

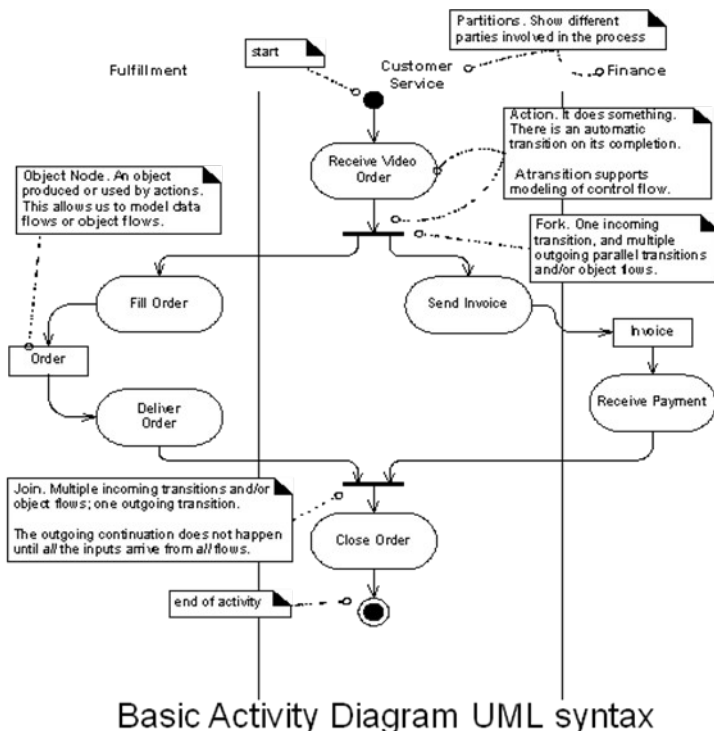
# Chapter 27 – Iteration-3

The third iteration takes a border view again, exploring a verity of analysis and design topics, including:

- More GoF design patterns and their application to the design of frameworks; in particular, a persistence framework
- Architectural analysis and documenting
- Process modeling with UML activity diagrams
- Generalization and specialization
- The design of packages

# Chapter 28 – UML ACTIVITY DIAGRAMS

- Activity Diagrams are basically modernized flowcharts/Data Flow Diagrams
- Biggest improvement is showing parallel activities
- Easy to understand
- Most underutilized UML model
- Used to model processes, workflows & complex UCs



Basic Activity Diagram UML syntax

# Chapter 29 – STATE MACHINE DIAGRAMS

- Models an object’s changes in state due to events (e.g., a message that changes it’s state)
- Depicts states and transitions between states
- Depicts an object’s lifecycle
- Typically used for objects with complex lifecycle

### Event

- Significant or noteworthy occurrence

### State

- Condition of an object at a moment in time

### Transition

- Directed relationship between 2 connected states
- Labeled by an event
- When the event occurs, the object moves to the next state

### Common Applications→

- [1] Physical devices
- [2] Transactions related to Business Objects
- [3] Communication protocols
- [4] Screen flow in GUIs
- [5] Session management (server-side)
- [6] Use Case operations (also handled by activity diagrams)
- [7] UI event handling



# Chapter 30 – RELATING UCs

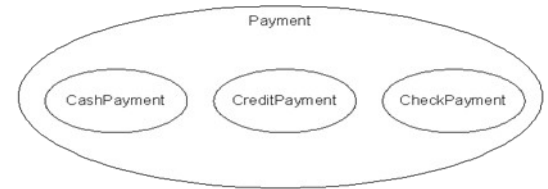
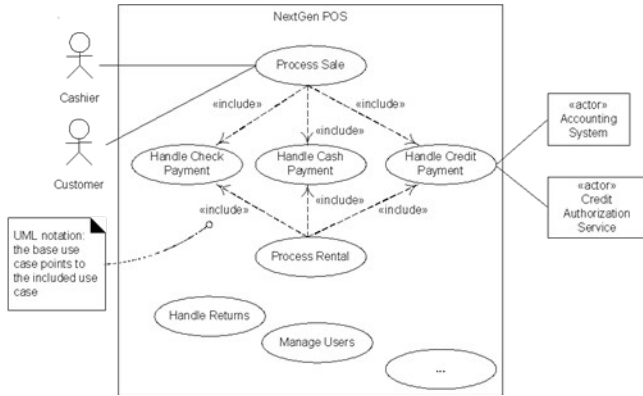
### Include relationship

