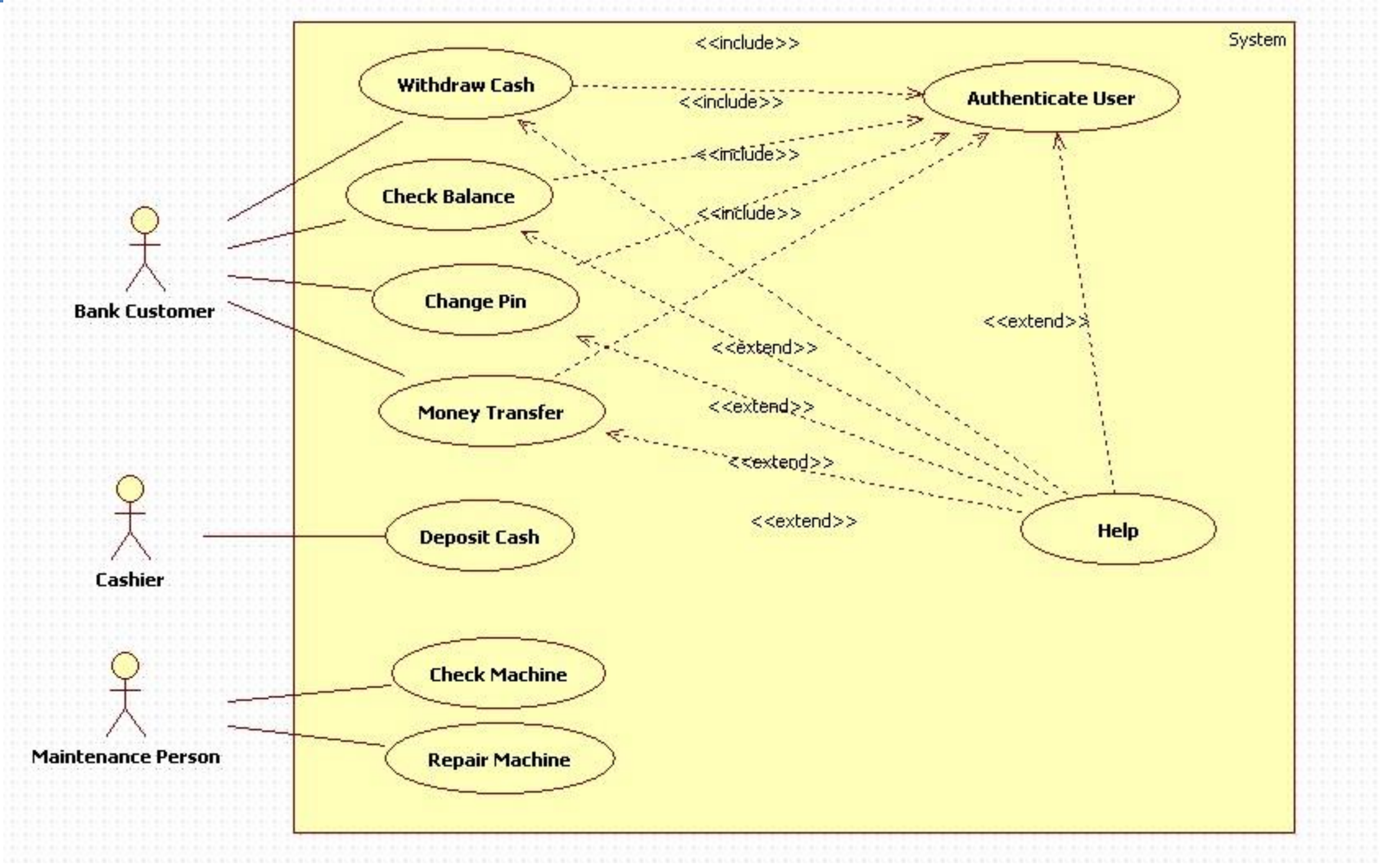
Usecase diagram: Example



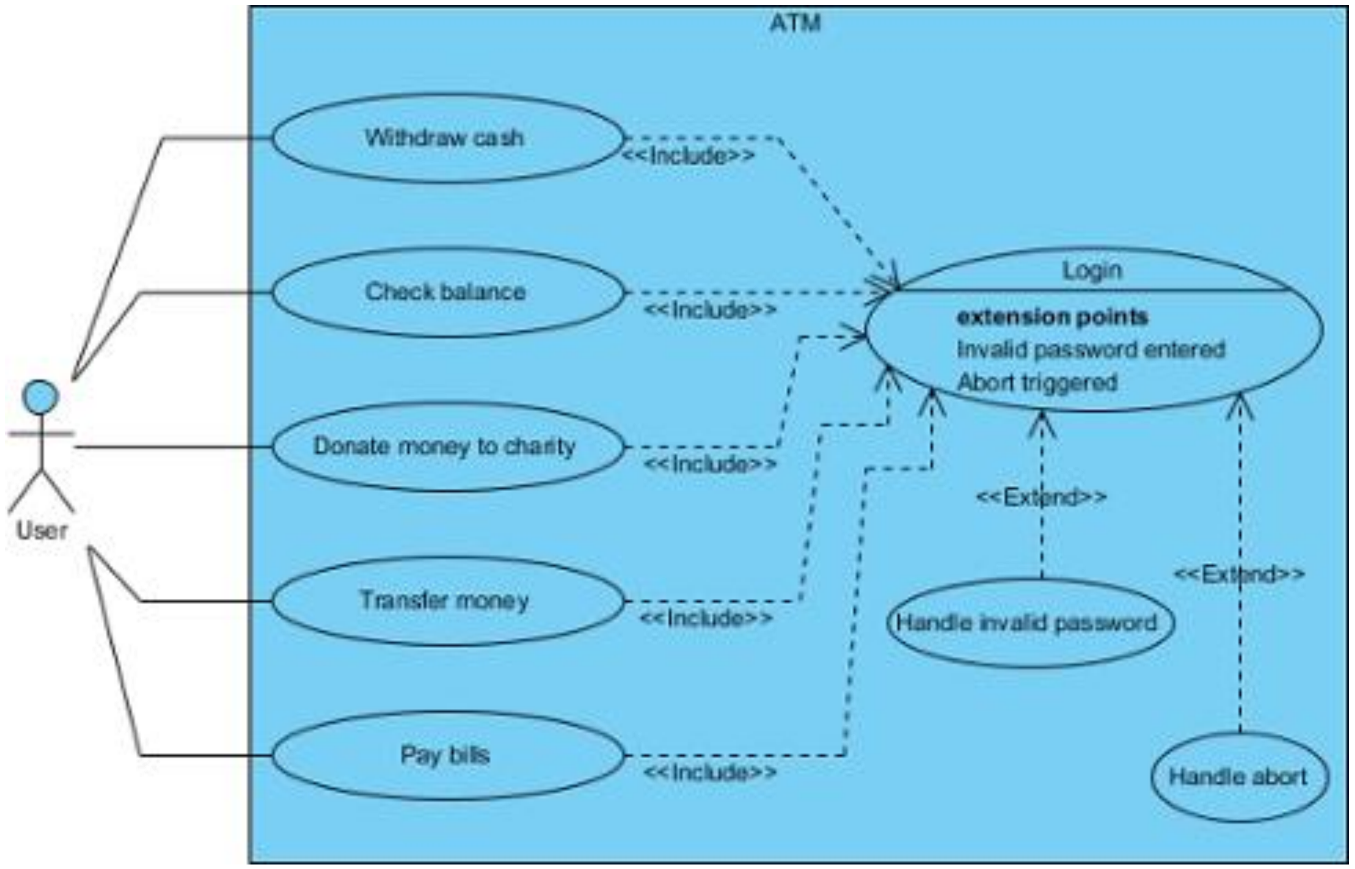
Usecase diagram: Example



Usecase diagram: ATM machine

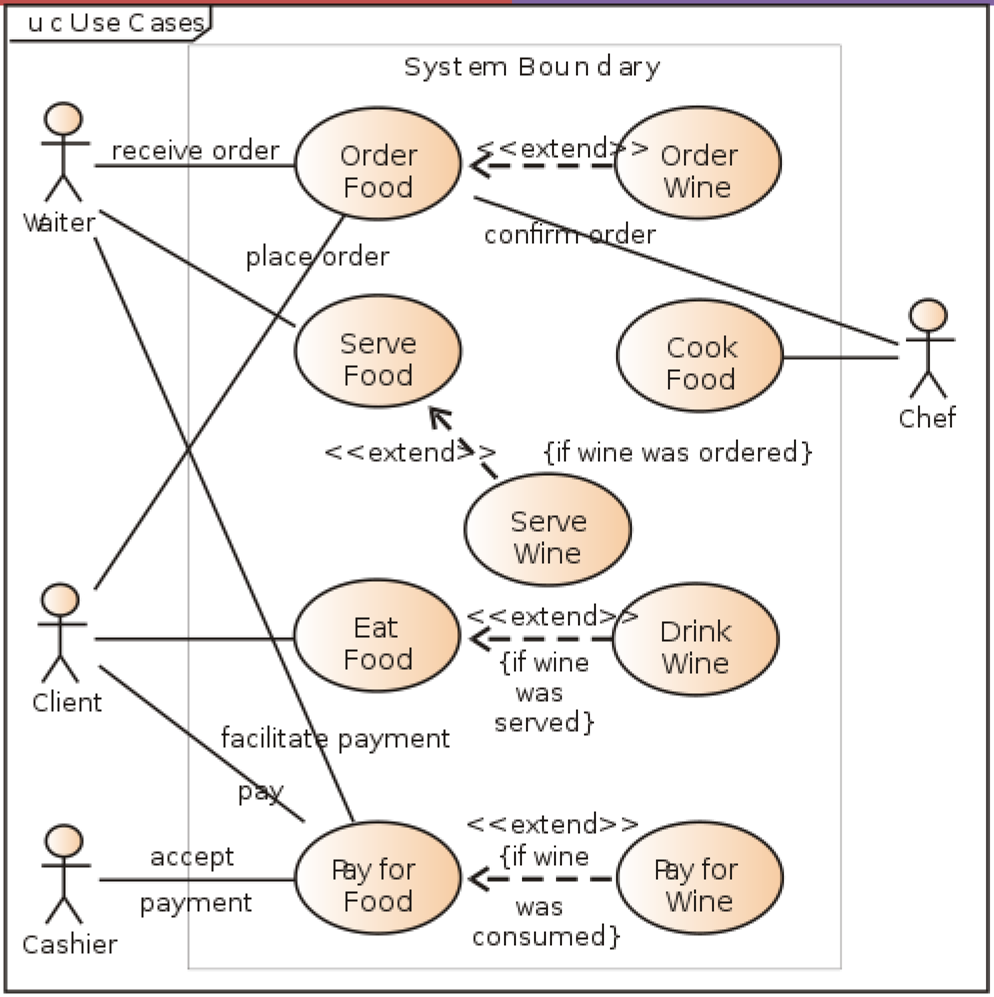


Usecase diagram example



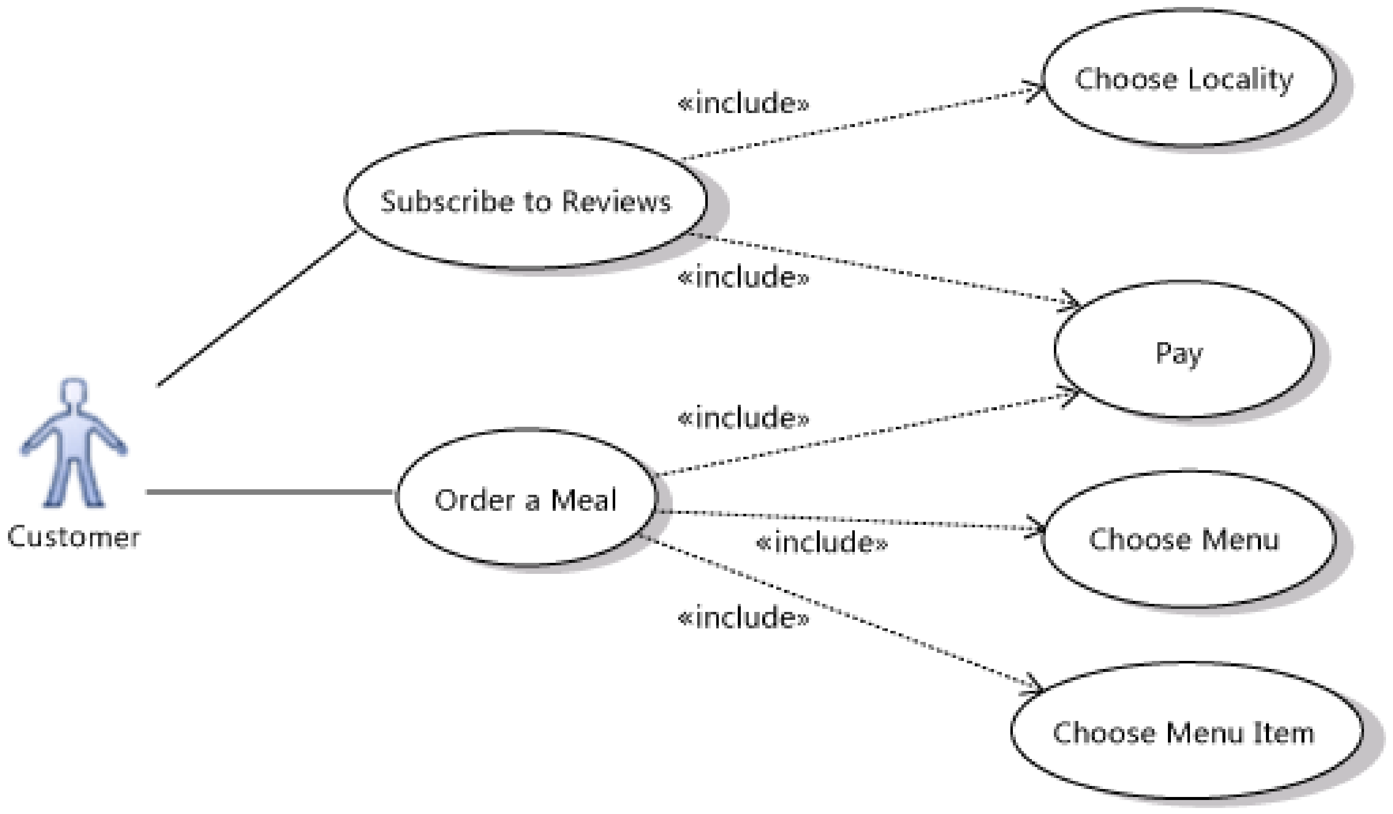
Img src:http://images.visual-paradigm.com/docs/vp_user_guide/11/94/2575/2745/sample_use_case_diagram_19967.png

