Modeling with UML

Presented by:

Basma Hussien

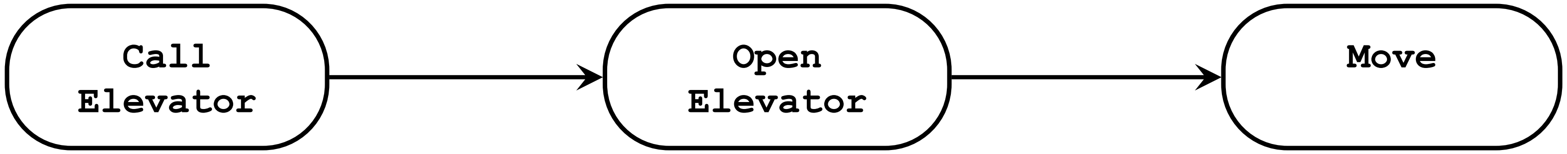
12/21/2022

Contents

Activity Diagrams

Activity Diagrams

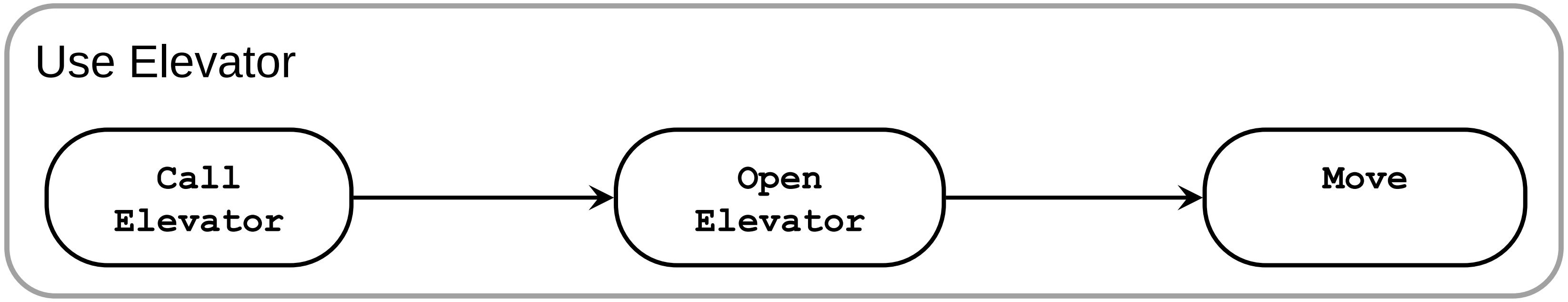
- **An activity diagram shows flow control within a system**
  - ✓ i.e. Shows a procedural flow for a process

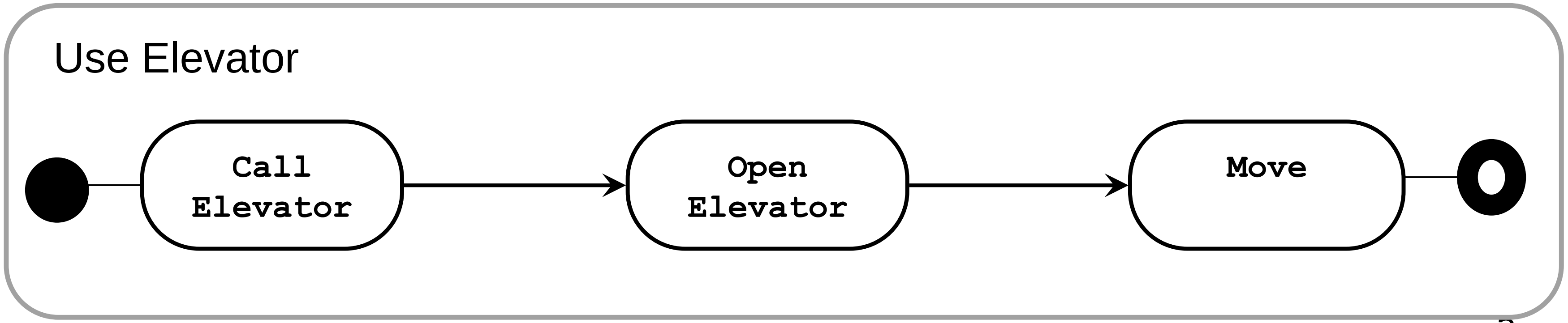


- **An activity diagram is a special case of a state chart diagram in which states are activities (“functions”)**
- **Two types of states:**
  - ✓ *Action state:*
    - Cannot be decomposed any further
    - Happens “instantaneously” with respect to the level of abstraction used in the model
  - ✓ *Activity state:*
    - Can be decomposed further
    - The activity is modeled by another activity diagram

Activity Diagram (cont.)

Activity with details:

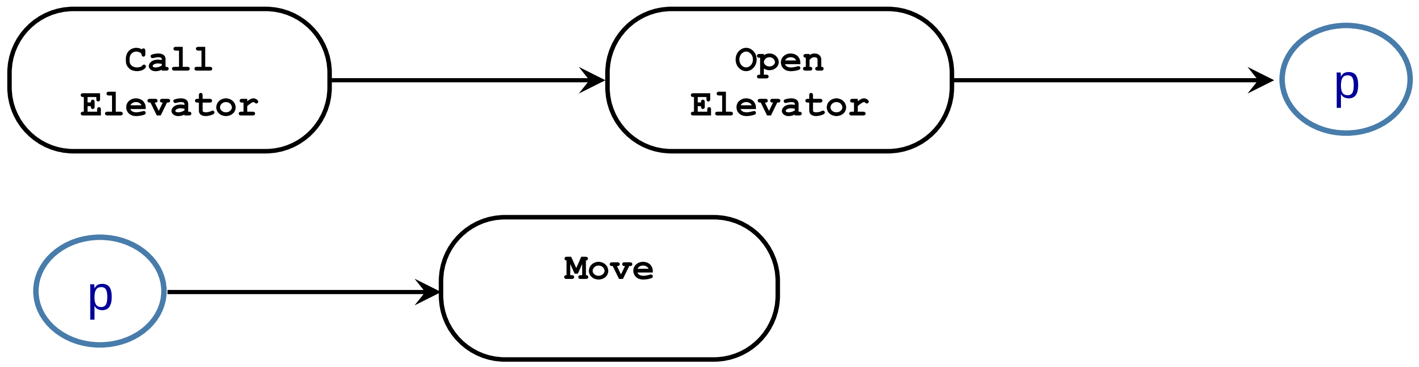




Activity Diagram (cont.)

