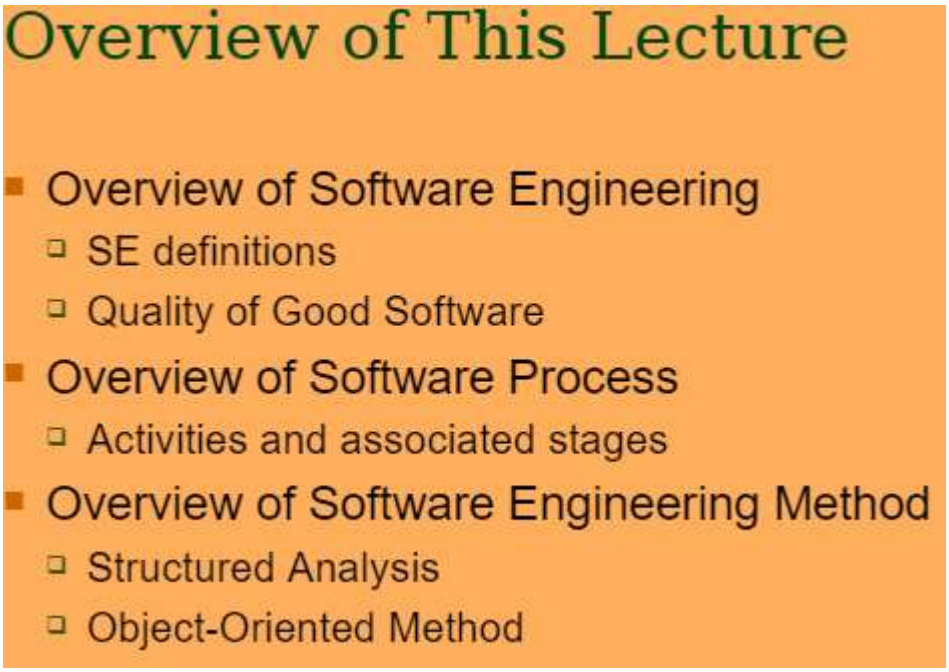
Topic Review

Sunday, March 19, 2023 4:28 PM

L1:



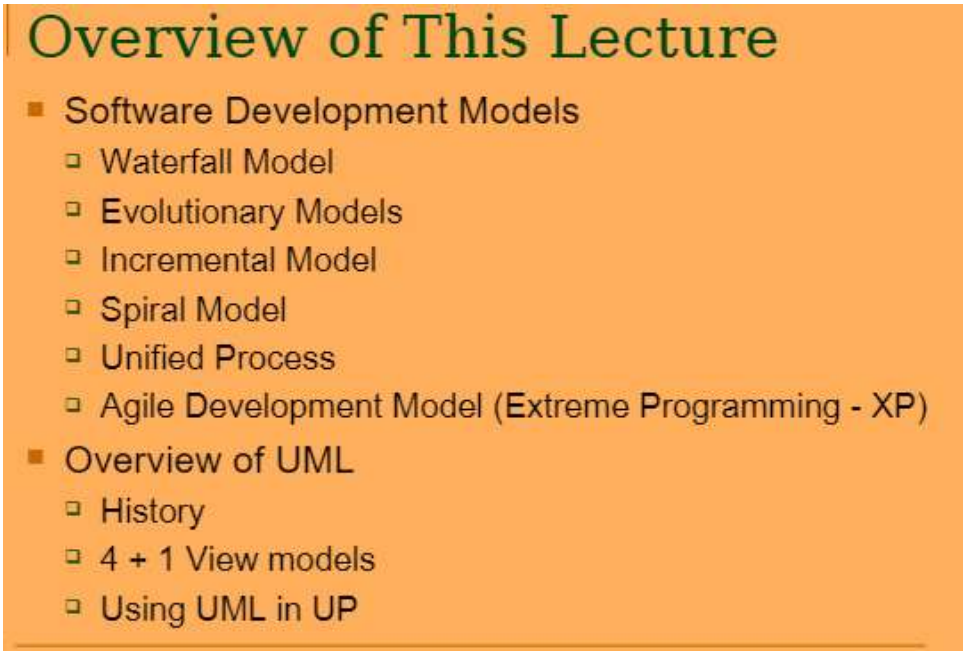
3 major softwares:
- Generic product
- Customized product
- Embedded product