Usecase diagram: Example



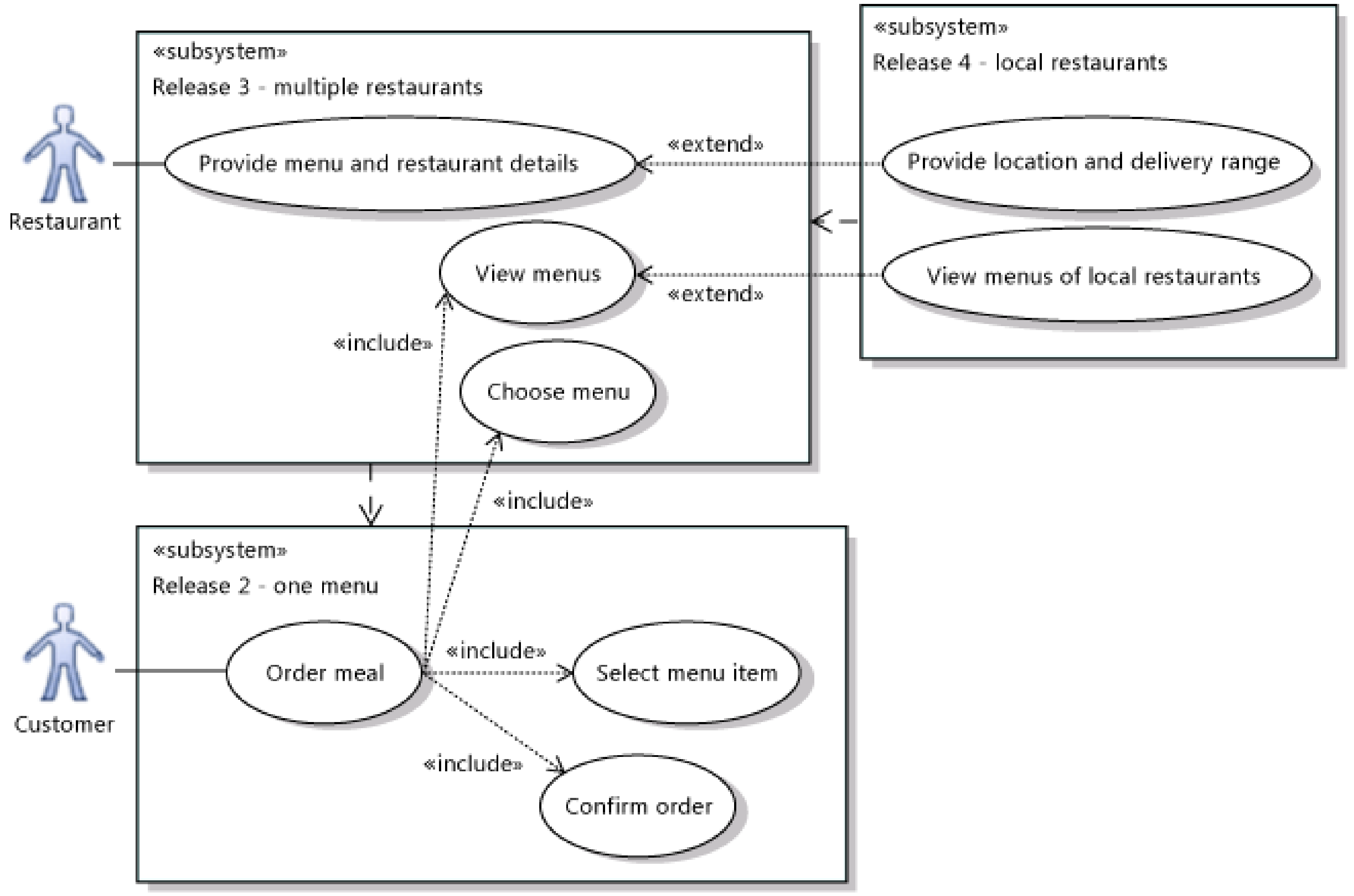
Src:wikipedia

Usecase diagram: Example



SRC:https://msdn.microsoft.com/en-us/library/dd409432.aspx#Actors

Usecase diagram: Example



Exercise

- Draw a use case diagram for the scenario below
  - Grade system
  - Teachers should login to upload grades for the courses they teach. Students can login to see their grades and also for the failed courses, they should be able to apply for the re-exam.

Interaction Diagrams

Deepak K.C. ; deepak.kc@kamk.fi ;2015

Interactions Diagrams

- To explain interactions among different elements in the model
- Purpose: to visualize the interactive behavior of the system
- Part of the dynamic behavior of the system
- Interaction behavior represented by two diagrams
  - Sequence diagram
  - Collaboration diagram

Drawing interaction diagrams

- Identify the following things
  - Objects (part of interaction)
  - Message (flowing among the objects)
  - Sequence (the order of the flow of message)
  - Organization of the object

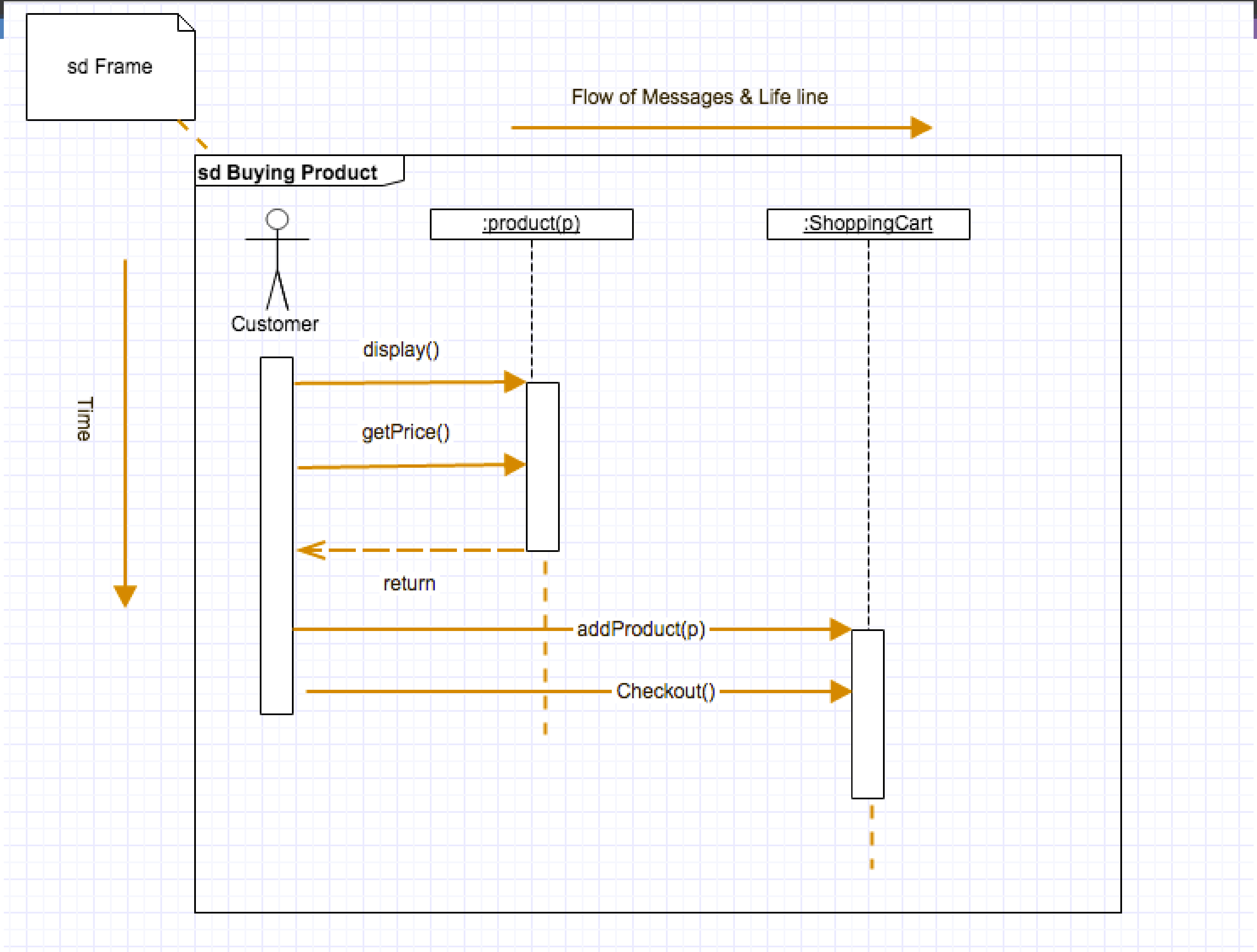
Sequence diagrams

- They are interaction diagrams and explain how operations are carried out
- Model runtime interactions between different parts of a system
- Shows how objects interact with one another
- Shows the order of the interaction
  - X axis
    - Represents objects
    - Object initiating interaction is left most
  - Y axis
    - Represents time
    - Messages sent and received are ordered by time
- Shows simple iteration and branching

How to draw sequence diagram?

- For the selected scenario, identify object that are involved in the scenario
- List object in order at the top as the order of their use
- Draw dotted lines down to indicate lifelines
- Triggering event (identify the event that initiates the whole process)
- Draw horizontal arrows from the object that sends message to the receiving object
- Identify the next event and the object involved
- Continue last two steps until the diagram is complete

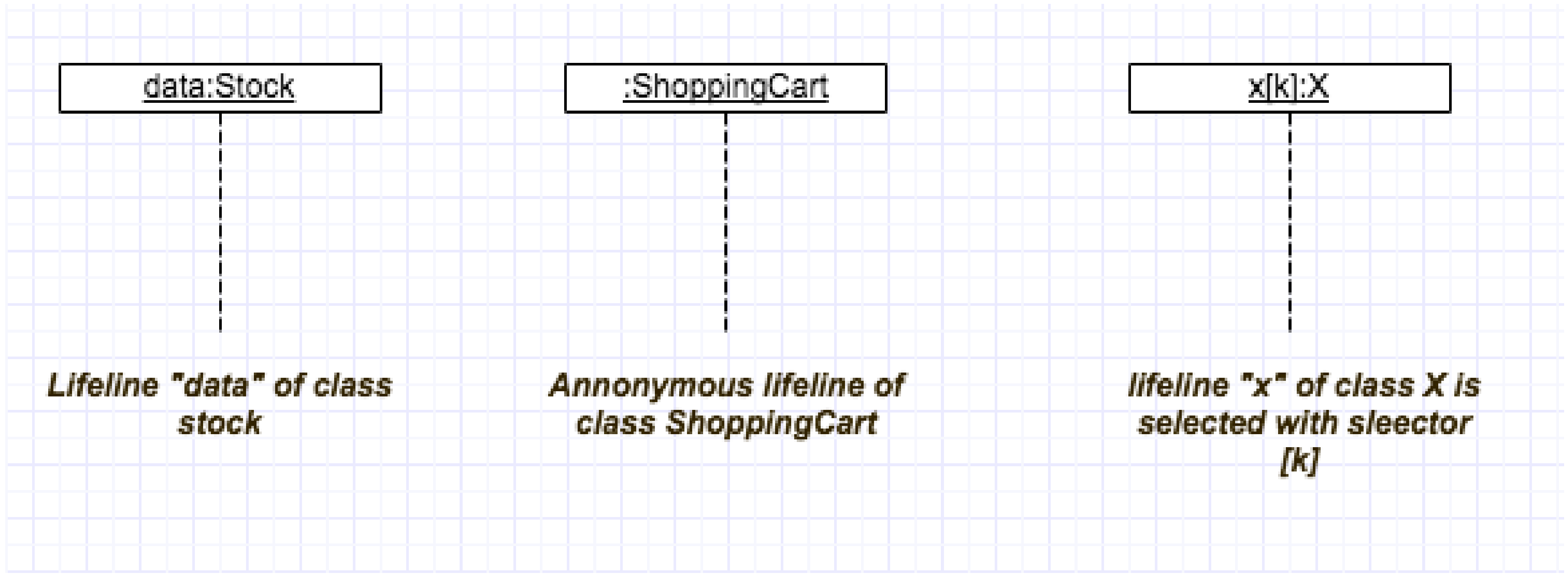
Sequence Diagram: Example



Lifelines

- Represent either roles or object instances participating in the sequence diagram
- Notation for lifeline are placed at the top
- Name of the lifeline is placed inside the box
  - Instance Name : Class Name
- Vertical dotted line is a lifeline
  - It represents the time that an object exists

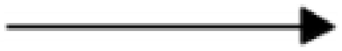
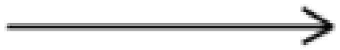
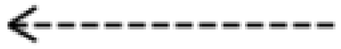
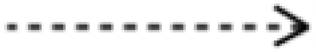
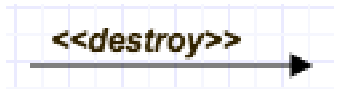
Lifelines


data:Stock
:ShoppingCart
x[k]:X
Lifeline "data" of class stock
Annonymous lifeline of class ShoppingCart
lifeline "x" of class X is selected with sleector [k]

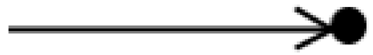
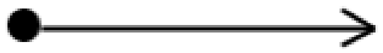
Message

- Message defines specific kind of communication between lifelines
- First message starts at top and is located on the left side of the diagram
- Subsequent messages are then added slightly lower than the previous message

Message type notations -I

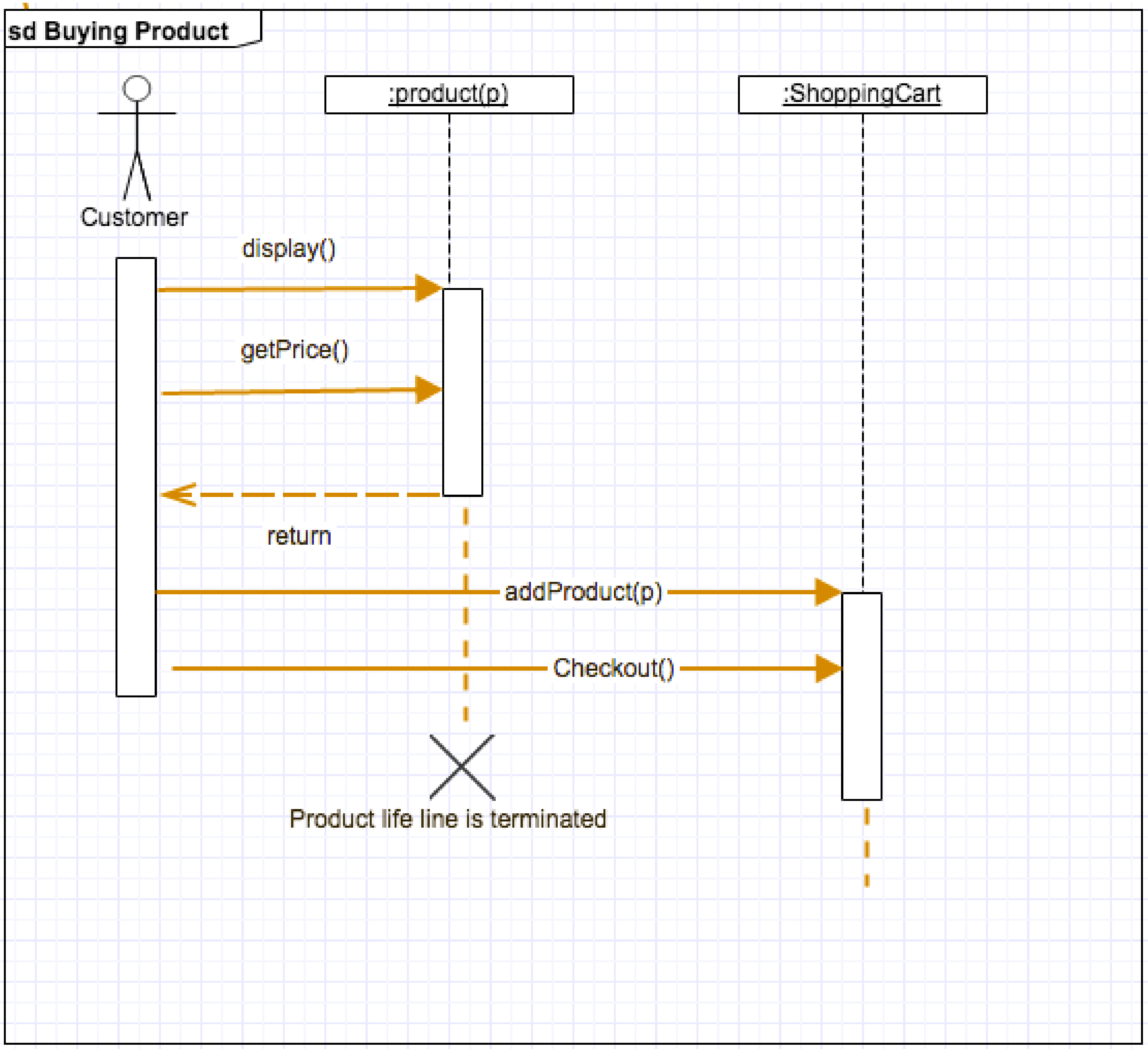
Type of Message	Description	Notation
Synchronous call	Sends message and suspend execution while waiting for response	
Asynchronous call	Sends message & proceeds immediately without waiting for return value No explicit return message to the caller	
Reply	Response or return message from another message	
Create	Creates a new object	
Delete	Sent to terminate another lifeline Usually ends with a "X" at the bottom	

