CSE 223: Programming -2
03-OOD & UML (II)

Prof. Dr. Khaled Nagi

Department of Computer and Systems Engineering,

Faculty of Engineering, Alexandria University, Egypt.

Object Oriented Design (OOD)

OOA and OOD

- **OOA** is concerned with modeling the application domain
  - Application domain represents all aspects of the user's problem (physical environment, users, work processes)
  - Analysis models do not focus on implementation
  - Interface details, network communication, and database storage are not represented
- **OOD** is concerned with modeling the solution domain
  - Solution domain is the modeling space of all possible systems
  - Modeling the solution domain represents the system design and object design activities of the development process

Cohesion and Coupling

- **Coupling**
  - the degree to which one class is connected to or relies upon other classes.
- **Cohesion**
  - the degree to which the attributes and behaviors of a single class are related to each other.
- The two goals of software design are:
  - **low** coupling and **high** cohesion.
- Allows for object **reuse**.

Cohesion

- The degree of interaction within an object
- Seven categories or levels of cohesion (non-linear scale)

7.	Informational cohesion	(Good)
6.	Functional cohesion	
5.	Communicational cohesion	
4.	Procedural cohesion	
3.	Temporal cohesion	
2.	Logical cohesion	
1.	Coincidental cohesion	(Bad)

Cohesion levels

- Coincidental Cohesion
  - A module has coincidental cohesion if it performs multiple, completely unrelated actions
    - Read_file; Calculate_ square_root; Convert_string;
  - It degrades maintainability
- Logical Cohesion
  - A module has logical cohesion when it performs a series of related actions, one of which is selected by the calling module
    - A module performs buy a painting, sell a painting, produce a report
    - A module that edits insertions, deletions, and modifications of a file record
  - The interface is difficult to understand
  - Difficult to maintain
- Temporal Cohesion
  - A module has temporal cohesion when it performs a series of actions related in time
  - Initialize_object_1; Initialize_object_2; Initialize_object_3;

Cohesion levels

- Procedural Cohesion
  - A module has procedural cohesion if it performs a series of actions related by the procedure to be followed by the product
  - Read_file; Edit_file; Save_file;
  - The actions are still weakly connected
- Communicational Cohesion
  - A module has communicational cohesion if it performs a series of actions related by the procedure to be followed by the product, but in addition all the actions operate on the same data
  - Read_data; Modify_data; Save_to_database
  - Still lack of reusability
- Functional Cohesion
  - A module with functional cohesion performs exactly one action
  - Calculate_pi; Read_file; Format_file;
- Informational Cohesion
  - it performs a number of actions, each with its own entry point, with independent code for each action, all performed on the same data structure

Coupling

- The degree of interaction between two objects
  - Five categories or levels of coupling (non-linear scale)

5.	Data coupling	(Good)
4.	Stamp coupling	
3.	Control coupling	
2.	Common coupling	
1.	Content coupling	(Bad)

Coupling Levels

- Content Coupling
  - Two modules are content coupled if one directly references contents of the other
  - Go to line 10 of module M;
- Common Coupling
  - Two modules are common coupled if they have write access to global data
  - Programming with FORTRAN
- Control Coupling
  - Two modules are control coupled if one passes an element of control to the other
  - Module A calls Module B and pass a flag; if the flag is true, module A print “X”, else module A print “Y”