Nature of software
Software dev probs
Known software disaster
"engineering"
Quality of good software
"software engineering"
Myths
Software process/software engineering method
- Software specification
- Software dev
- Software validation
- Software evolution

Modeling the system
- Structure chart

Software Engineering method
Structured analysis
Structured methods
Object-Oriented Methods
In software engineering course
UML Class Icon + Code

L2:



Software Dev Models
UML
UML Not
Use Case View
- Audience – sys analyst, end users, testers
- Usage – sys external...

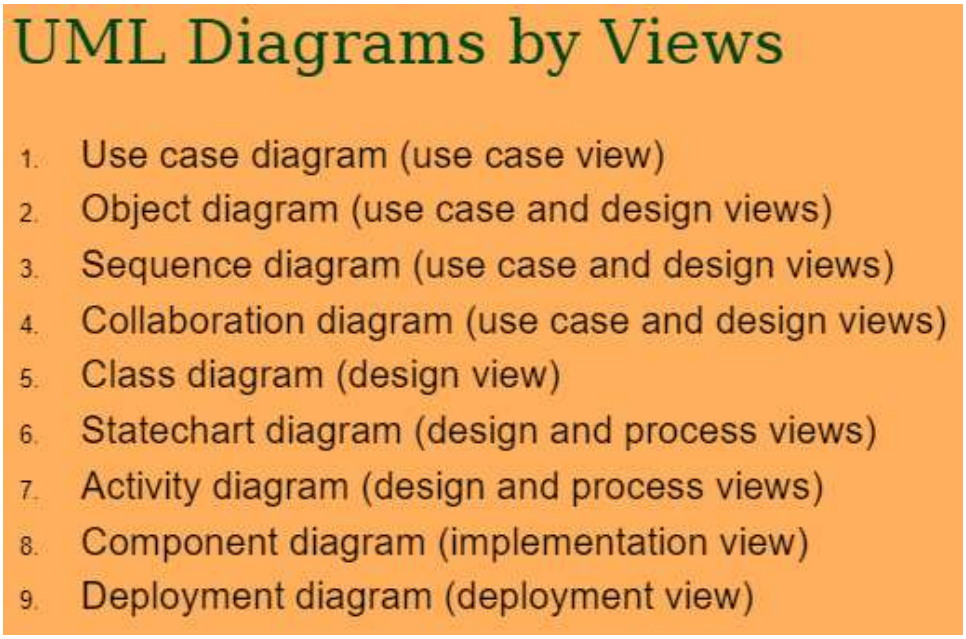
Design View
- Audience – sys analyst, pgrmers
- Usage – logical structures, …

Implementation View
- Audience – sys engineer, tester
- Usage – describe physical components...

Process View
- Audience – sys analyst, pgrmer, tester
- Usage – non-functional req, …