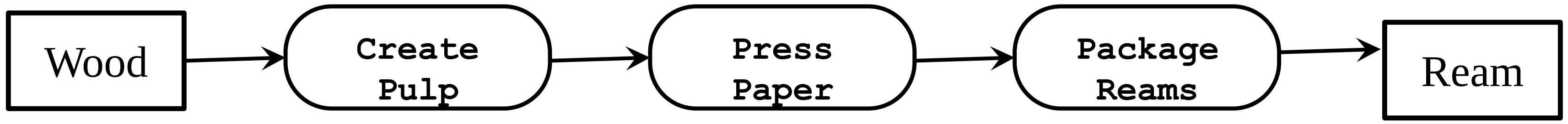
Connectors:

- Simplify large activity diagrams by splitting edges using connectors.
- Each connector is given a name.
- Place the name of a connector in a circle and then show the first half of an edge pointing to the connector and the second half coming out of the connector

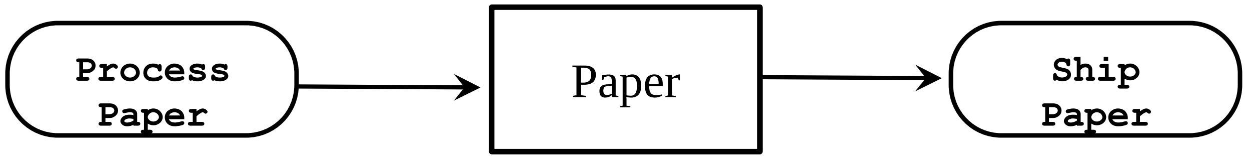


Activity Diagram (cont.)

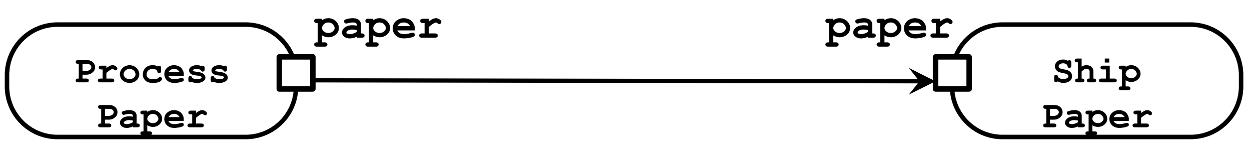
Parameter Nodes:



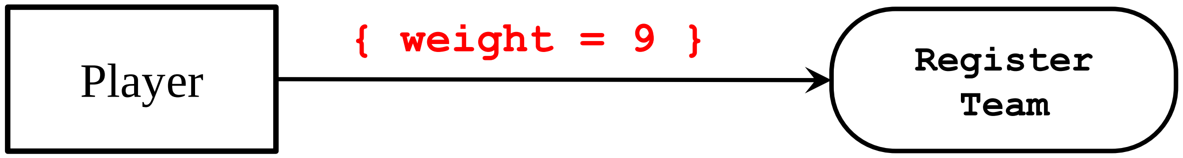
Object Nodes:



Pins:

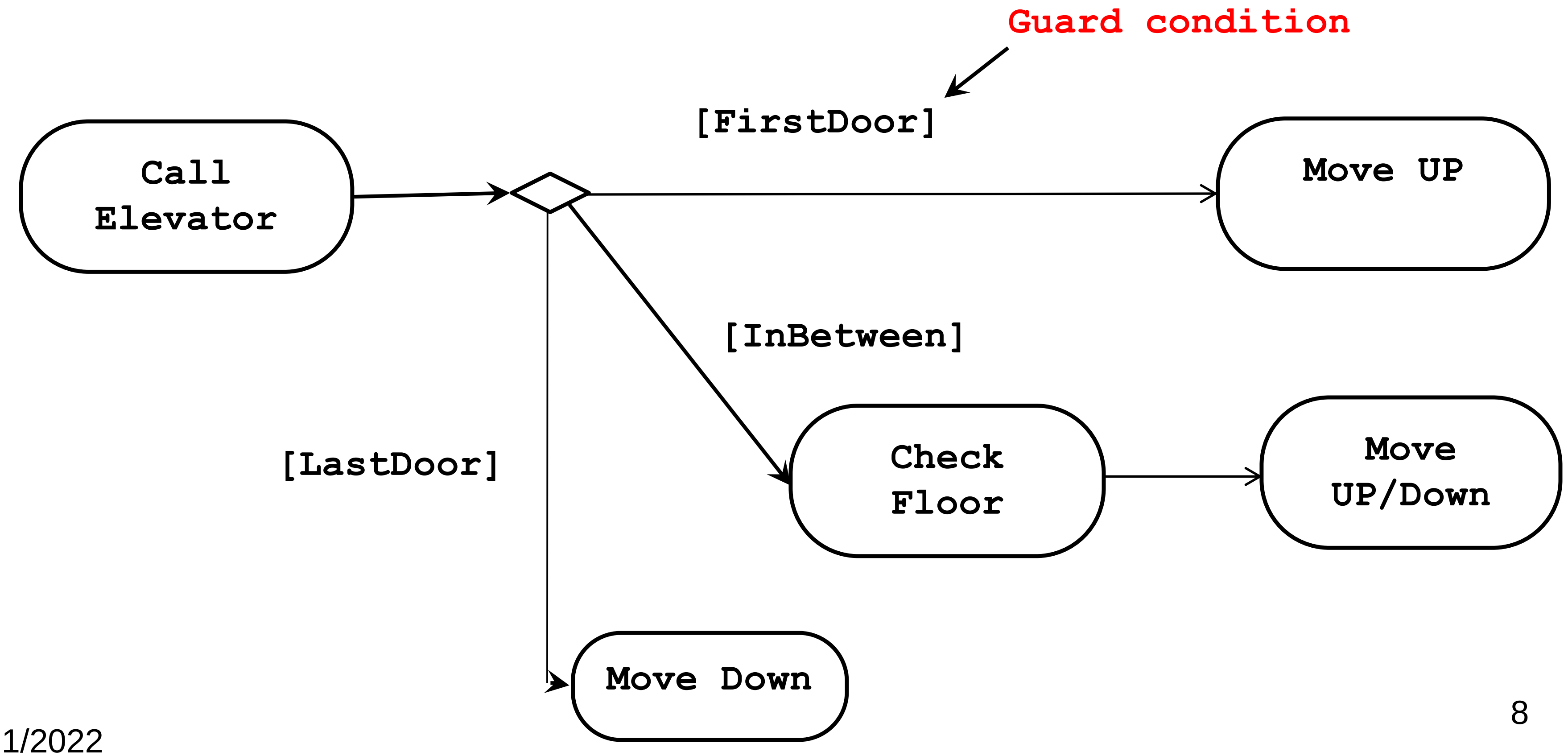


Tokens:



Activity Diagram (cont.)

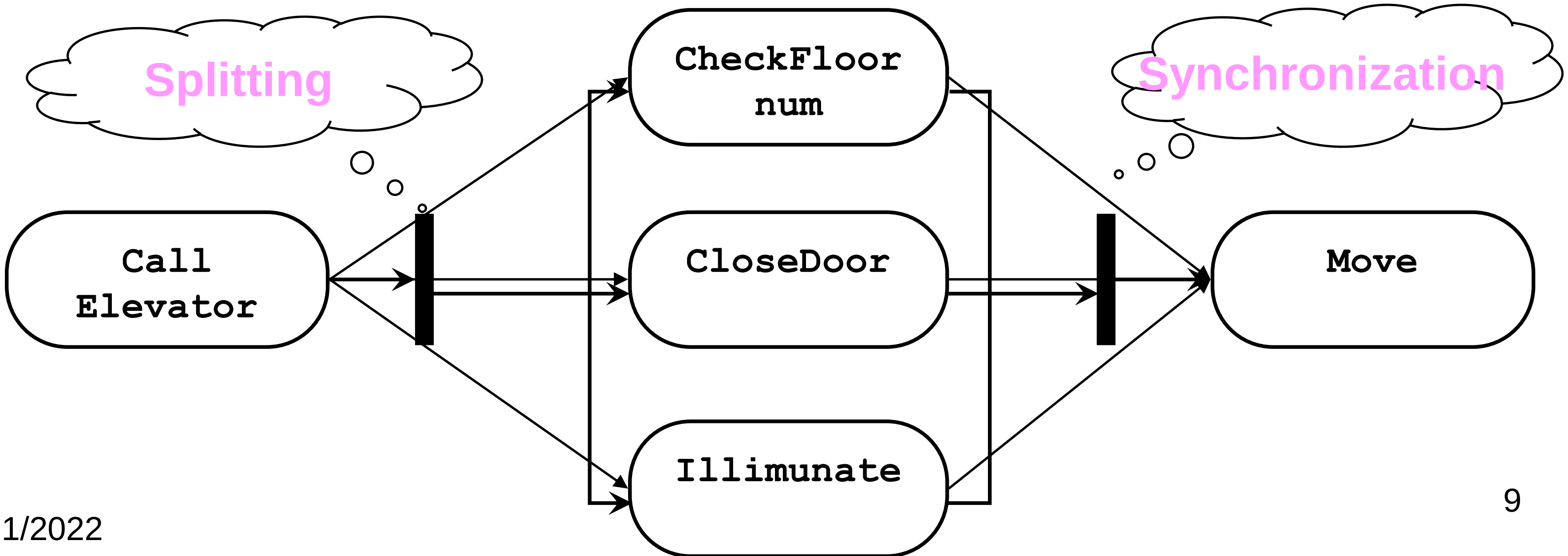
Modeling Decisions:



Activity Diagrams (cont.)