Coupling Levels

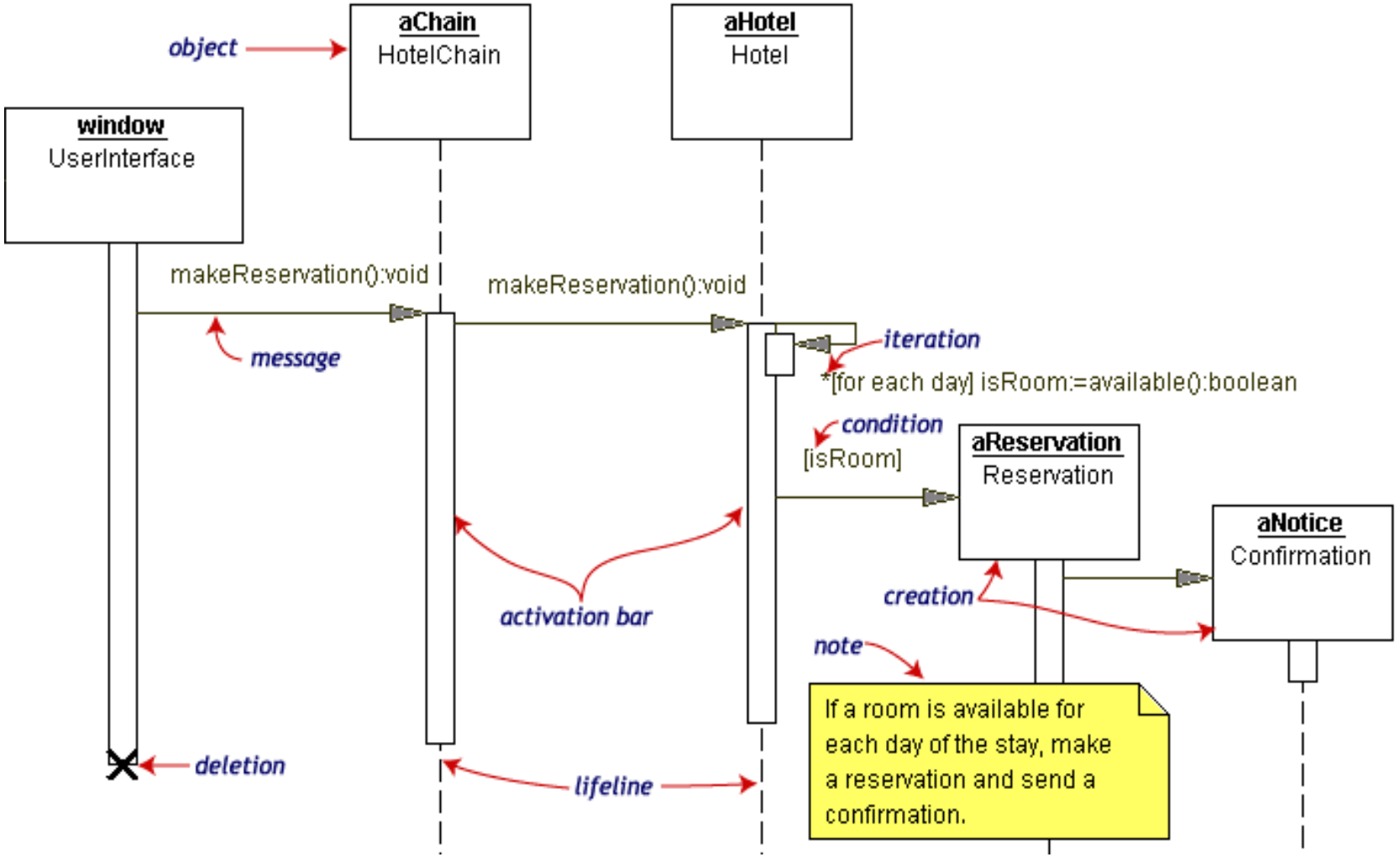
- Stamp Coupling
  - Two modules are stamp coupled if a data structure is passed as a parameter, but the called module operates on some but not all of the individual components of the data structure
  - It is not clear, without reading the entire module, which fields of a record are accessed or changed
  - Difficult to understand
  - More data than necessary is passed
- Data Coupling
  - Two modules are data coupled if all parameters are homogeneous data items (simple parameters, or data structures all of whose elements are used by called module)
  - Maintenance is easier