Message type notations -II

Type of Message	Description	Notation
Lost	Interpreted as if message never reached its destination Sending event is known but there is no receiving event	
Found	Interpreted as if the origin of the message is outside the scope of the description Receiving event is known but sending event is unknown	

Message occurrence

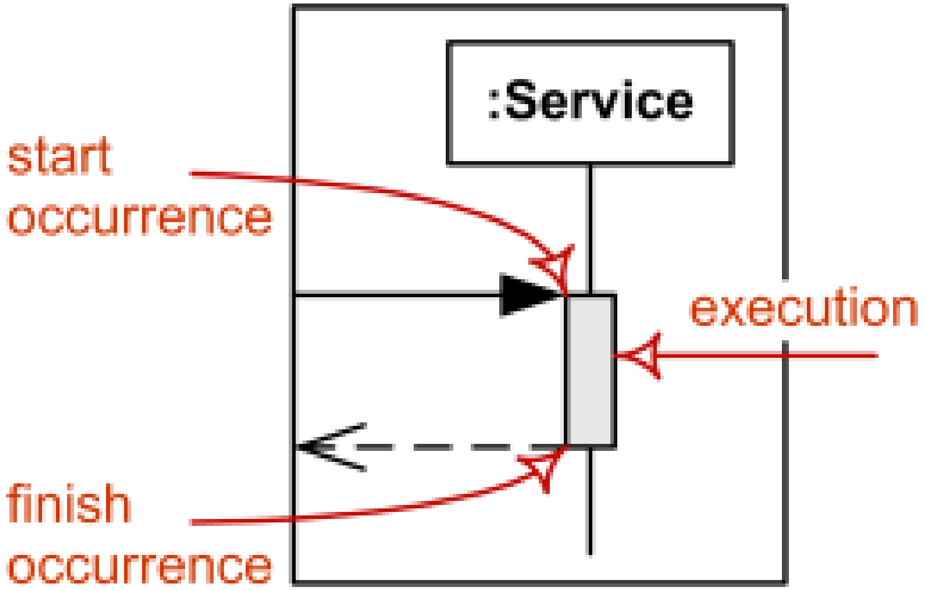
- Destruction occurrence
  - Represents the destruction of the instance described by the lifeline
  - May also result in the subsequent destruction of other objects
  - No other occurrence may appear below the destruction event
  - Depicted by a cross in the form of X at the bottom of lifeline



sd Buying Product
Customer
:product(p)
:ShoppingCart
display()
getPrice()
return
addProduct(p)
Checkout()
Product life line is terminated

Execution occurence

- Represents time at which action start or finish



Img src: http://www.uml-diagrams.org/notation/sequence-execution-spec.png

Combined fragment

- Used to group sets of messages together
  - To show conditional flow in a sequence diagram
- Type of fragment is indicated on the top left corner
- Tpes of fragment
  - Alt : **alternative multiple fragments**: only the one with true condition will execute
  - Opt : **Optiona**l Similar to alt but with only one trace
  - Par: **Parallel,** parallel execution of fragments
  - Loop: **Loop,** fragment may execute multiple times
  - Break, strict, seq, critical, ignore, consider, assert, neg

Read more about combined fragment: http://www.uml-diagrams.org/sequence-diagrams-combined-fragment.html

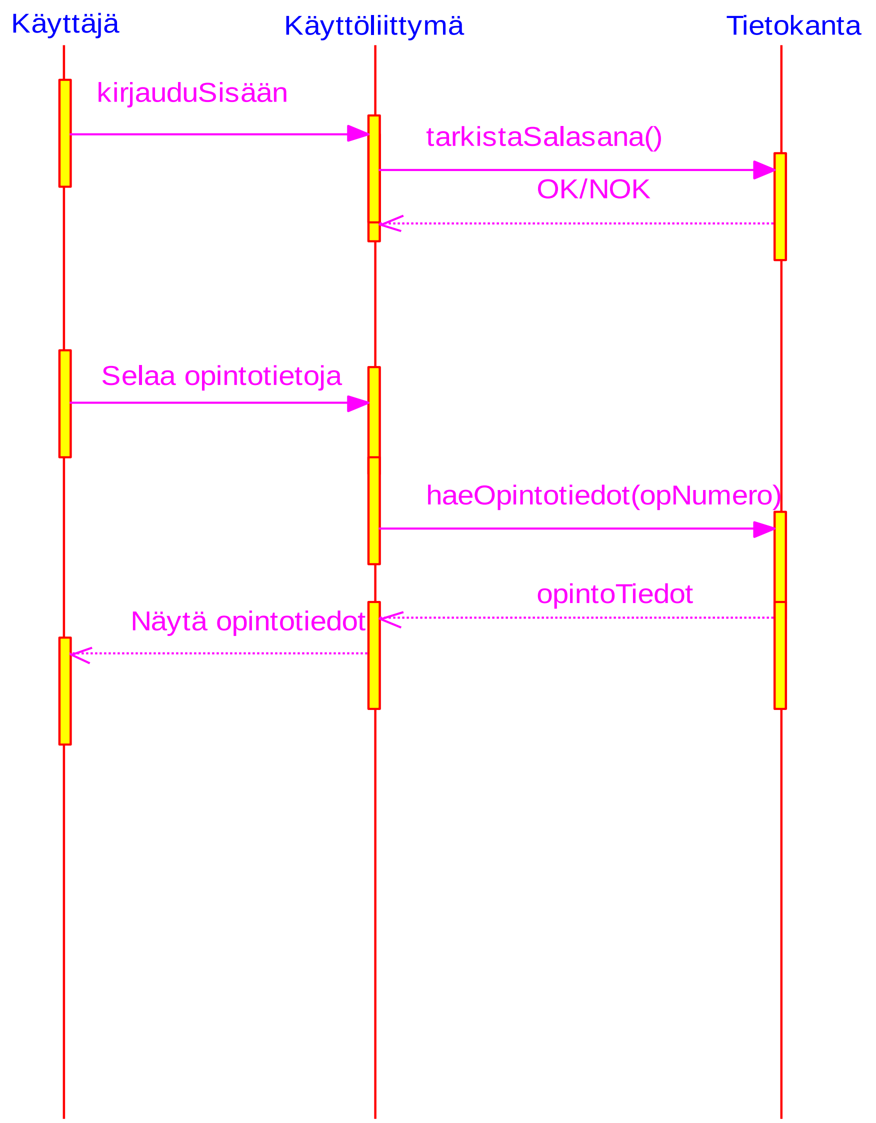


Image Src: Heikkinen Leena <Leena.Heikkinen@kamk.fi>: teaching material

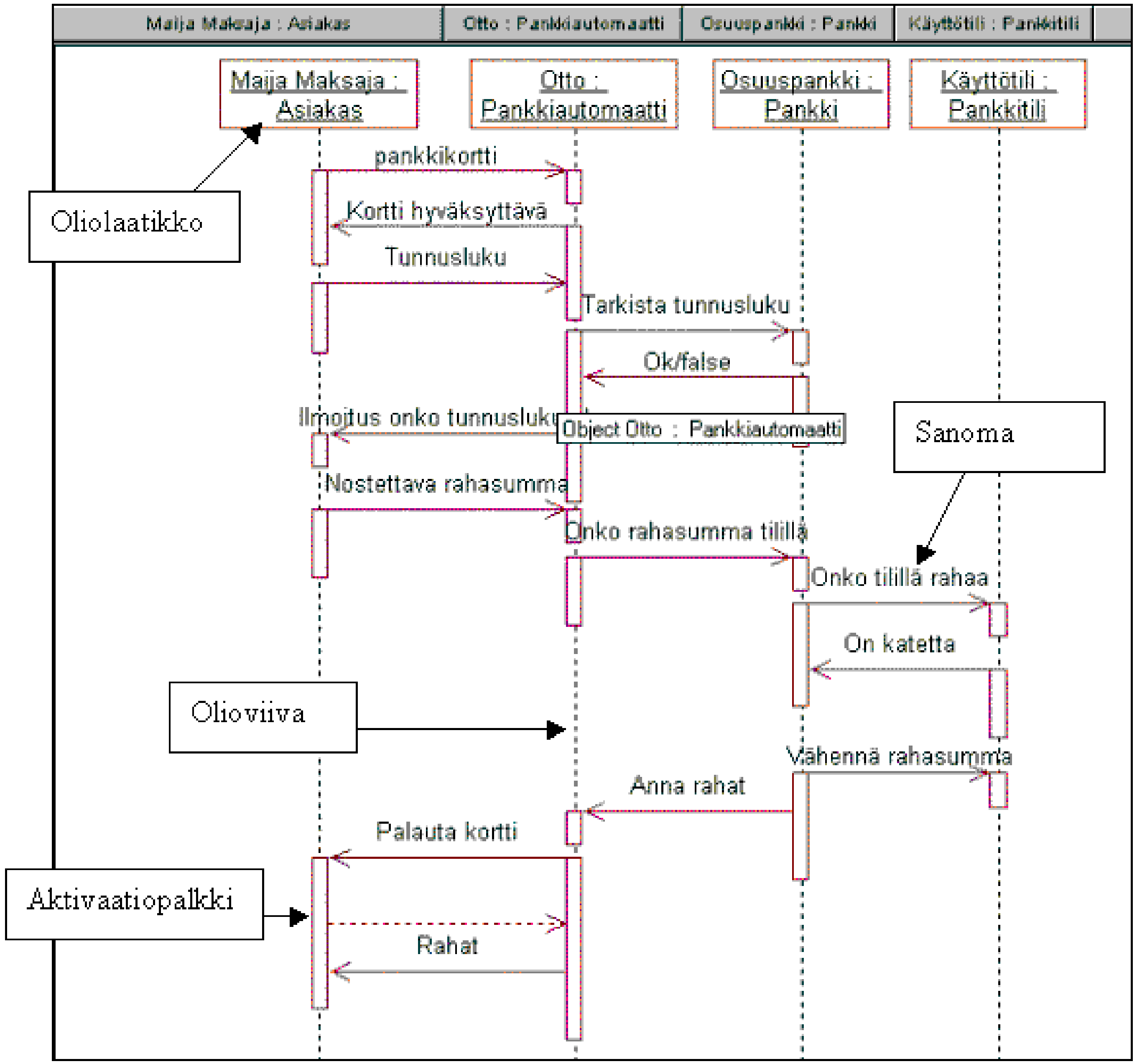


Image Src: Heikkinen Leena <Leena.Heikkinen@kamk.fi>

Esimerkki sekvenssikaavion ehdoista

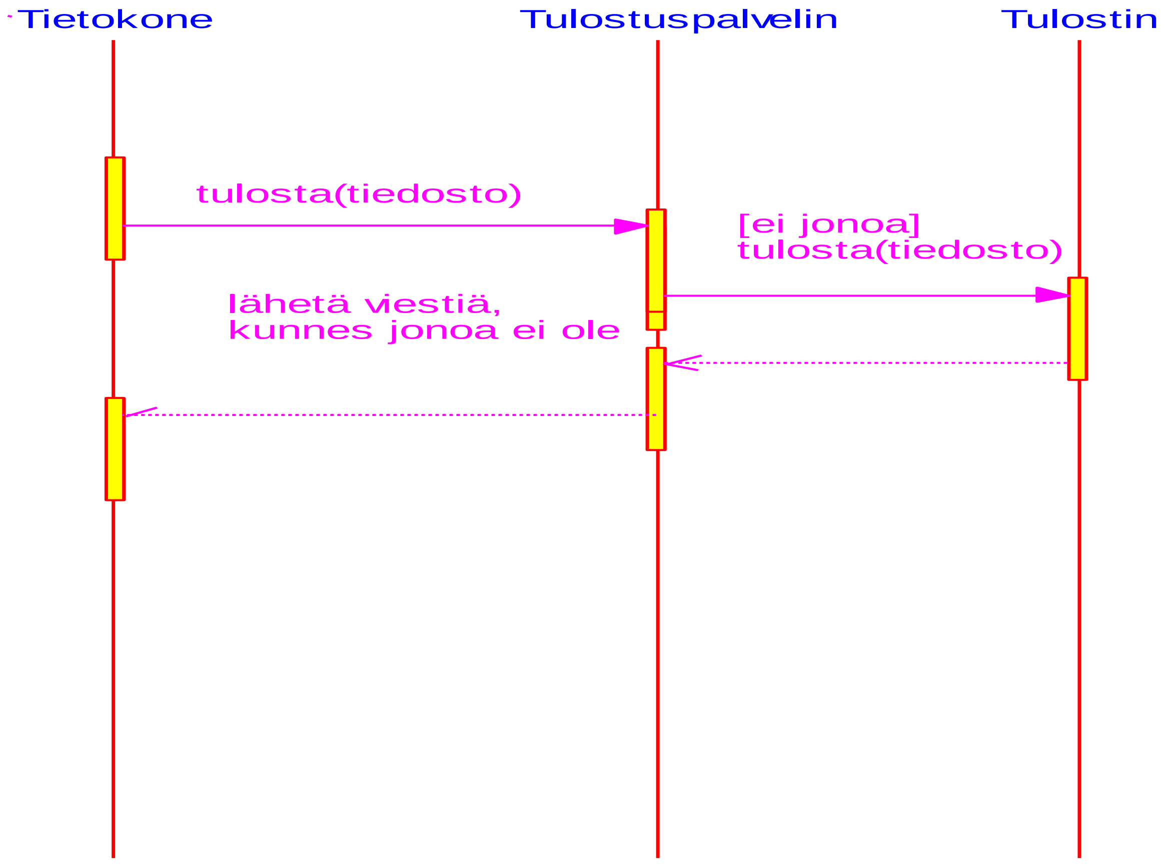
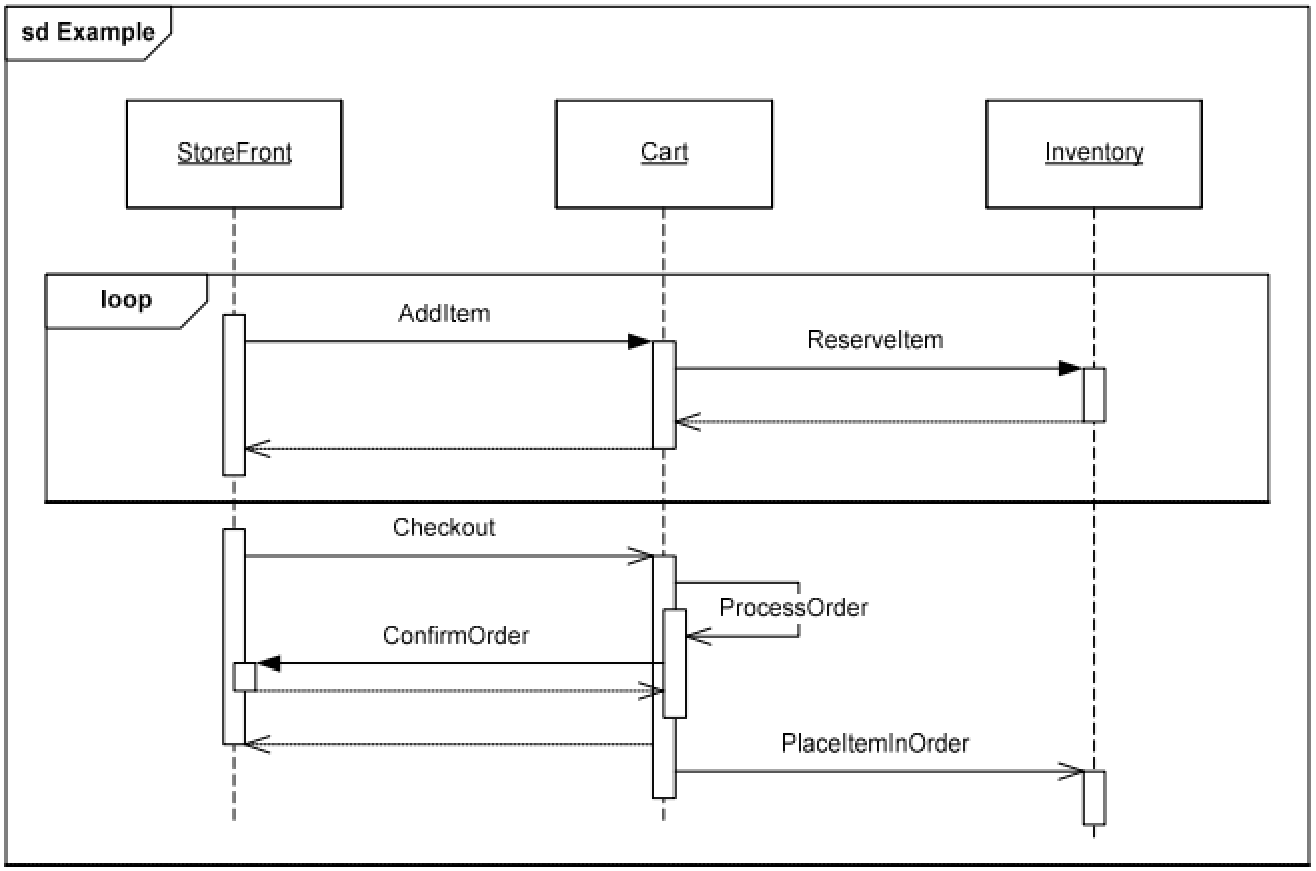