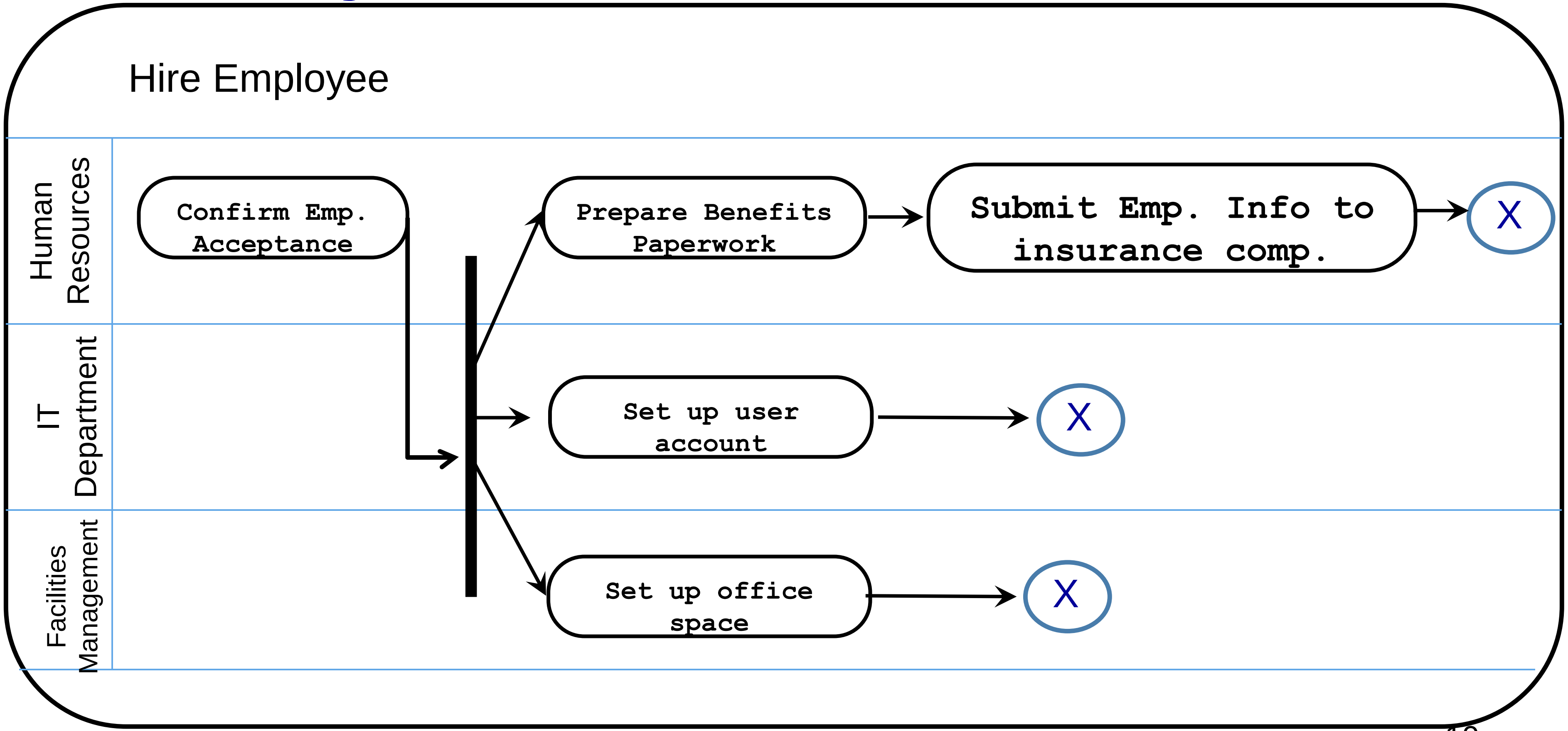
Modeling Concurrency:

- *Fork:* **Splitting the flow of control into multiple threads.**
- *Join:* **Synchronization of multiple activities.**



Activity Diagram (cont.)

Activity Partitions:



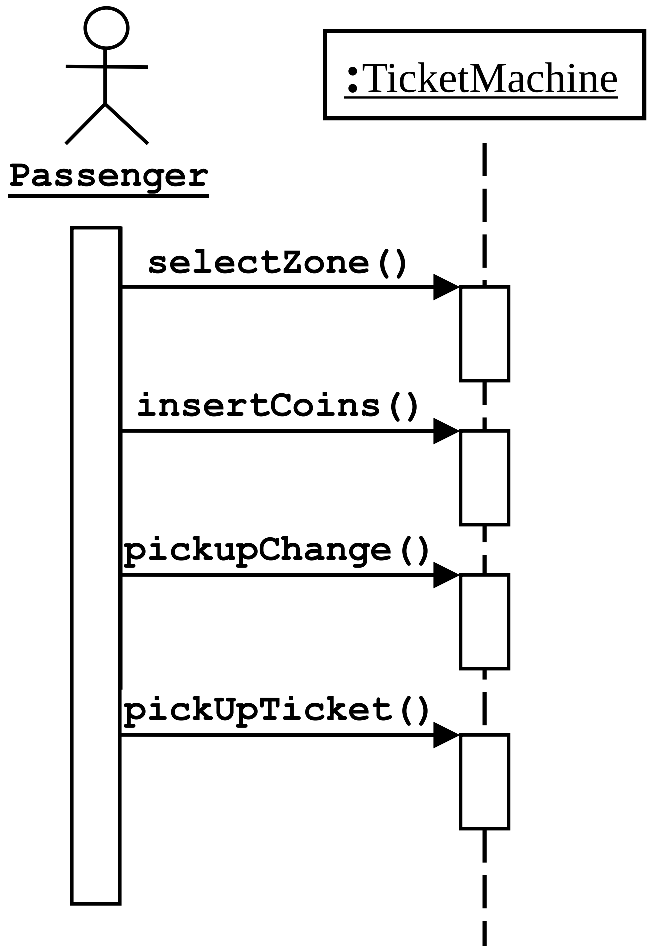
FlowChart & Activity Diagram

- Is Flowchart a UML diagram?

  - Both are similar…
  -
  - But, An **activity diagram** is a **UML diagram**. A **Flowchart**, on the other hand, is **NOT** a UML diagram, it is a graphical diagram that represents algorithm to solve a given problem “a step by step procedure”