Unified Modeling Language (UML)
… continued

Sequence Diagram

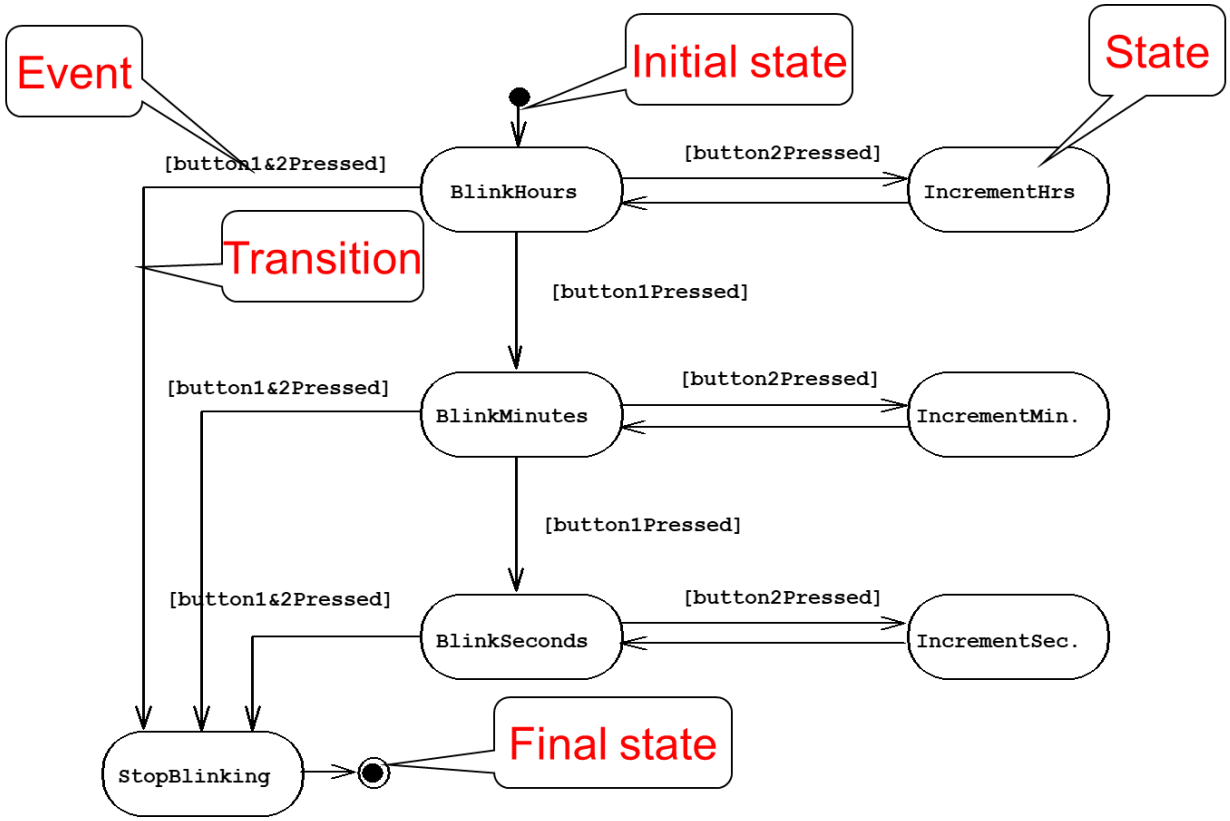
- UML sequence diagram represent behavior in terms of interactions.
- Useful to find missing objects.
- Time consuming to build but worth the investment.
- Complement the class diagrams (which represent structure).
- Users and objects represented by vertical lines
- An object is represented by the underlined name of the class in lowercase
- Start by listing actors in scenario from OOA
- Insert each event from scenario into sequence diagram