Deployment View
- Audience – sys integrator
- Usage – non-functional, …

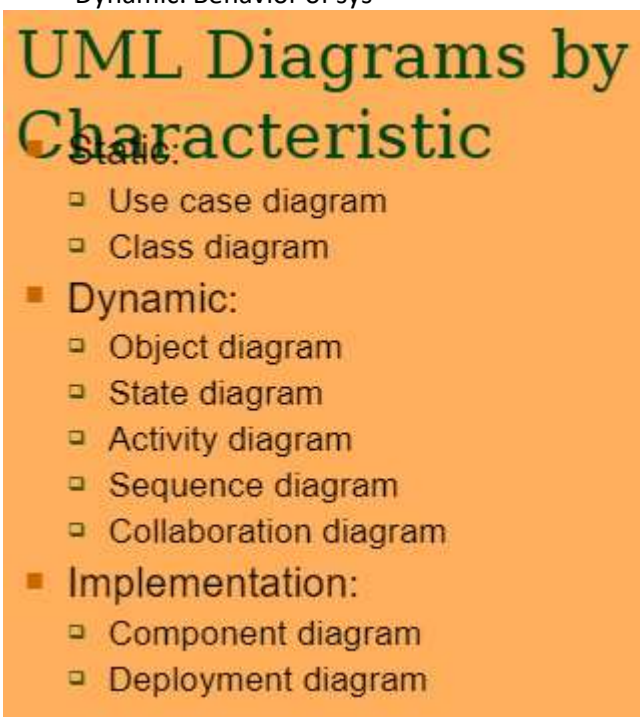
UML Terms

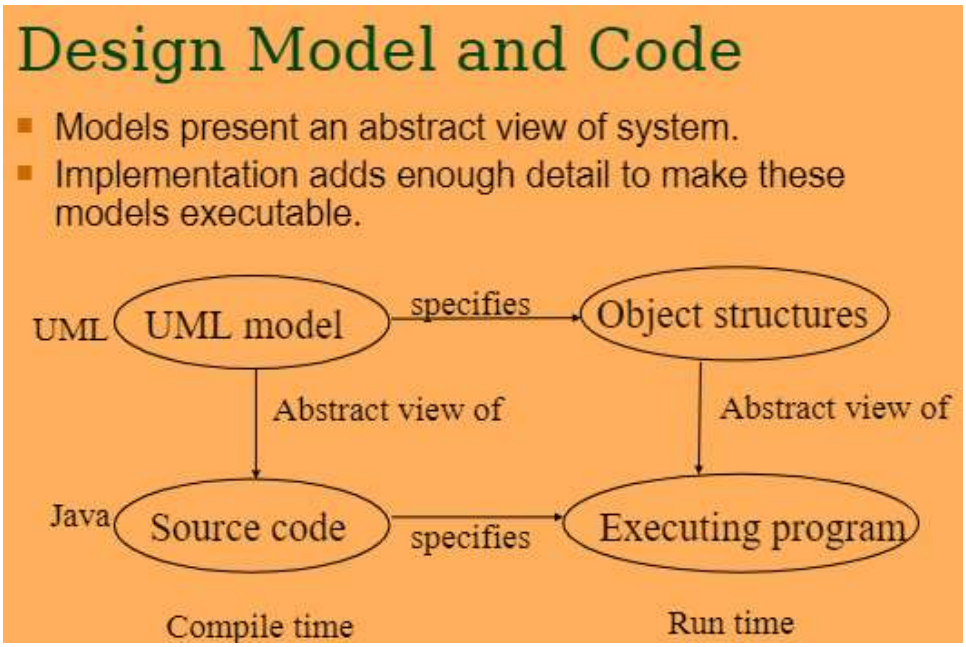


Characteristics

Static: logical structure

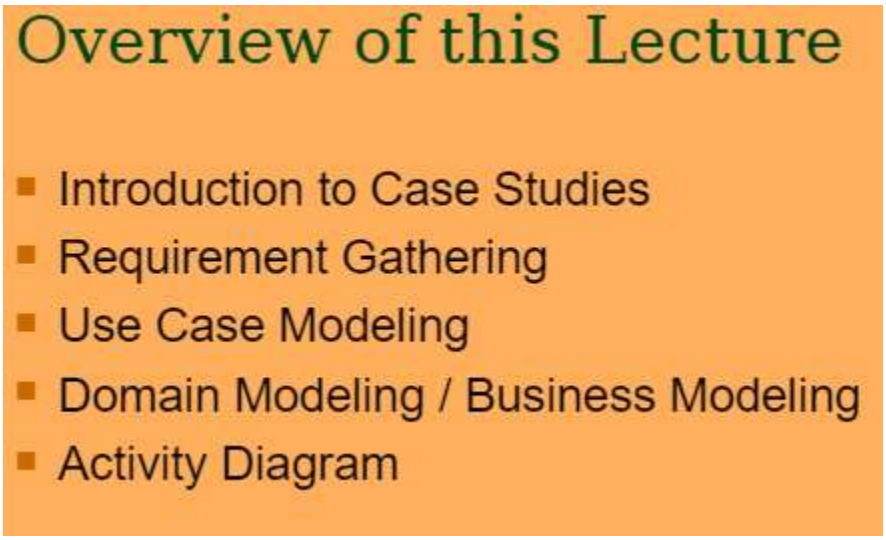