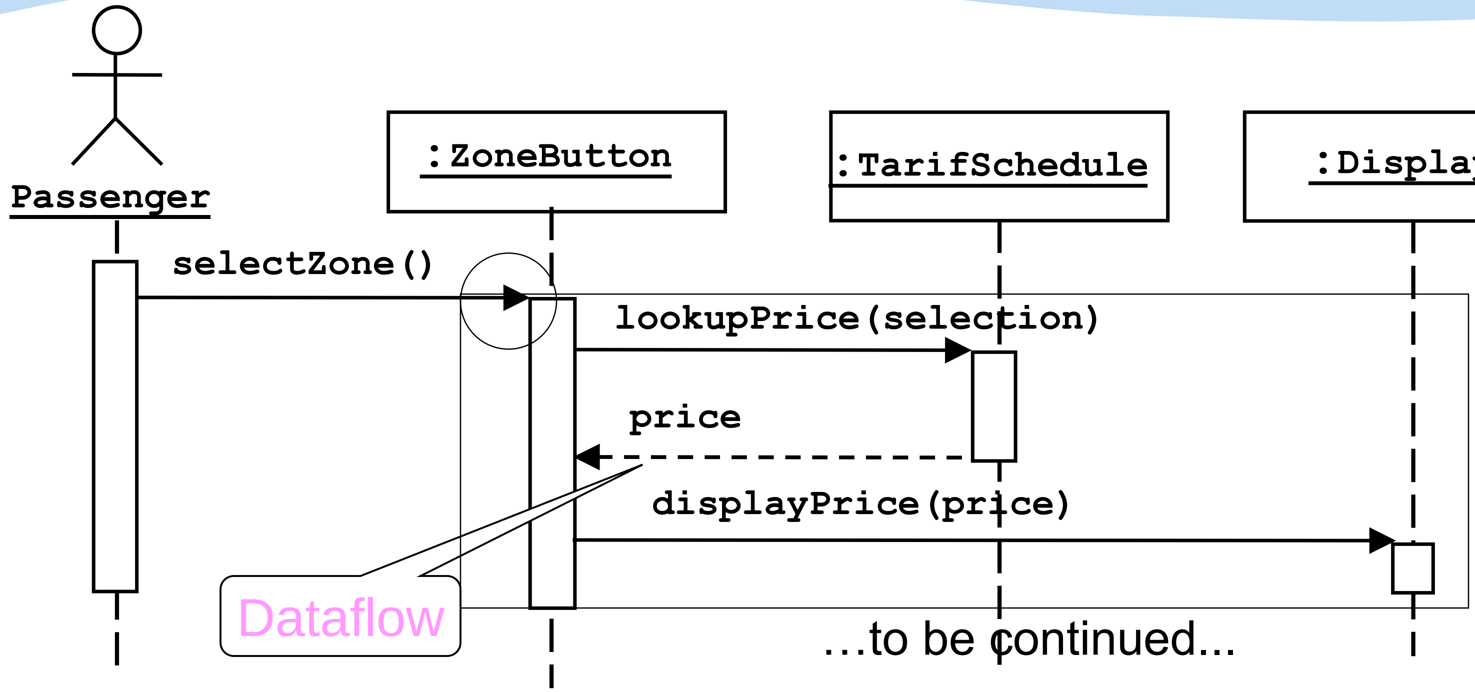
Sequence Diagrams

UML Sequence Diagrams



- *Objects* **are represented by rectangles "Underlined"**
- *Messages* **are represented by arrows**
- *Activations* **are represented by narrow rectangles**
- *Lifelines* **are represented by vertical dashed lines**
- **Used during requirements analysis**
  - ✓ To refine use case descriptions
  - ✓ to find additional objects ("participating objects")
- **Used during system design**
  - ✓ to refine subsystem interfaces