- [1] Most common & most important
- [2] Use it when you’re repeating yourself across UCs
- [3] Refactor duplicate UC triplets that occur in multiple UCs into separate UC
- [4] Reuse refactored UC by ‘including’ it in those UCs where it is needed
- [5] Similar to refactoring duplicate code into a method and calling the method
- [6] Calling UC can hyperlink to included UC

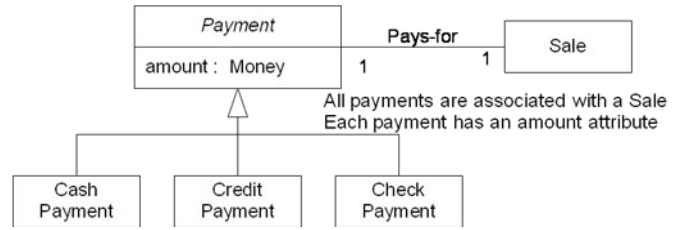
### Extend relationship

- [1] A UC that extends or adds functionality to a base UC

- [2] Very similar to inheritance
- [3] Base UC does not reference extending UC

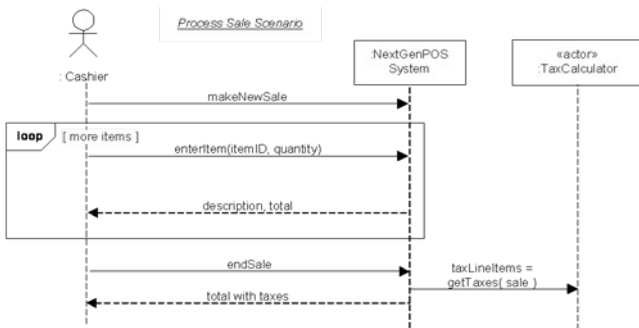


Venn diagram representation of inheritance hierarchy



All payments are associated with a Sale  
Each payment has an amount attribute

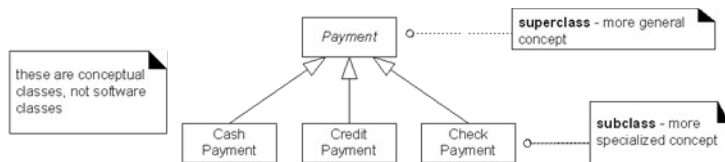
## Chapter 31 – More SSDs and Contracts



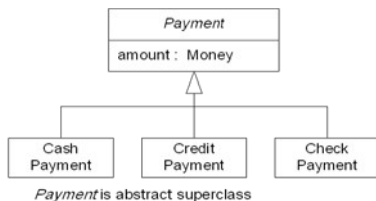
Note collaboration with TaxCalculator system

## Chapter 32 – Domain Model Refinement

- Need to refine domain model based on new reqs.
- Use concepts category list to suggest new domain classes
  - Can also harvest nouns from revised Process Sale UC



Use of Inheritance to group similar domain concepts



### 100% Rule →

100% of the superclass's definition should be applicable to each subclass, WRT

- Attributes
- Associations

Subclasses also inherit superclass methods but can override them

### Define a subclass when

- [1] Complies with the 100% rule
- [2] Conforms to the 'is-a' set membership rule

### Define a subclass when it has

- [1] Additional attributes of interest
- [2] Additional associations of interest
- [3] Additional methods or overrides inherited methods

### Define a superclass when subclasses

- [1] Are variation of a single concept
- [2] Conform to 100% Rule and 'Is-a'
- [3] Have common attributes and/or associations that can be factored out into superclasses

### Classes should be in the same package when they ...

- [1] Are in the same subject area
- [2] Are in same class hierarchy
- [3] Participate in the same use cases
- [4] Are strongly associated



Package diagram for NexGen POS