Dynamic: Behavior of sys



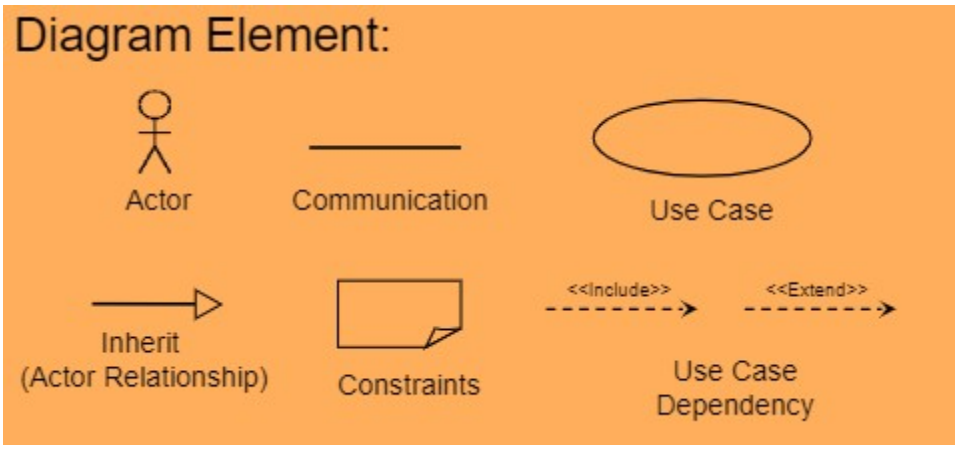


Unified Process (UP) and UML
UP Req & Analysis
UP Realization and Refinement
UP Specifying Behavior
Testing Phases

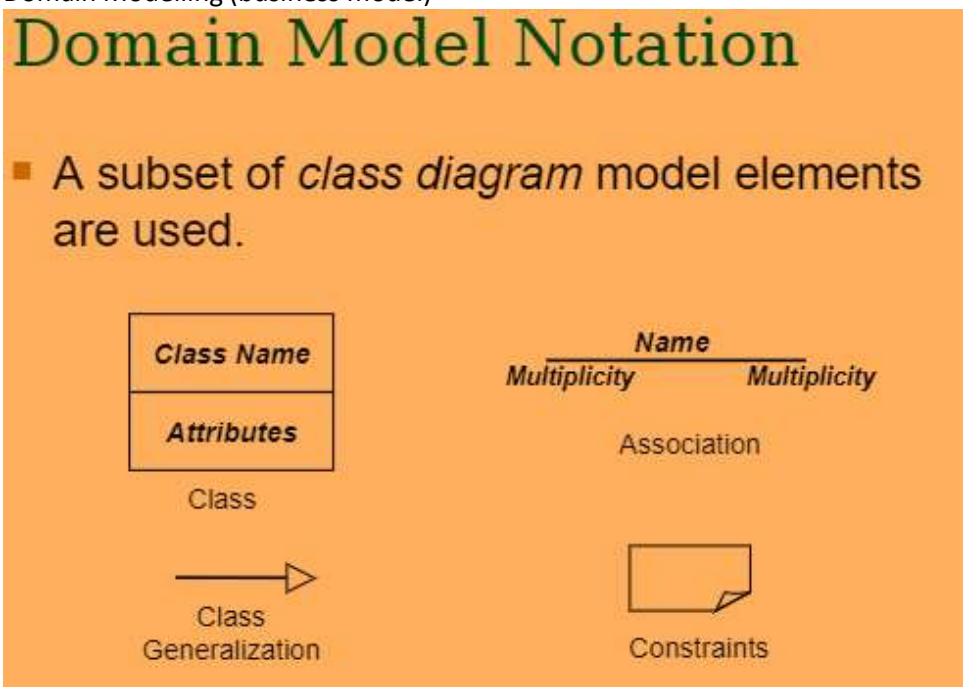
L3:



Case Study Restaurant
Case Study Monopoly
Requirement Analysis
Requirement Overview
Use Case View
Use Case and Scenarios
Actor
Use Case Diagram



Alternate and Exceptional Events
Shared Functionality
Use Case Inclusion
<<include>> dependency
Actor Generalization
Use Case Extension
<<extend>> Dependency
Guidelines
Strengths and Problems
Domain Modelling (business model)



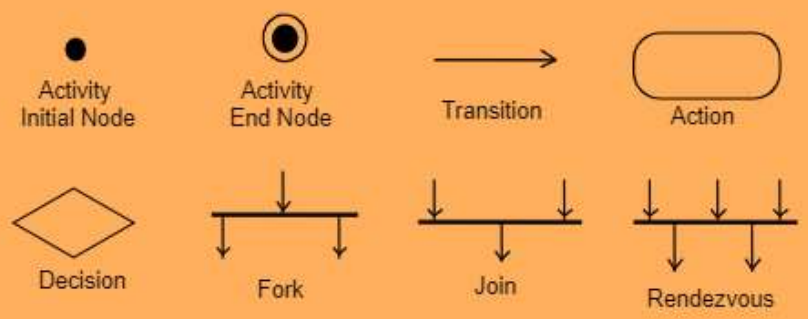
Domain Model Notation

- **Classes** represent real-world entities.
- **Attributes** represent the data held about entities.
- **Associations** represent relationships between the entities.
- **Generalization** can be used to simplify the structure of the model.
- **Constraints** can be used to indicate conditions.

Define a Relationship
Use of Constraints
Use of Generalization
Supplementary Docs
Activity Diagrams
    Useful in:
        Business modelling
        Use cases
        Design

Activity Diagrams (cont)

- Diagram Elements:



Activity Diagrams (cont)

- **Action:**
  - Fundamental block in an activity diagram.
  - Represents a unit of work (something is done).
  - Automatic transition upon completion.
- **Transition:**
  - Represents the control flow: it is simply a movement between *actions*.
- **Initial and End Node:**
  - Show the beginning and ending points in an activity diagram.

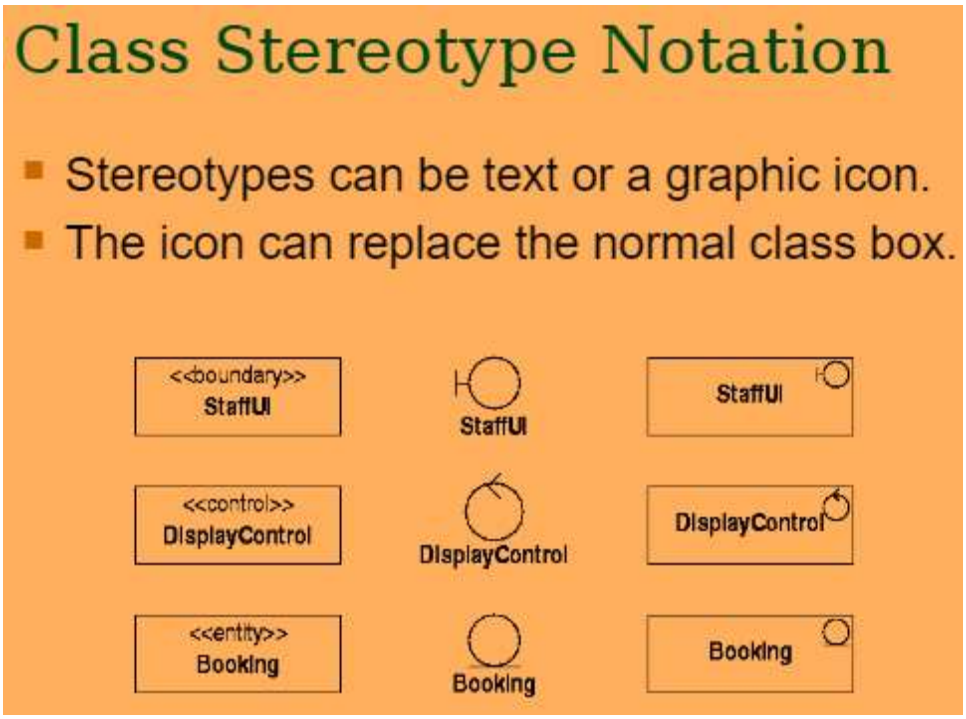
Concurrent Actions
Representing Concurrency
Swimlanes