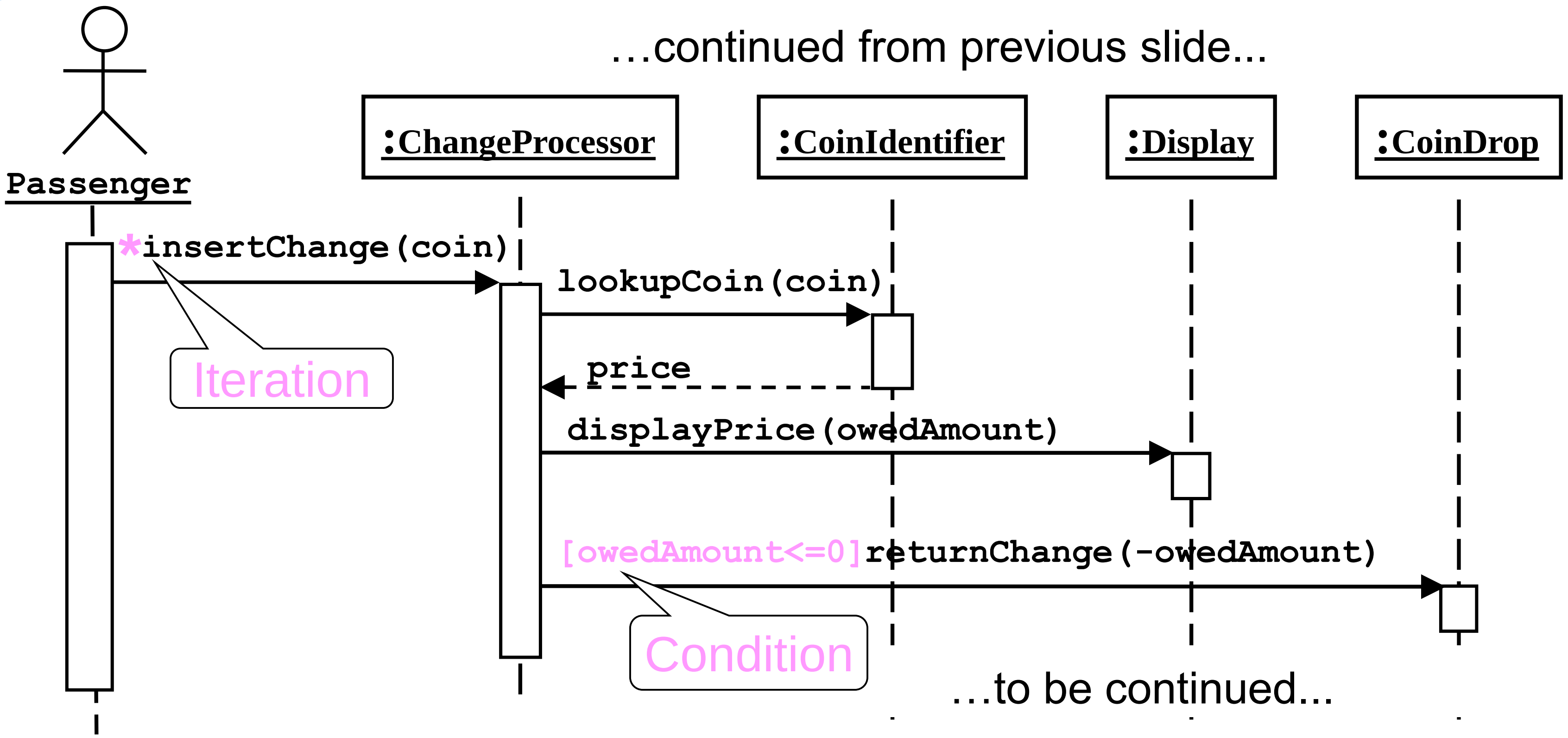
Nested messages



- The source of an arrow indicates the activation which sent the message
- Horizontal dashed arrows indicate data flow
- An activation is as long as all nested activations

Iteration & condition

…continued from previous slide...

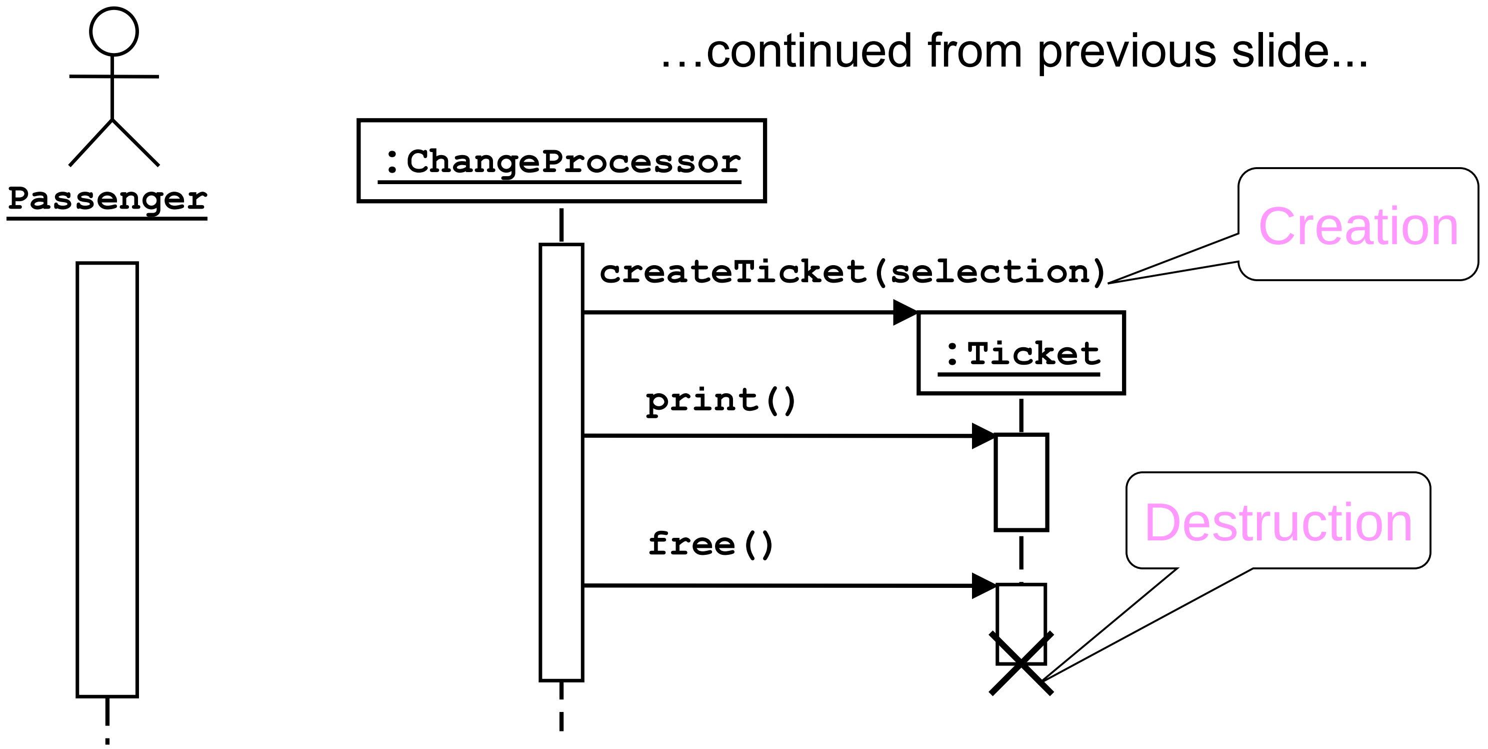


…to be continued...

- Iteration is denoted by a * preceding the message name
- Condition is denoted by Boolean expression in [ ] before the message name

Creation and destruction

…continued from previous slide...