Sequence Diagram



State Chart Diagrams

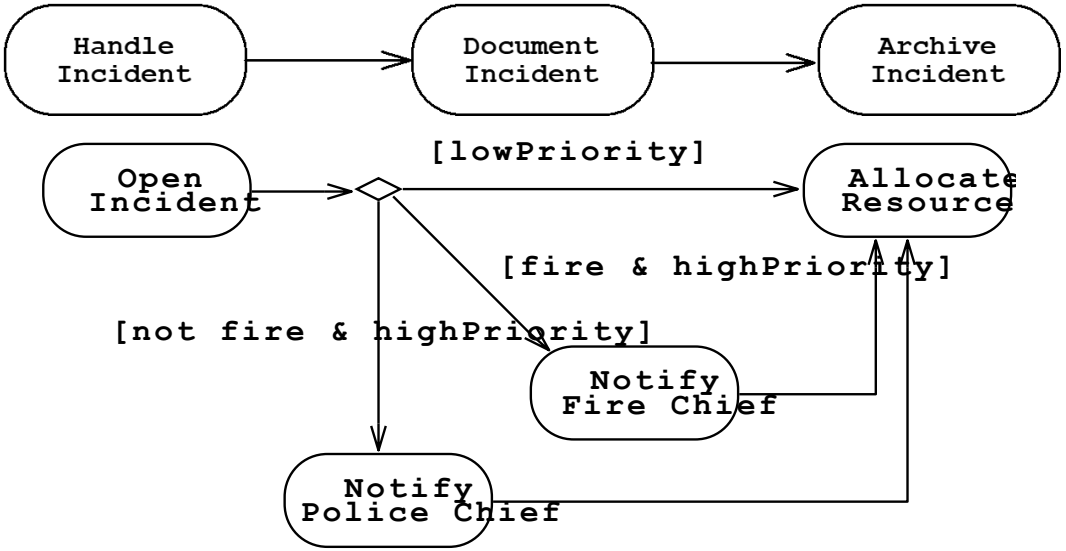
- do/action, activity, entry, exit – see textbook
- Hierarchical states
- In this class, we focus on the basics

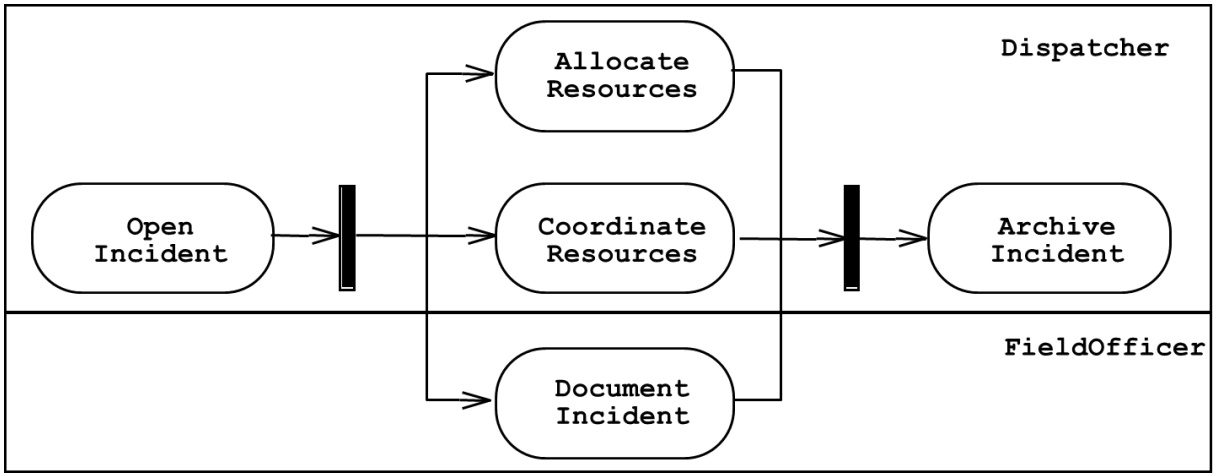


Activity Diagrams

- An activity diagram shows flow control within a system
- 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