Image Src: Heikkinen Leena <Leena.Heikkinen@kamk.fi>: teaching material



More examples: http://www.uml-diagrams.org/sequence-diagrams-examples.html

Exercise

- Refer to the previous exercise done (usecase- grading system). Now draw the sequence diagram for the grading system..

State Machine Diagram

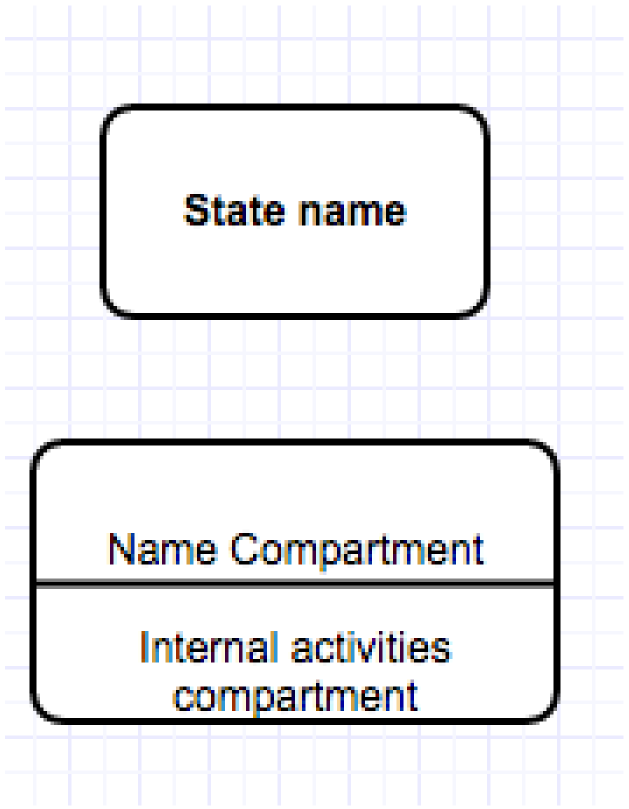
- Illustration of the states an object can attain & the transitions between those states
- A behaviour diagram showing the discrete behaviour of a part of designed system through finite state transitions
- Three main elements
  - States of an object
  - Transitions between states and
  - The events that trigger transitions
- Statechart describes all events & states & transitions for a single object whereas sequence diagram describes events for a single interaction across all object involved

Two kinds of state machines in UML 2.4

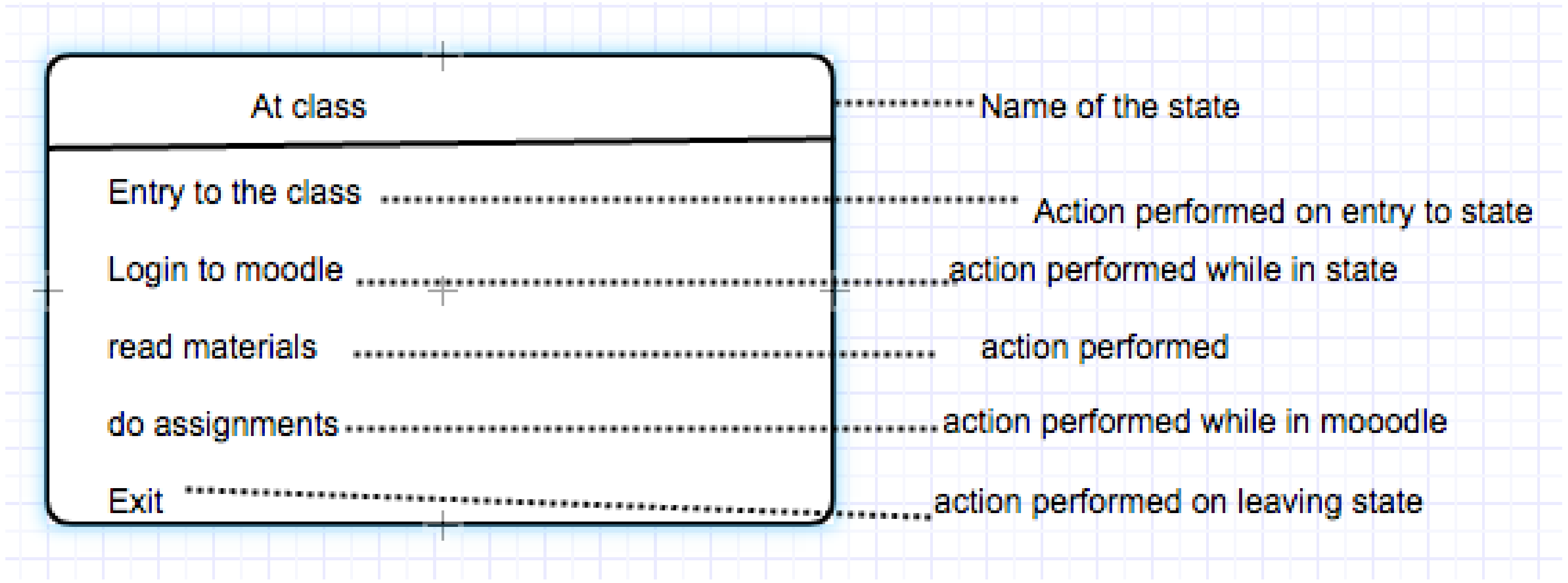
- Behavioral state machine
  - Specialization of behaviour
  - Use to specify discrete behaviour
  - 3 kinds of states
    - Simple state
    - Composite state
    - Submachine state
- Protocol state machine
  - To express usage protocol or a lifecycle
  - Shows what operations of the classifier may be called in each state

Simple state

- A state with no sub states
- Represented by a rectangle with rounded corners
- State name is inside the rectangle
- Name compartment : holds the name of the state
- States without names are called anonymous state
- Internal activities compartment
  - Holds list of internal actions or state (do) activities (behaviors)

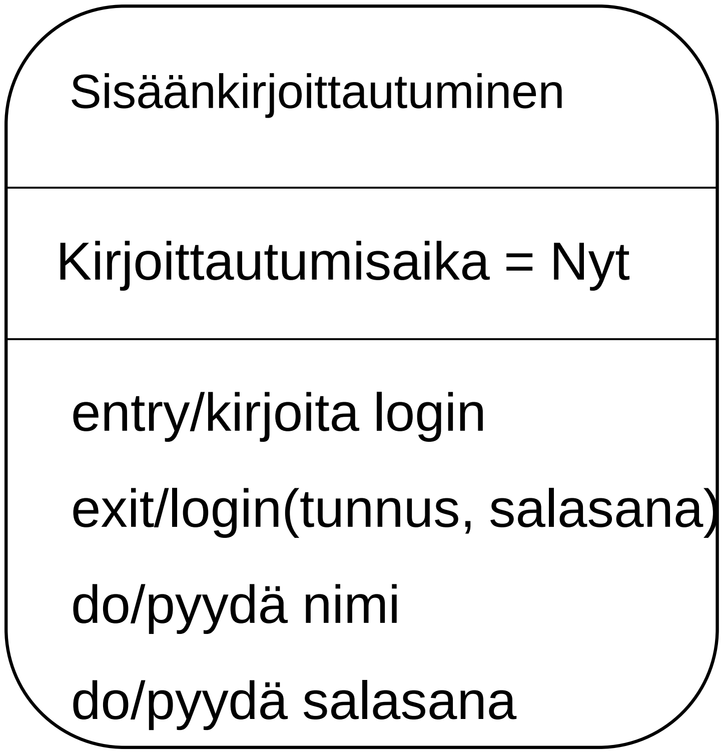


States:Example


At class
Entry to the class
Login to moodle
read materials
do assignments
Exit
Name of the state
Action performed on entry to state
action performed while in state
action performed
action performed while in mooodle
action performed on leaving state

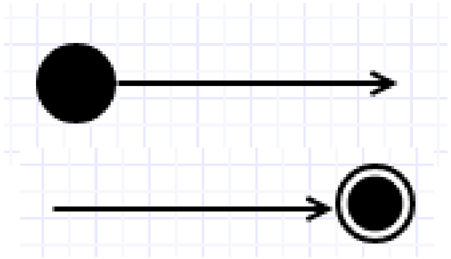
Tilakaavio

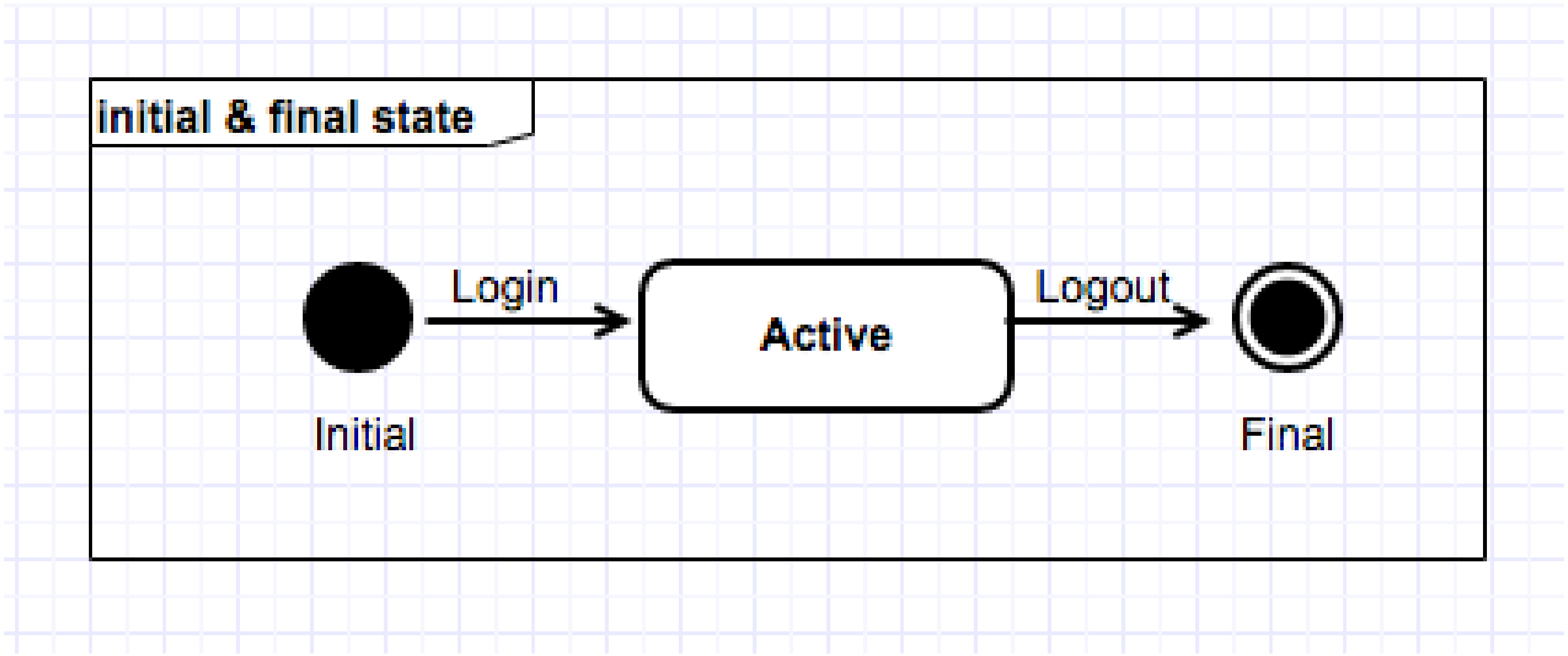
- Tilasymbolin osat
  - Nimi
  - Tilamuuttujat
  - Toiminnot
    - entry/tulotoimet
    - exit/jättötoimet
    - do/aktiviteetti
- Tilojen välillä on tapahtumien aiheuttamia tilasiirtymiä