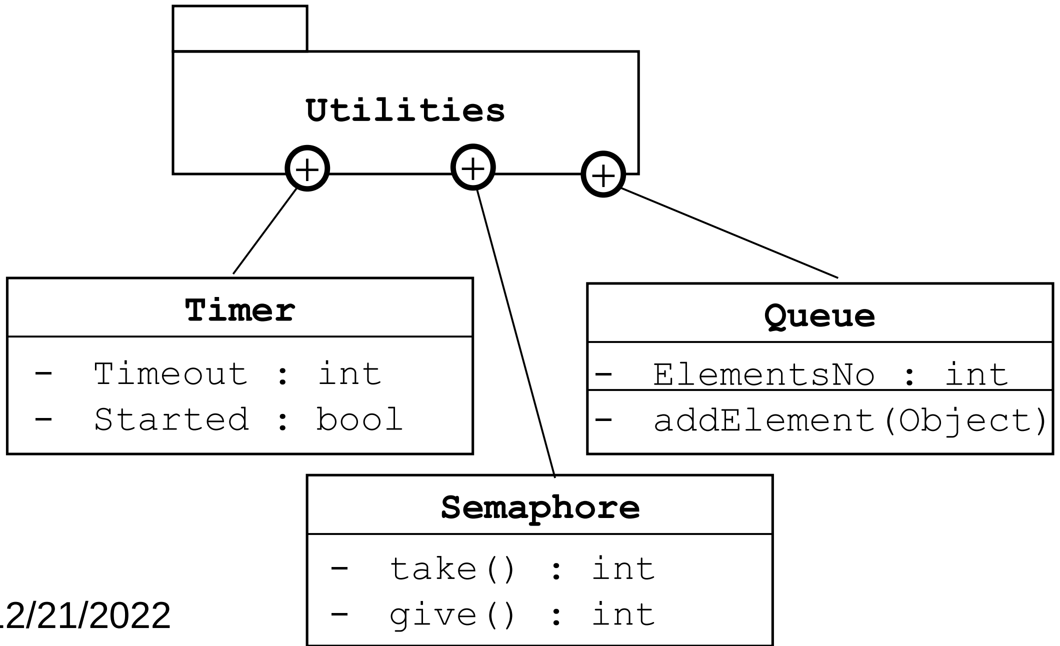


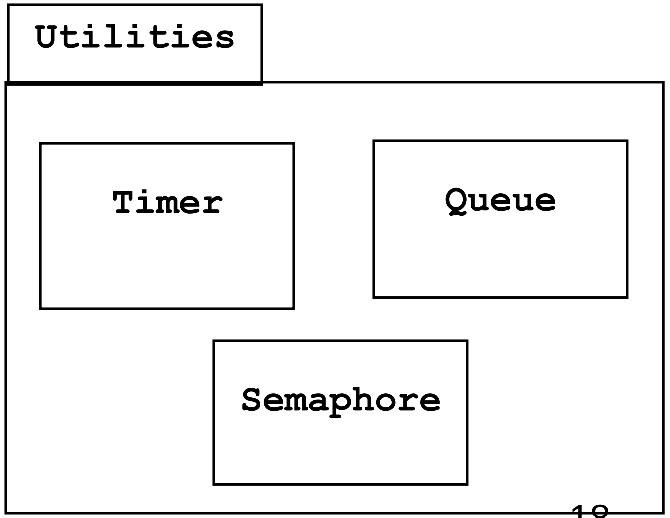
- **Creation is denoted by a message arrow pointing to the object.**
- **Destruction is denoted by an X mark at the end of the destruction activation.**

Package Diagrams

Package Diagram

- Provide a way to group related UML elements and scope their names
- Provide a great way to visualize dependencies between parts of system.
- Often used to look for problems or determine compilation order.
- All UML elements can be grouped into packages, including packages themselves.
- Each package has a name that scopes each element in the package.

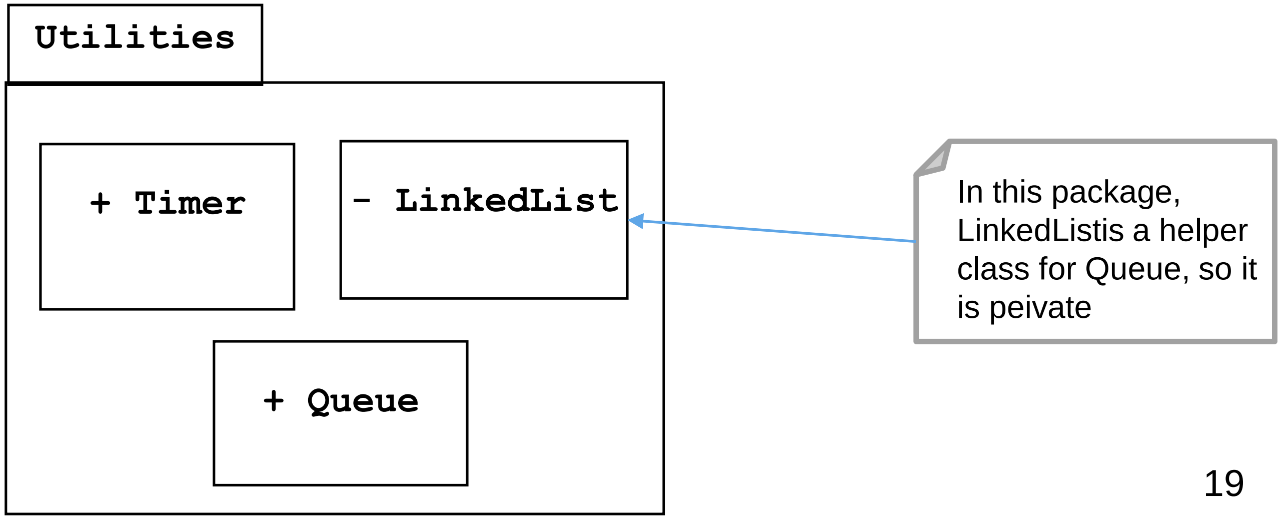




Package Diagram (cont.)