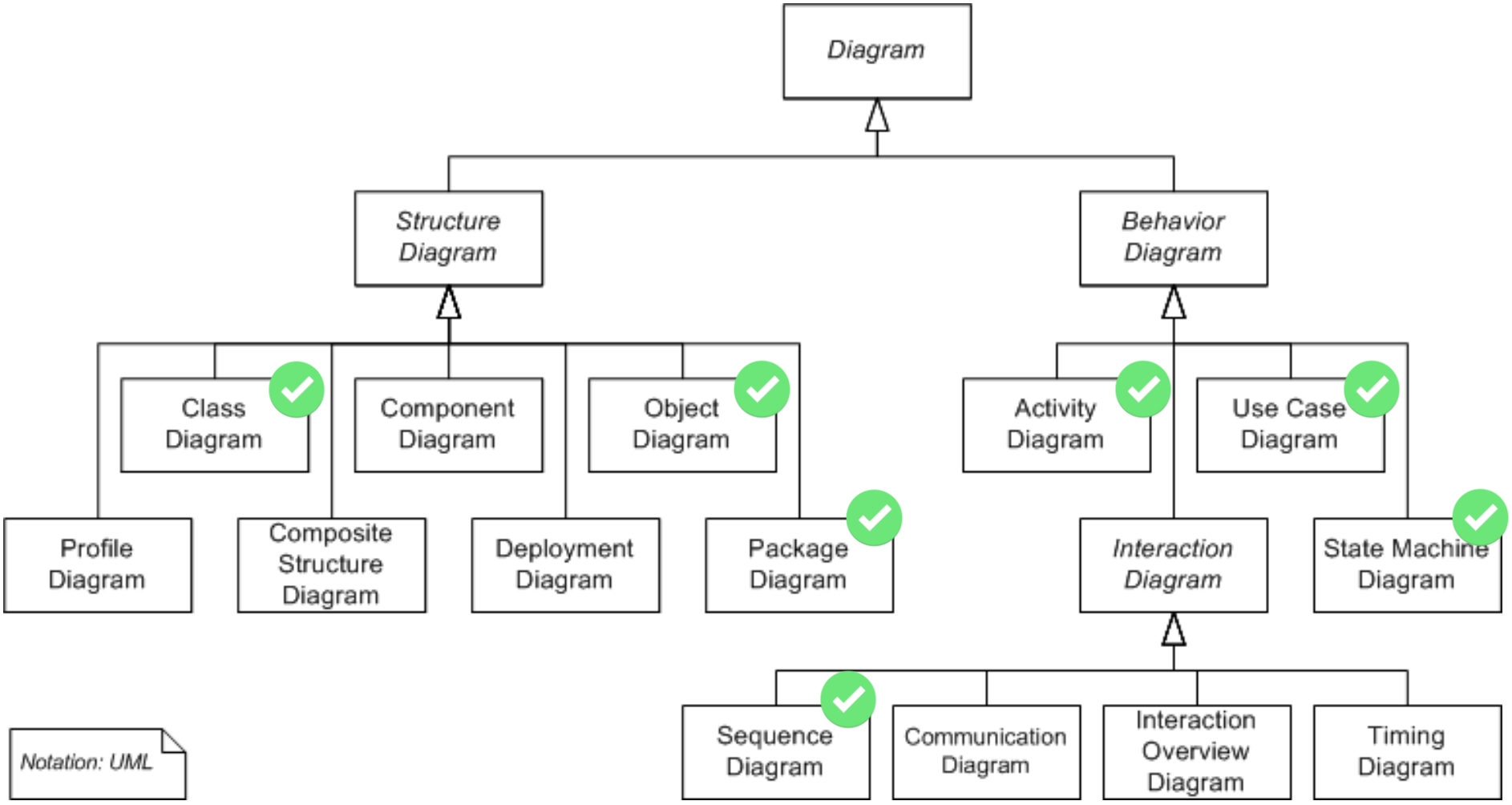
Activity Diagrams





Object Diagram (double check)

- Object Diagram shows the relationship between objects.



What should be done first? Coding or Modeling?

- It all depends….
- **Forward** Engineering
  - Creation of code from a model
  - Greenfield projects
- **Reverse** Engineering
  - Creation of a model from code
  - Interface or reengineering projects
- **Roundtrip** Engineering
  - Move constantly between forward and reverse engineering
  - Useful when requirements, technology and schedule are changing frequently