Initial & Final states

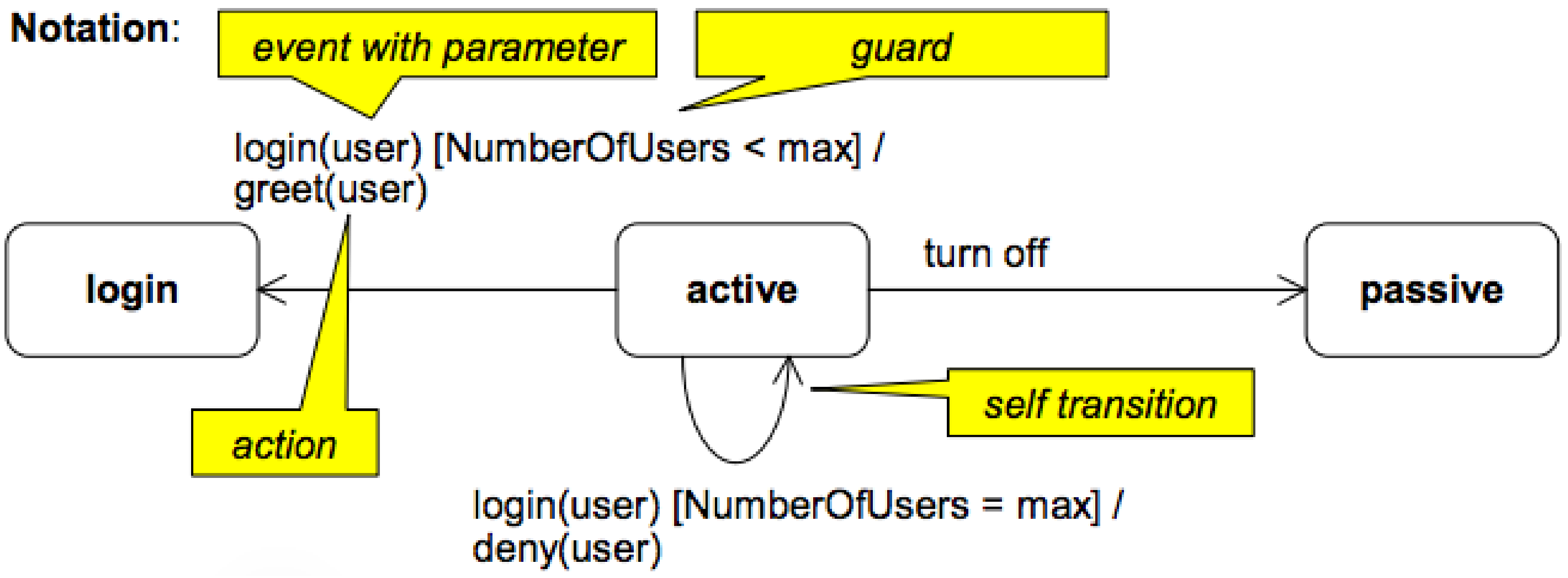
- Initial state
- Final State





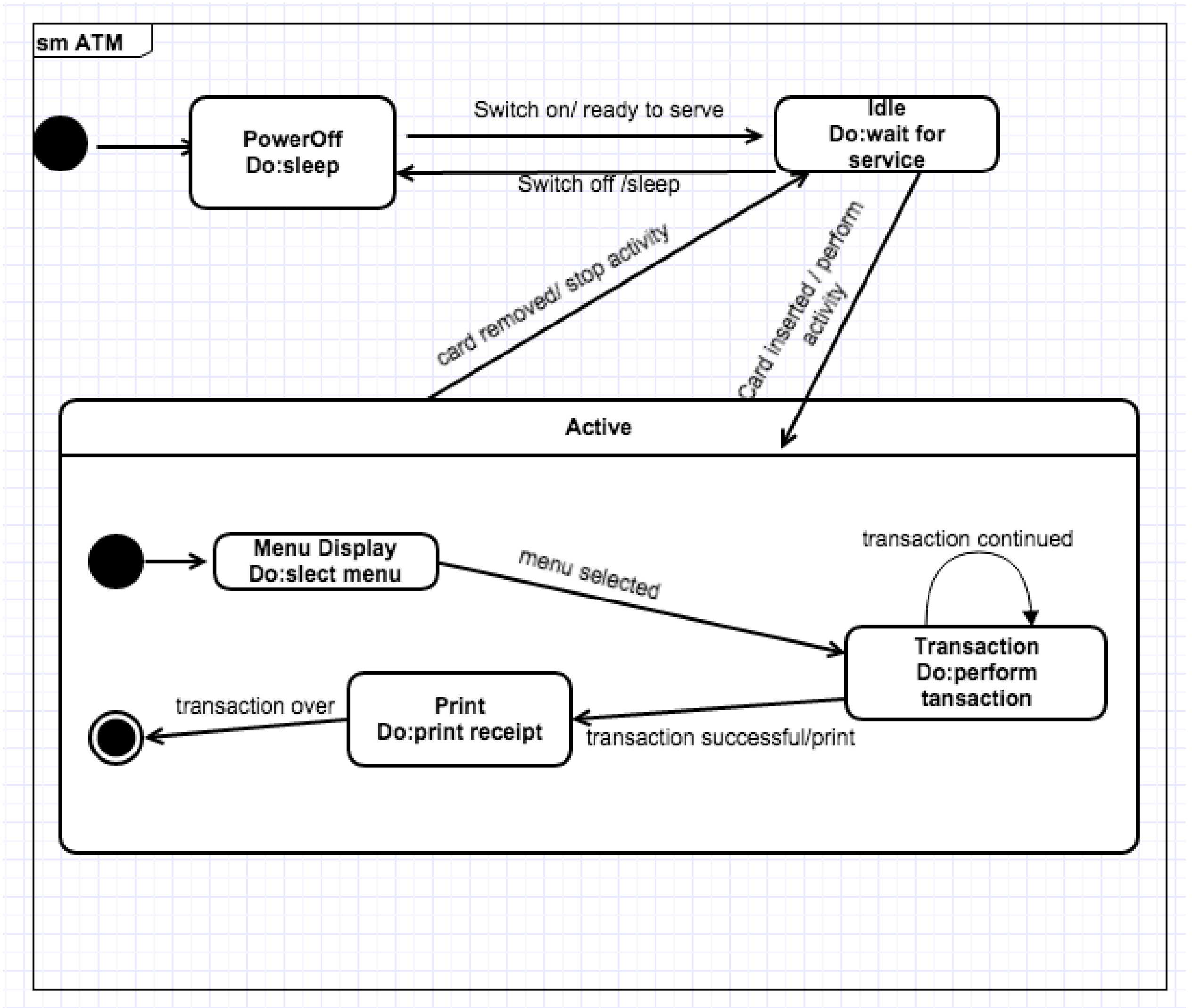
Transitions

- Progression from one state to another
- Depicted by line with arrowheads
- Transition may have a trigger, a guard and an effect

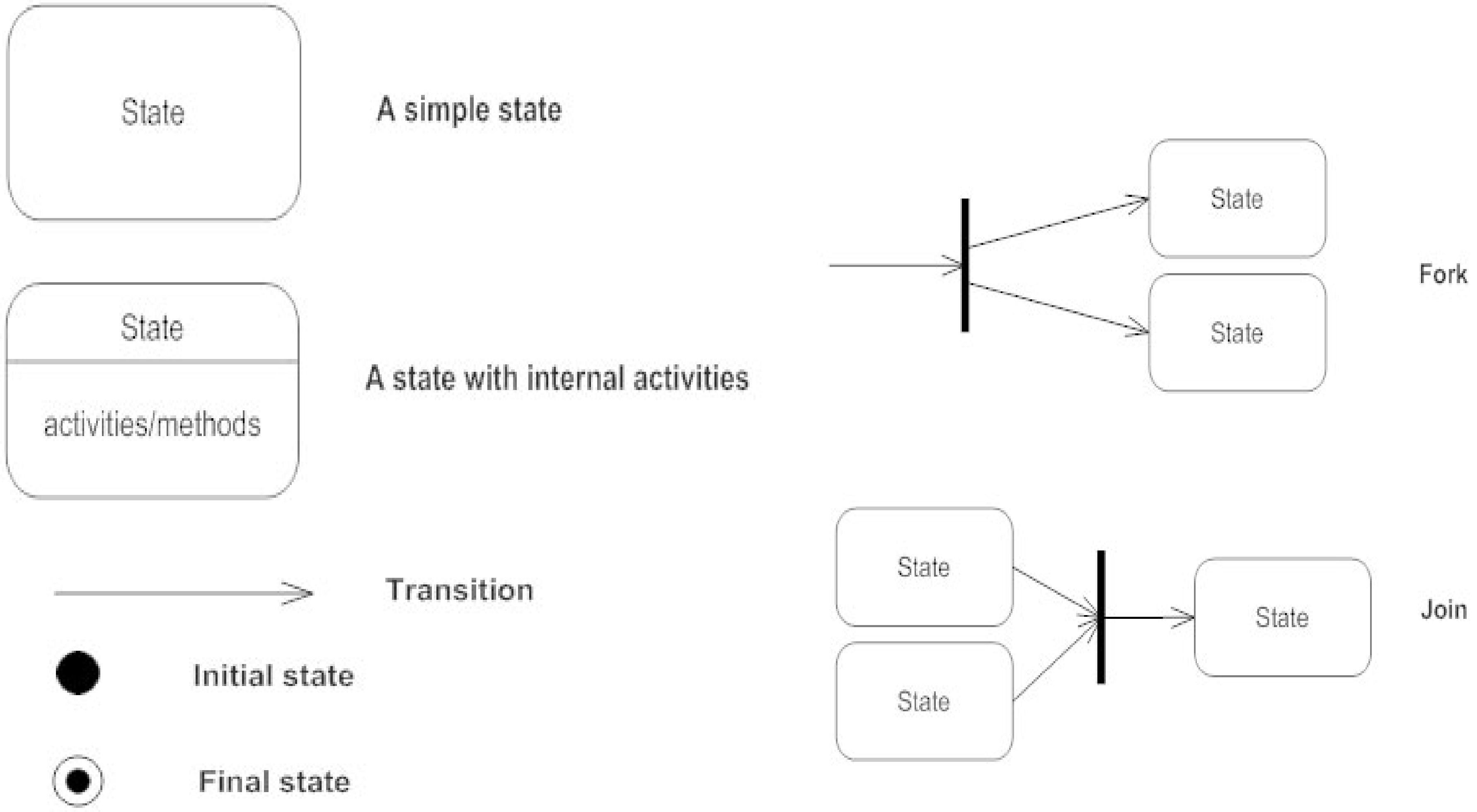

Notation:
event with parameter
guard
login(user) [NumberOfUsers < max] / greet(user)
login
active
turn off
passive
self transition
action
login(user) [NumberOfUsers = max] / deny(user)

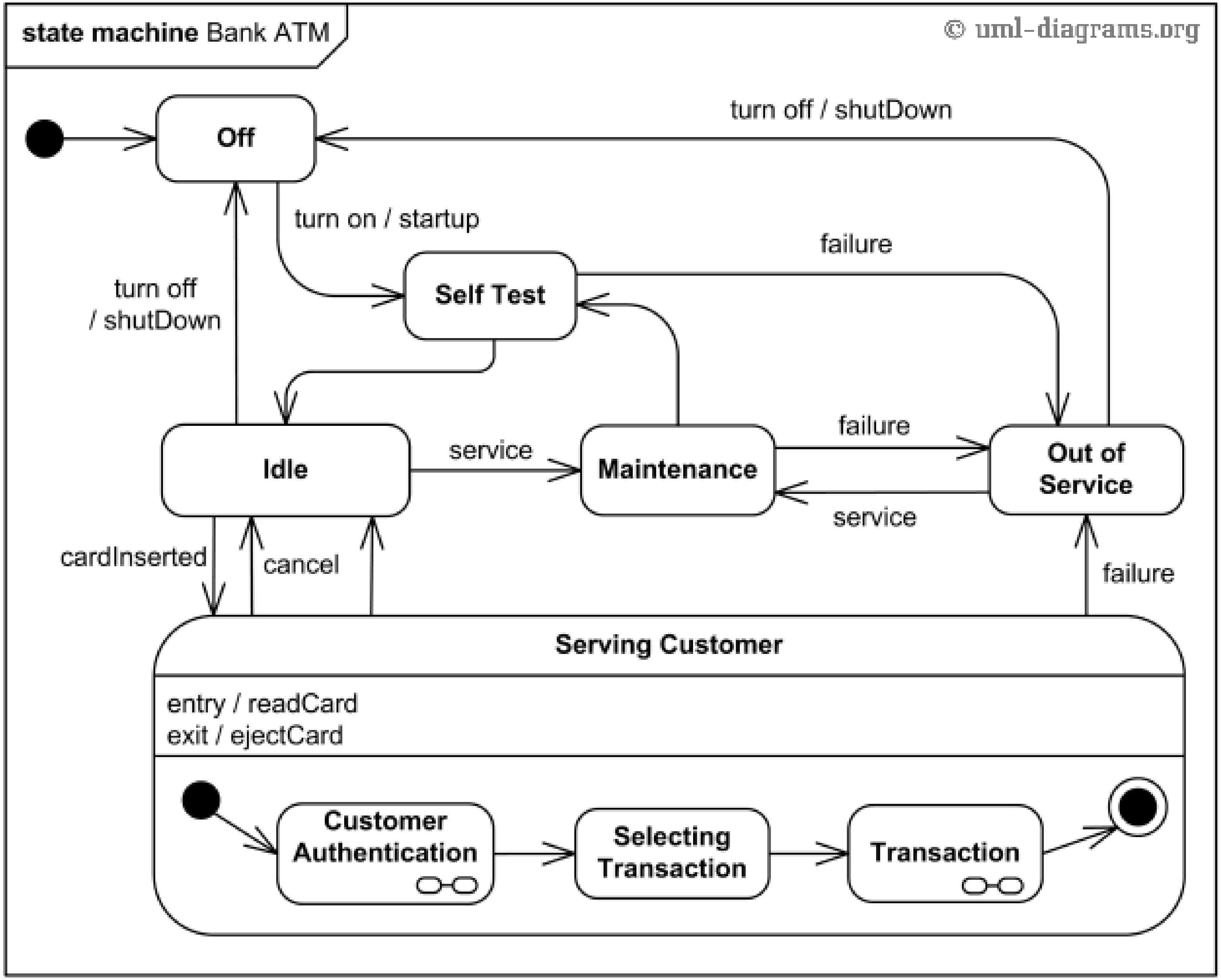
Composite state

- Has sub states (nested states)
- Sub states could be sequential or concurrent

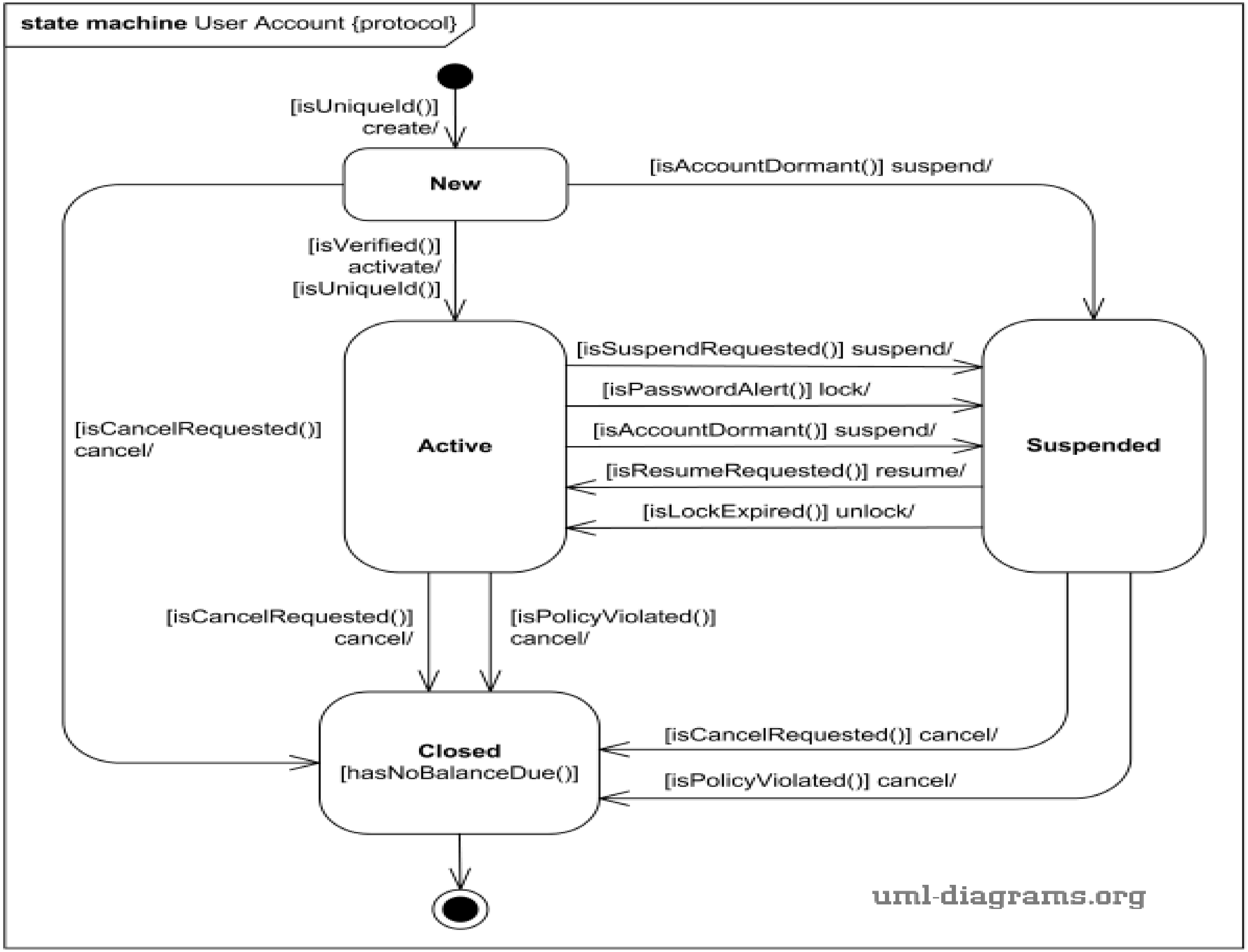

sm ATM
PowerOff Do:sleep
Switch on/ ready to serve
Idle Do:wait for service
Switch off /sleep
card removed/ stop activity
Card inserted / perform activity
Active
Menu Display Do:slect menu
menu selected
transaction continued
Transaction Do:perform tansaction
transaction successful/print
Print Do:print receipt
transaction over