Visibility:

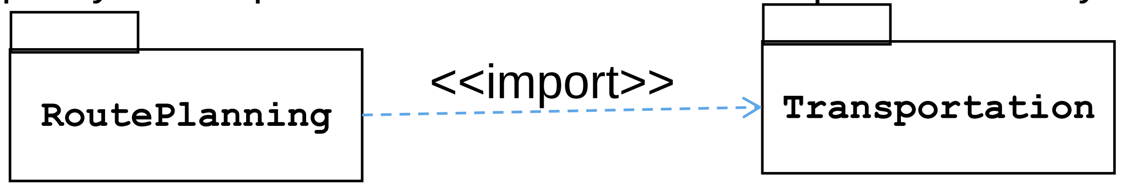
- Elements may have only one of two levels of visibility: **public** or **private**.
- Public visibility means the element may be used outside the package (Utilities::Timer)
- Private visibility means the element may be used only by other elements of the same package

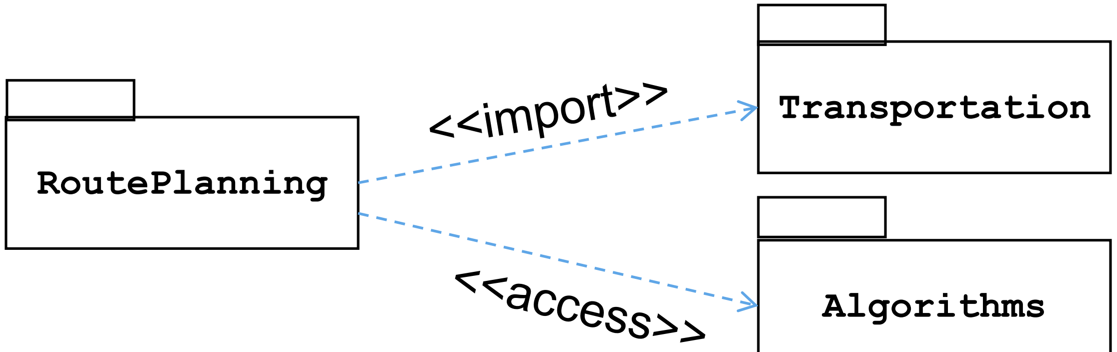

Utilities
+ Timer
- LinkedList
+ Queue
In this package, LinkedListis a helper class for Queue, so it is peivate

Package Diagram (cont.)

Importing and Accessing Packages:

- When accessing elements in one package from a different package, you must qualify the name of the element you are accessing
- UML allows a package to *import* another package. Elements of the imported package are available without qualification in the importing package
- By default, imported elements are given public visibility in the importing package.
- Use <<access>> to specify that imported elements should have private visibility



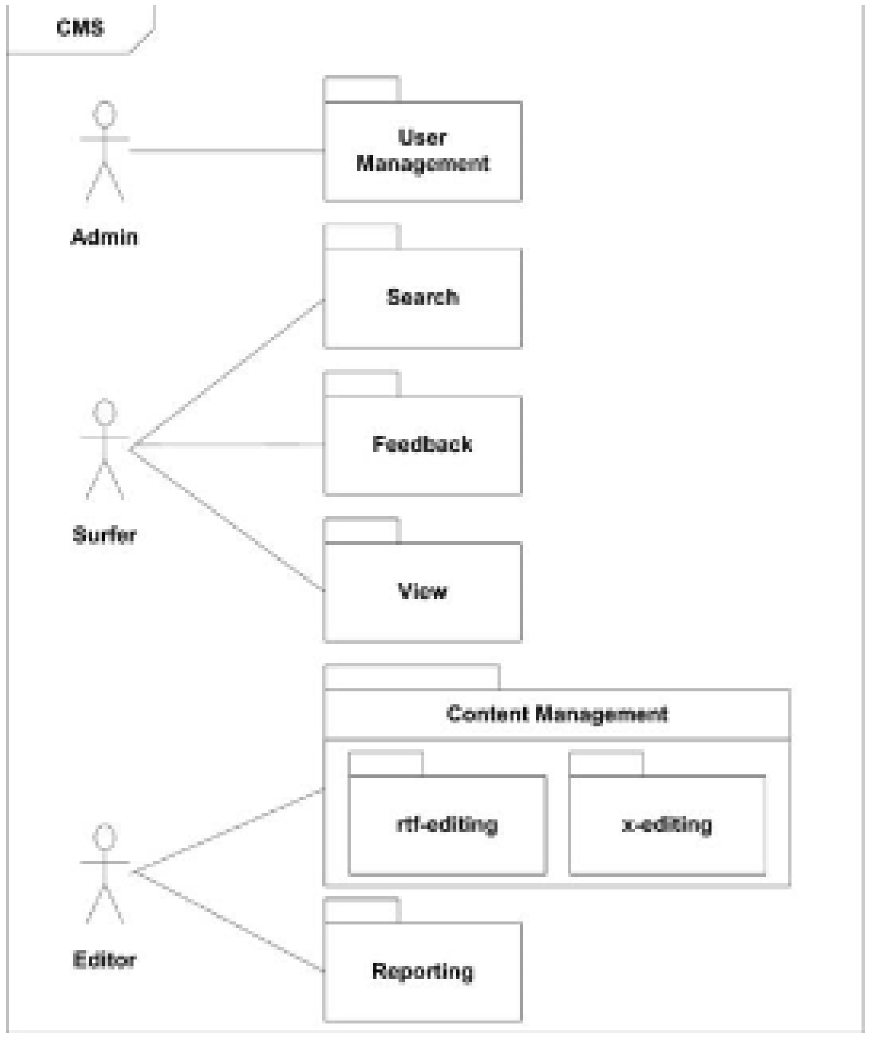


Package Diagram (cont.)

Use Case Packages:

- Use case packages organize the functional behavior of a system during analysis.
- The packages provide understandable terms for team members outside the analyst team. Managers can discuss the project at an appropriate level of detail without getting bogged down in details.
- This example shows the major functional areas of a content management system.

Figure 3-9. A set of functional major use case packages



UML Certification

OCUP

OMG Certified UML Professional

Three Certification Levels:

- ***OCUP Fundamental***
- ***OCUP Intermediate***
- ***OCUP Advanced***

Link : http://www.omg.org/uml-certification/

Thanks