L4:

Overview of This Lecture

- Object-Oriented Analysis
  - Software Architecture
  - Use Case Realization
    - Realizing Use Cases into Sequence Diagrams
  - Domain Model Refinement
    - Refining the Domain Model into a Partial Class Diagram

Analysis: Overview
Analysis vs Design
Software Architecture
    Cohesion – set of things that work well together
    Coupling – inter-dependency btwn 2 things
Software Architecture in Minesweeper (w/ code)
Layered Architecture
UML Package Diagram
Analysis Class Stereotypes



Use Case Realization
Sequence Diagram
    Code
System Sequence Diagram (SSD)
Refining Association
    Refined the domain model, sequence diagram, class hierarchy

L5:

Design: Overview
Design and Layered Architecture
Design: Presentation Layer
    Appropriate type of UI
    Draft UI
    Study interaction to perform each of use cases
    Derive classes needed
Storage Layer: Revisited
Designating Persistent Classes
Saving & Reloading an Object
Preserving Association

Simple example
Shows memory view
Observations
Persistency Architecture
Storage Layer: Summary
Design Phase: rest of the journey
Detailed Class Design
Class Diagram: Review
Class Multiplicity
Attribute Type
Enumeration
Attribute Scope
Attribute Visibility
Attribute Multiplicity
Operations
Constructing Class Diagram
Object Diagram

L6:

Class Generalization
Meaning of Generalization
Substitutability
Abstract Class
Generalization Hierarchies
Inheritance
Modifying Subclasses
Abstract Operations
Association: Aggregation (hollow diamond)
Cyclic Object Structures
Property of Aggregation
Association: Composition (filled diamond)
Component Relationship
Associations and Composites
Composite Boundary
Association: Review
Properties of Links
Association Class
Association Class Property
Reification
N-ary Association

Qualified Associations
Interface
Realizing an Interface
Interface Dependency
Evolution of Class Diagram

L7:

Interaction Diagrams
Collaboration Diagrams
Classifier Roles
Roles and Objects
Association Role
Association Stereotypes
- Base Association - <<Association>>
- Parameter - <<parameter>>
- Local Instantiation - <<local>>
- Global Variable - <<global>>
- Self-Directed - <<self>>

Sequence Diagram: Review
Collaboration Diagram
Collaboration vs Sequence Diagram
Hierarchical Numbering
Interaction Diagram: Additional Notations
Sequence Diagram: Object Creation
Collaboration Diagram: Object Creation
Sequence Diagram: Object Destruction
Collaboration Diagram: Object Destruction
Role Multiplicity
Iterated Messages
Multiobject
Property of Multiobject
Sequence Diagram: Conditional Message
Alternative Flows
Sequence Diagram: Alternative Flow
Sequence Diagram: Message to Self
Collaboration Diagram: Message to Self

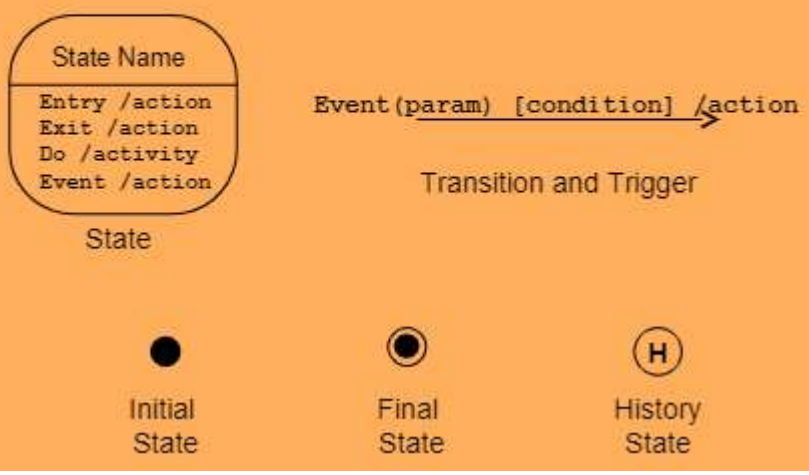
L8:

Overview of This Lecture

- State diagrams (a.k.a. statecharts):
  - State-dependent behaviors, events, transitions
  - Initial and final states, guard conditions
  - Actions, activities, composite states, history states

Object Behavior
Specifying Behavior in UML
State-dependent Behavior
CD Example
UML State Chart Semantics
Statechart: Deciding States
Statechart: Identifying Events
Initial and Final States
Non-Deterministic System
Guard Condition
Actions
Action: Entering/Exiting a State
Action and Activity
Completion Transition
Internal Transition
Composite State
Composite State: Additional Property
History State

Statechart Notation Summary

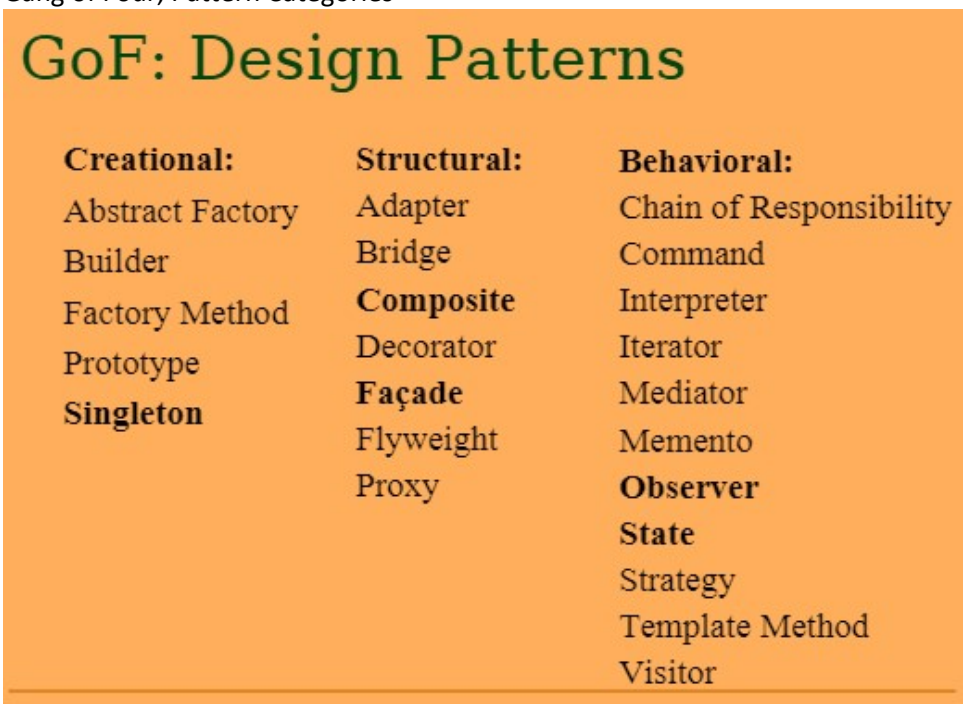


Steps to Make Statechart
Refining Statechart
Adding another interaction
Adding second interaction
Integrating the Interactions
Time Events
Activity States

Returning Change
Statecharts vs Activity Diagrams

L9:

Design Pattern: Intro
Design Pattern Def
Design pattern: Usefulness
Gang of Four, Pattern Categories



Singleton Pattern
Abstract-Occurrence
General-Hierarchy
Composite Pattern
Façade Pattern

Player-Role Pattern
State Pattern
Player Role vs State Patterns
Models, views, and Controllers (MVC)
MVC: Model
MVC: View
Update of View
MVC: Controller
Interaction in MVC
Observer Pattern
Design pattern: Recommendations