State diagram basic notations

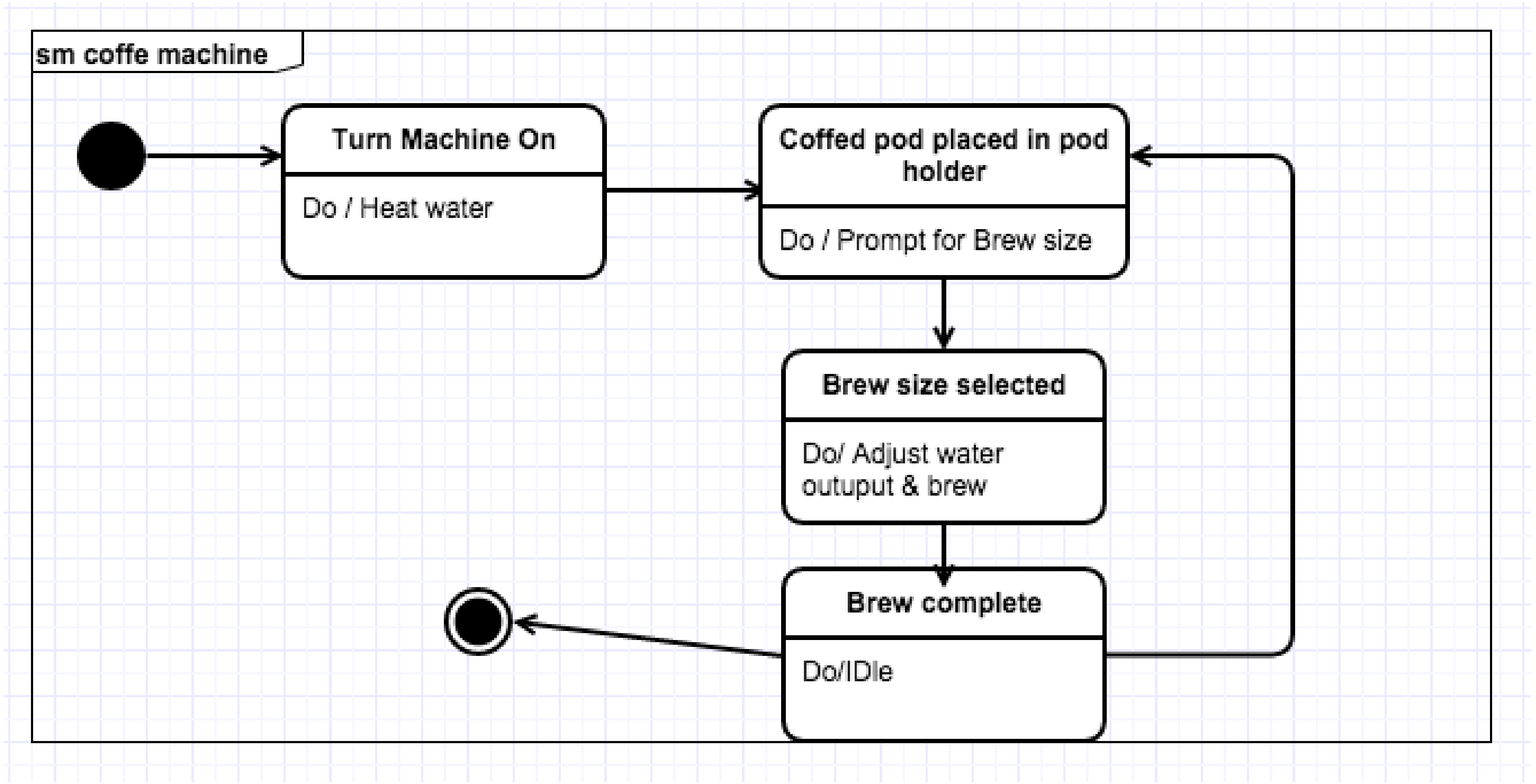


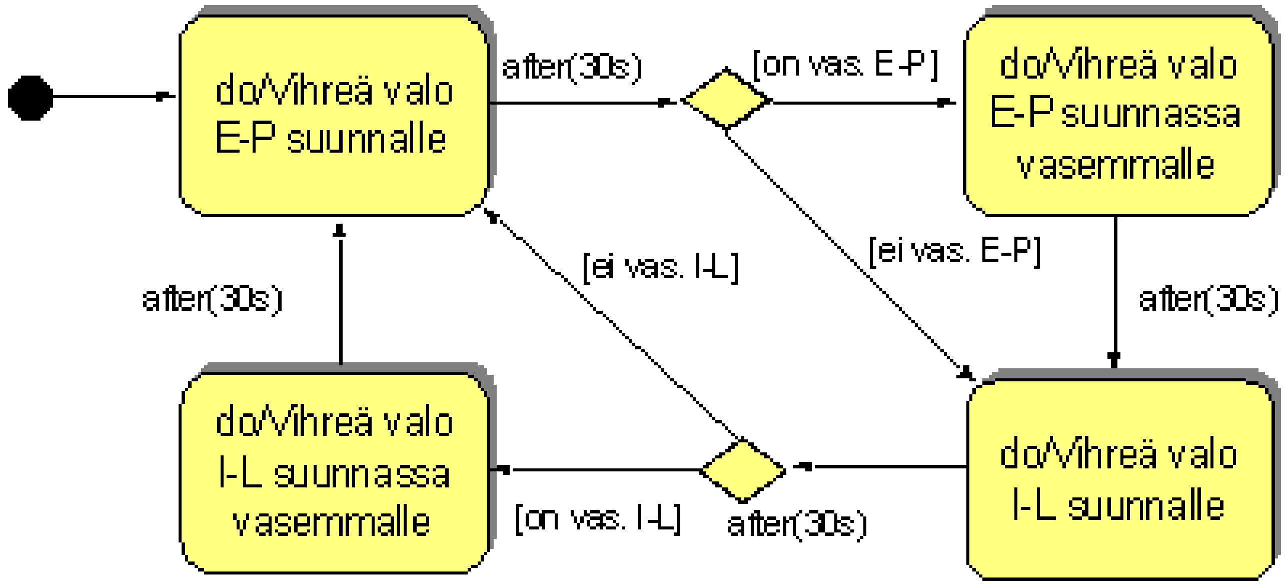


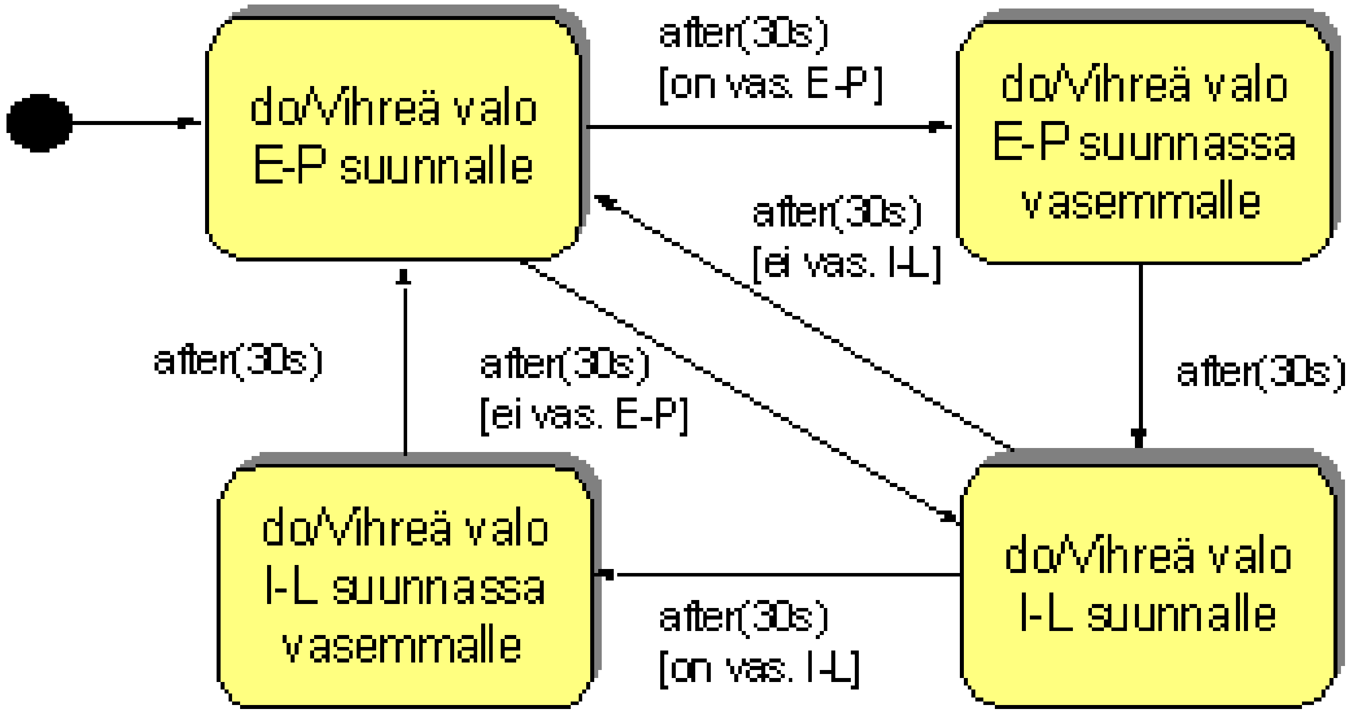
SRC:http://www.uml-diagrams.org/bank-atm-uml-state-machine-diagram-example.html?context=stm-examples


state machine User Account {protocol}
[isUniqueId()] create/
New
[isAccountDormant()] suspend/
[isVerified()] activate/ [isUniqueId()]
[isCancelRequested()] cancel/
Active
[isSuspendRequested()] suspend/
[isPasswordAlert()] lock/
[isAccountDormant()] suspend/
[isResumeRequested()] resume/
[isLockExpired()] unlock/
Suspended
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
Closed [hasNoBalanceDue()]
[isCancelRequested()] cancel/
[isPolicyViolated()] cancel/
uml-diagrams.org

SRC:http://www.uml-diagrams.org/examples/online-shopping-user-account-state-diagram-example.html?context=stm-examples


sm coffe machine
Turn Machine On
Do / Heat water
Coffed pod placed in pod holder
Do / Prompt for Brew size
Brew size selected
Do/ Adjust water outuput & brew
Brew complete
Do/IDle


do/Vihreä valo E-P suunnalle
after(30s)
[on vas. E-P]
do/Vihreä valo E-P suunnassa vasemmalle
[ei vas. E-P]
after(30s)
[ei vas. I-L]
after(30s)
do/Vihreä valo I-L suunnassa vasemmalle
[on vas. I-L]
after(30s)
do/Vihreä valo I-L suunnalle


do/Vihreä valo E-P suunnalle
after(30s) [on vas. E-P]
do/Vihreä valo E-P suunnassa vasemmalle
after(30s) [ei vas. I-L]
after(30s)
after(30s)
after(30s) [ei vas. E-P]
do/Vihreä valo I-L suunnassa vasemmalle
do/Vihreä valo I-L suunnalle
after(30s) [on vas. I-L]

Exercise:

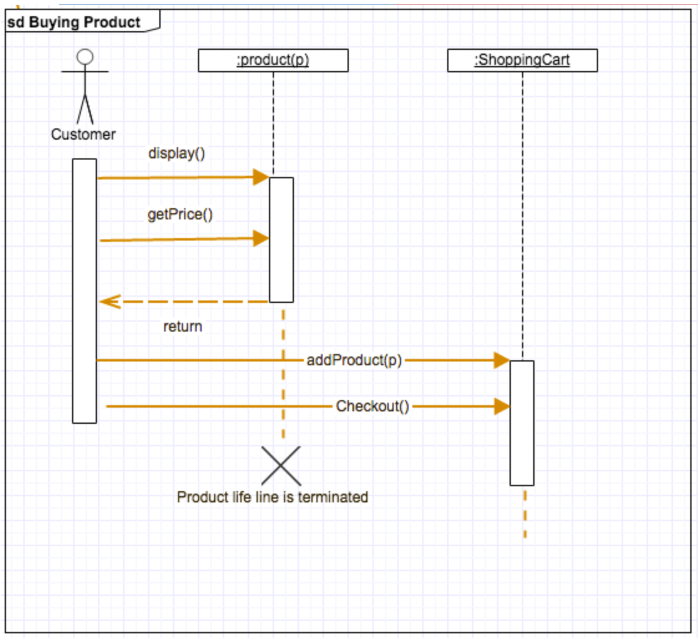
- Draw state diagram for the booking system through a web based application. Model at least the following dialogs
  - A suitable destination
  - Date search
  - Hotel/ flight schedule choosing
  - Entering passenger
  - Travel booking & payment
- Draw a state diagram for the following
  - The course registration process at KAMK
  - The power option of a laptop: shutdown, start, hibernate, reboot etc

Activity Diagram

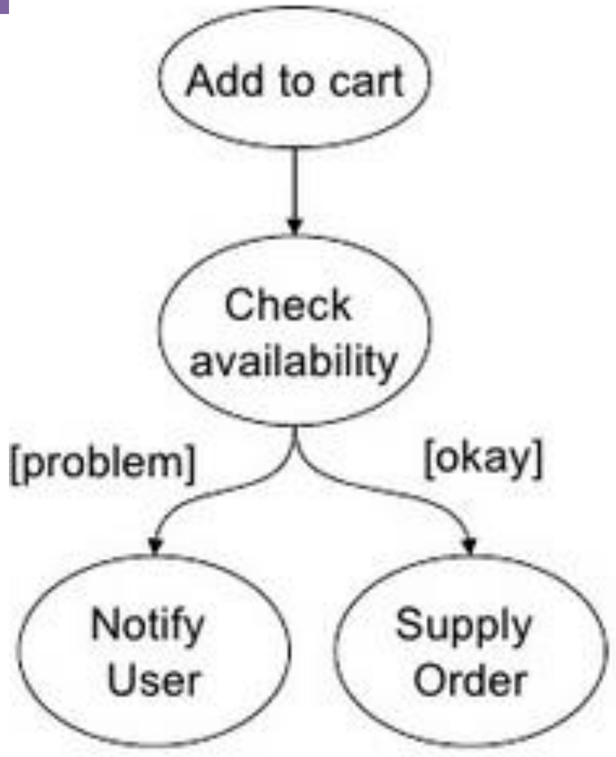
- UML behavior diagrams
- Activity diagram shows flow of control with emphasis on the sequence and conditions of the flow
- Activity >> flow of actions, flow from one activity to another
- Contains activity nodes
  - Action
  - Object
  - control

Behavior modeling

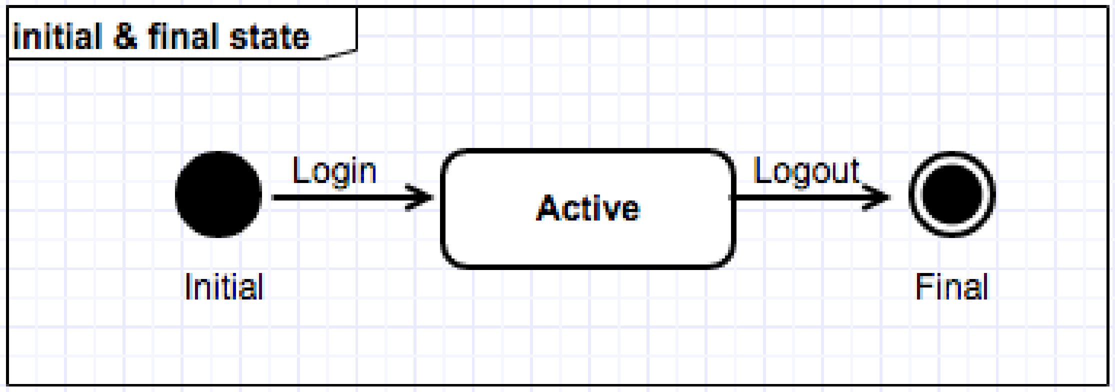
Sequence diagram

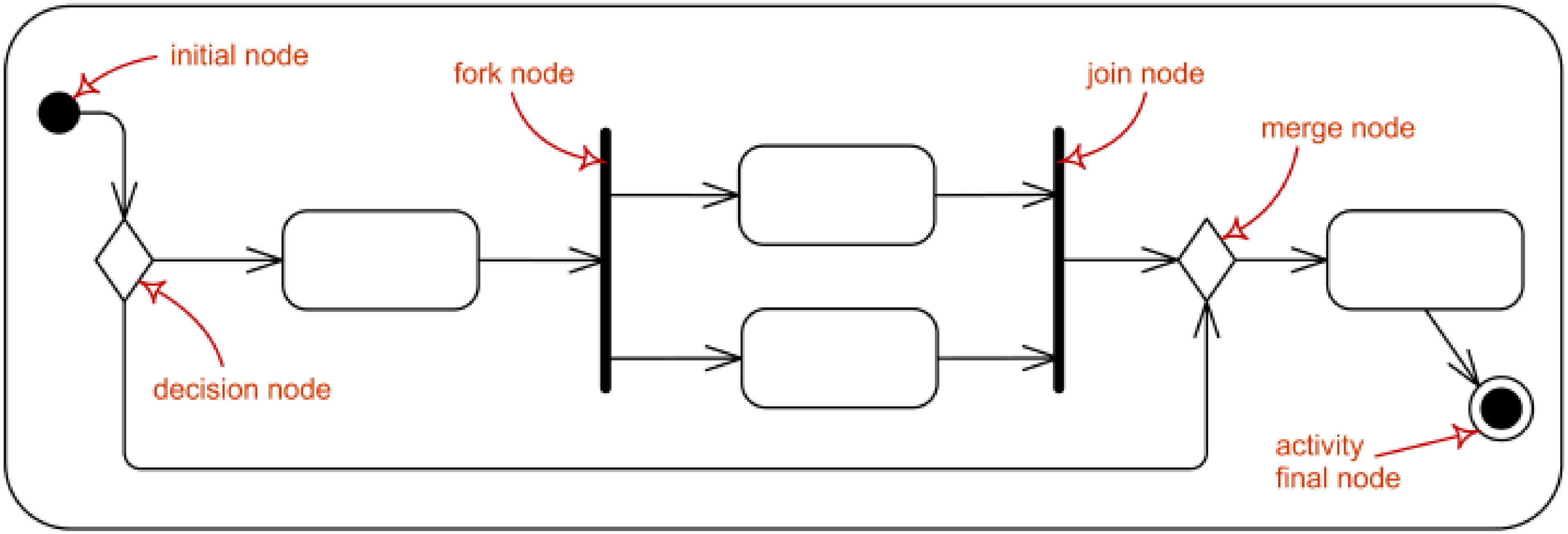


Activity diagram



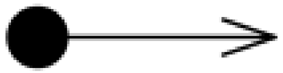
State digram



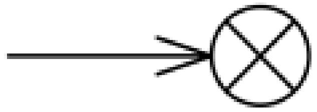

initial node
fork node
join node
merge node
decision node
activity
final node

Notations – Activity diagram

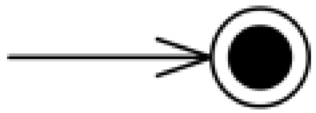
- Initial node
  - A control node at which flow starts when the activity is invoked



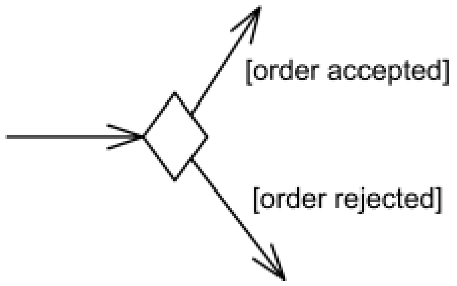
- Flow final node
  - Terminates a flow



- Activity final node
  - Stops all the flows in an activity

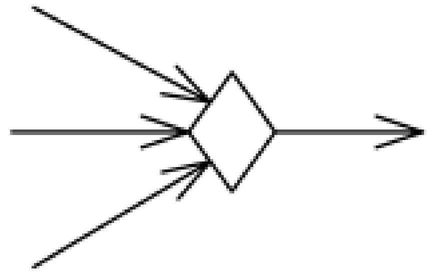


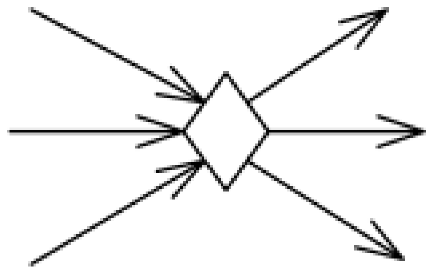
- Decision node
  - Set of guard conditions are defined
  - Conditions must be met to trigger the transition



Notations – Activity diagram

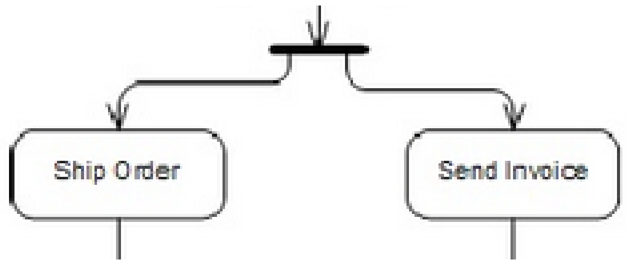
- Merge node
  - Brings together multiple incoming alternate flows to accept a single outgoing flow

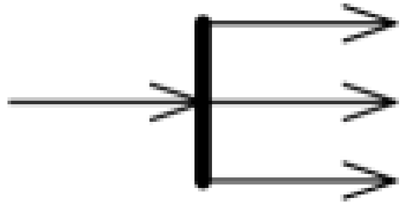

Merge node

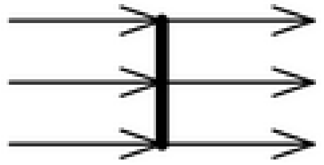

Merge node & decision node together

Fork Node

- One incoming edge that generates multiple outgoing edges
- To split incoming flow into multiple concurrent flows



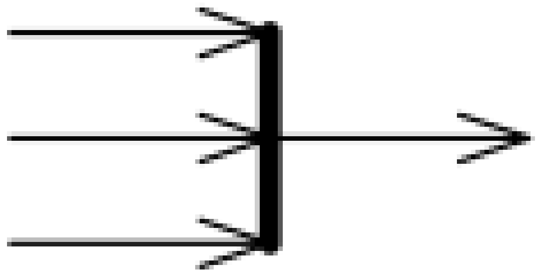




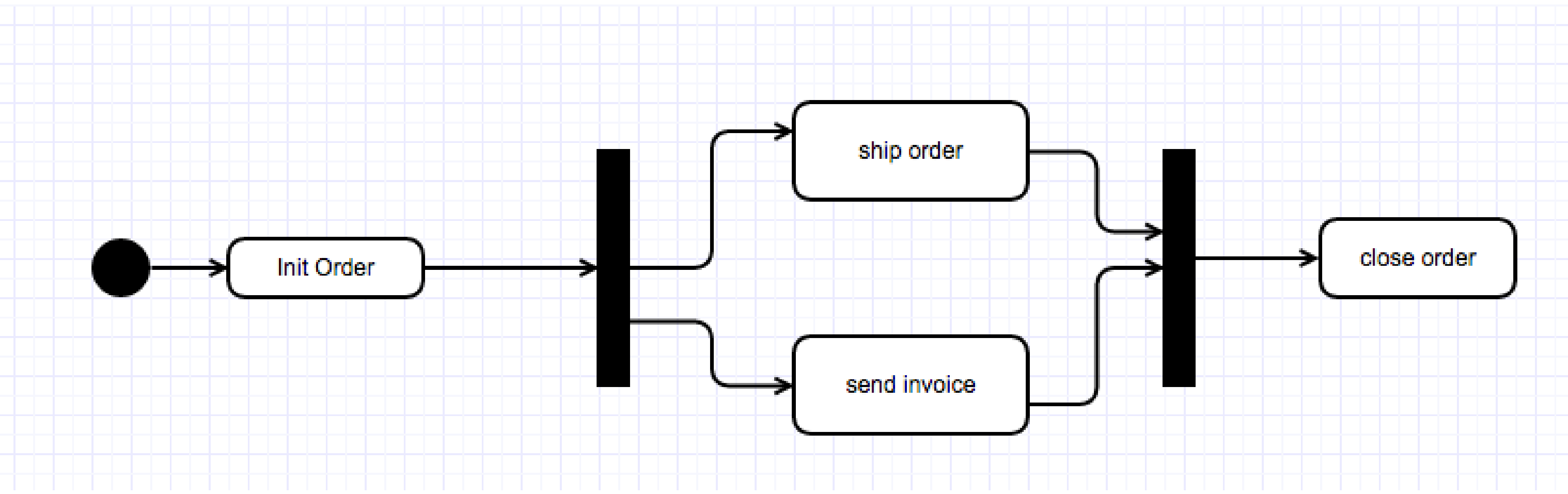
Join node & fork node can be combined using the same node symbol

Join node

- Multiple incoming edges into one outgoing edge
- To synchronize incoming concurrent flows

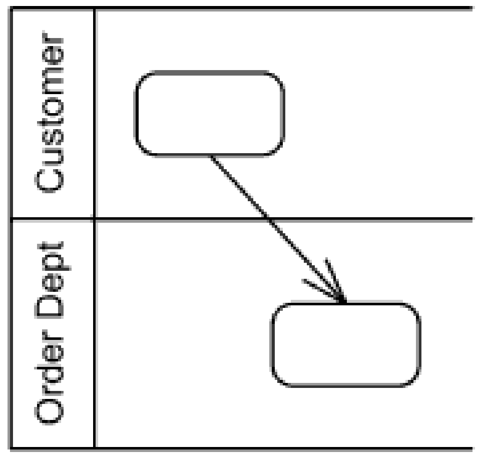


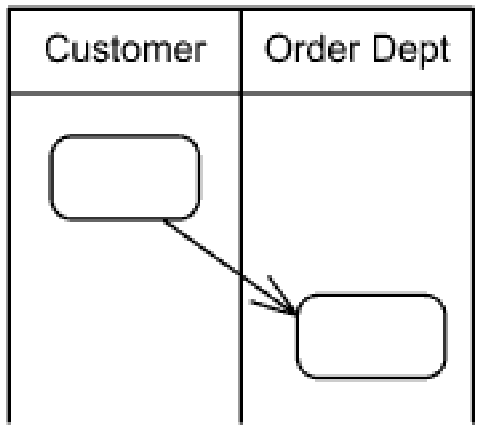
Example



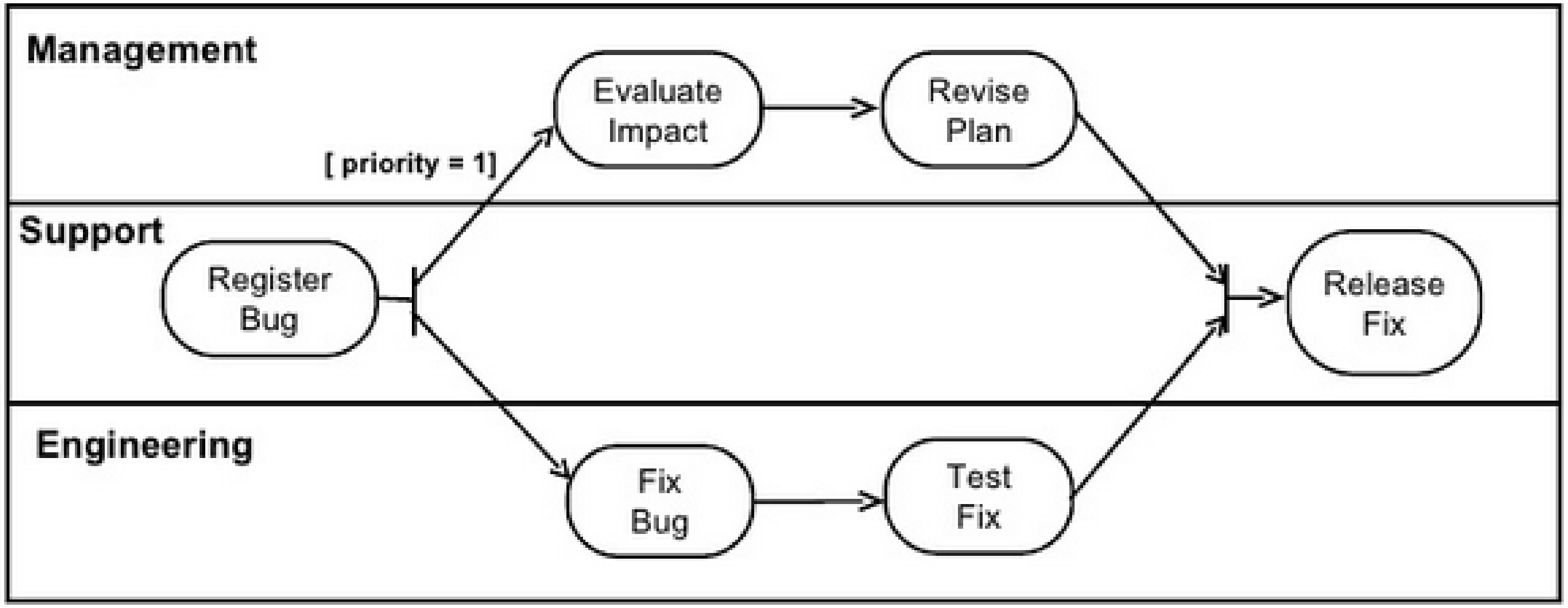
Activity partition

- Group of activities with common characteristic
- Swimlane notation can be used to show activity partition
  - Usually parallel lines either horizontal or vertical

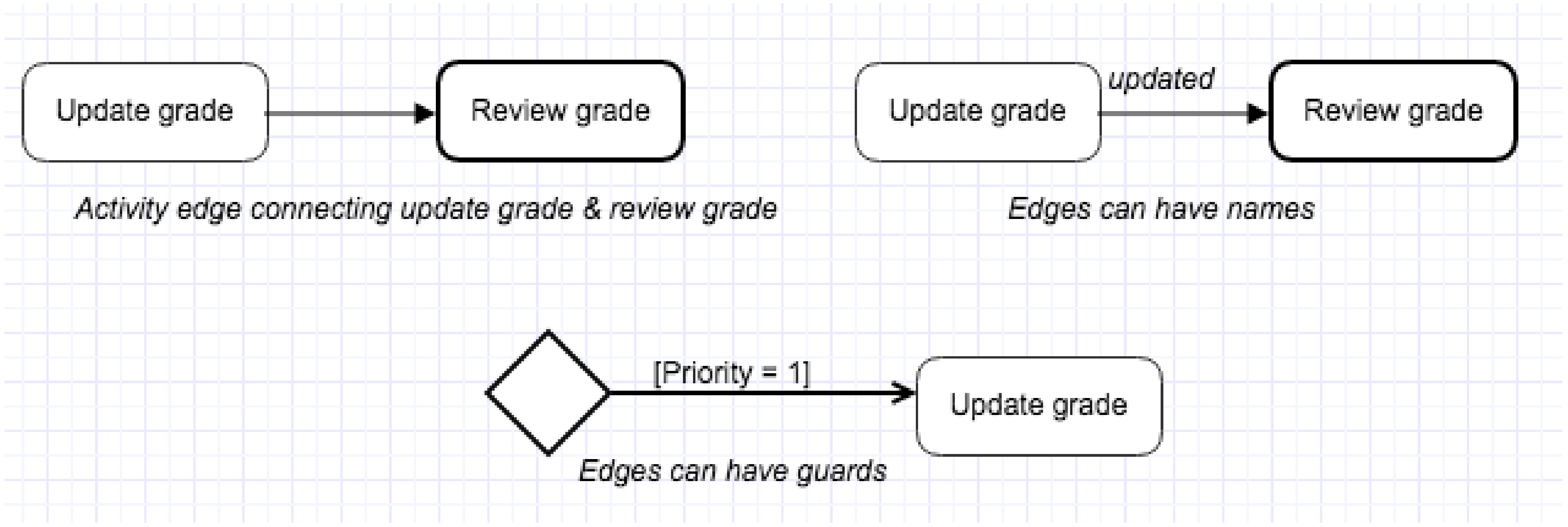




Example

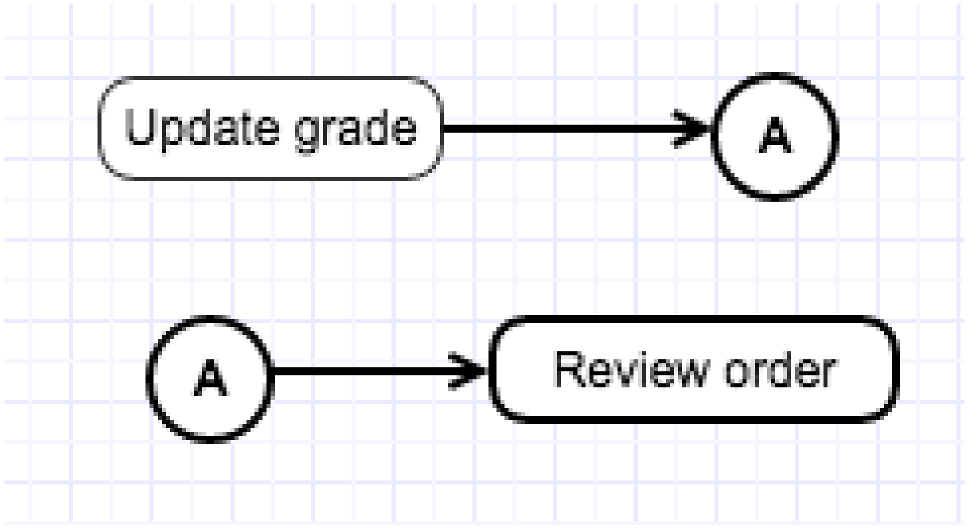

Management
Support
Engineering
Register Bug
[priority = 1]
Evaluate Impact
Revise Plan
Fix Bug
Test Fix
Release Fix

Activity edge



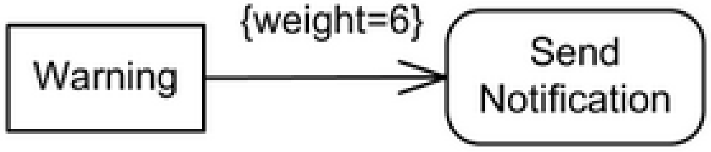
Connectors

- Used to avoid drawing a long edge
- Notational and does not affect the underlying model

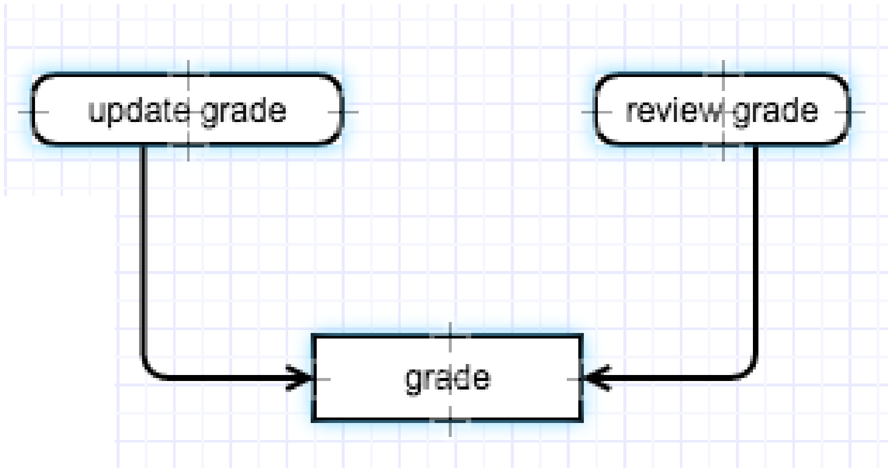


Object flow edge

- To show data flow of object and data tokens between action nodes
- Weight of the edge can be shown in curly braces

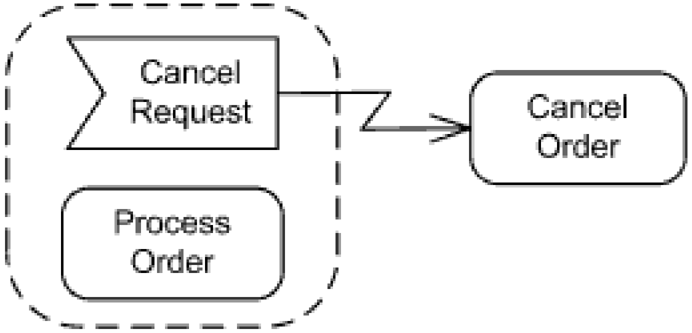


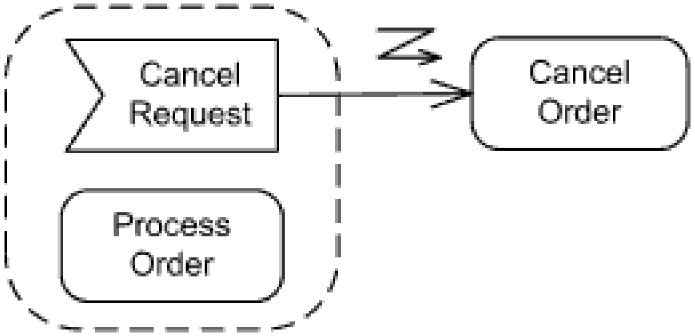
Send Notification when number of Warnings reaches 6.

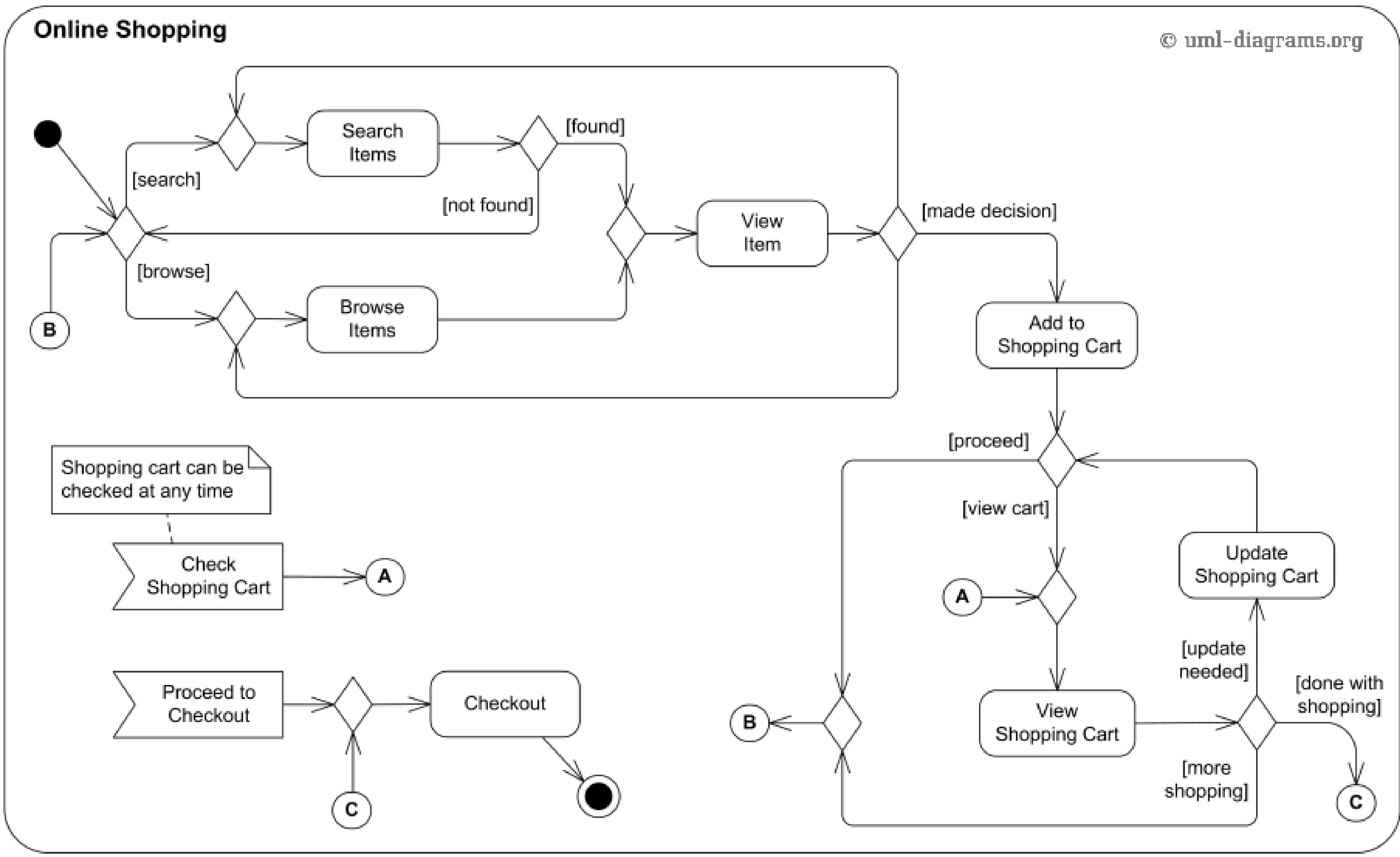


Interrupting edge

- Expresses interruption for regions having interruptions

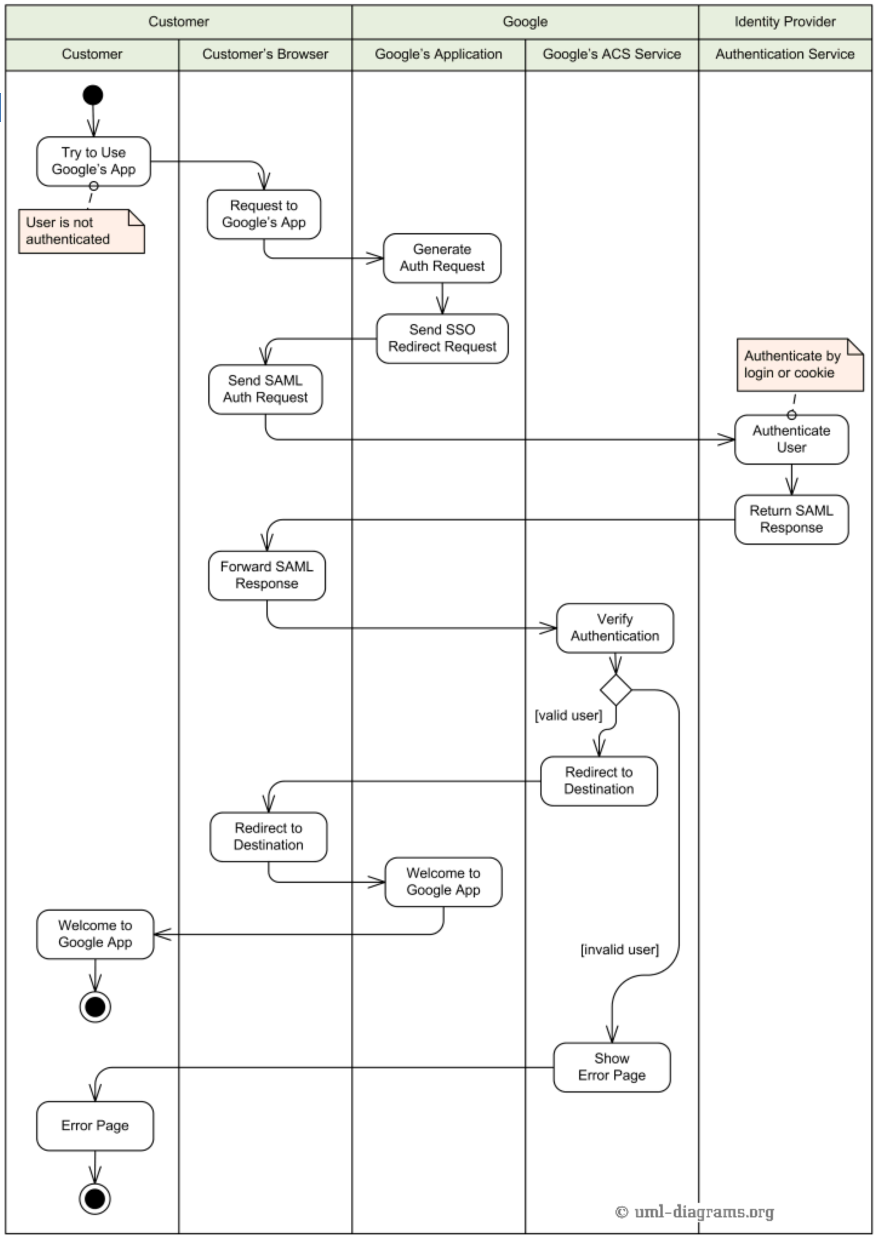






Src: http://www.uml-diagrams.org/online-shopping-uml-activity-diagram-example.html?context=activity-examples

Single Sign-On for Google Apps



http://www.uml-diagrams.org/examples/activity-example-google-sso.png

Exercise

- Draw activity diagram of an order management system.

References

- UML 2 & the Unified Process Second Edition – Practice Object Oriented Analysis & Design by Addison-Wesley
- OMG Unified Modeling Language (OMG UML) specification (UML Superstructure Specification version 2.4.
- Some diagrams & example : http://www.uml-diagrams.org/