
EECE 276

Embedded Systems

Object-oriented design techniques

Object-oriented Programming

- Data abstraction/encapsulation
 - » Data can be manipulated only by specific operations
- Inheritance
 - » Object classes are organized into an inheritance (generalization/specialization) hierarchy, where lower-level classes inherit properties of higher-level classes
- Polymorphism
 - » Operations on objects can use the base class' interface, the actual implementation is delegated to the specific object used
- Messaging
 - » Objects interact through method calls

Good points of OO

- Open-closed principle: classes are closed to *modification* but open to *extension*
- Once and only once: any aspect of a software system is captured only once, in one place.
- Dependency inversion: high-level modules should not depend on low-level modules – both should depend on abstractions of those modules.
- Liskov substitution principle:
 - » If for each object $o1$ of type S there is an object $o2$ of type T such that for all programs P defined in terms of T , the behavior of P is unchanged when $o1$ is substituted for $o2$, then S is a subtype of T .

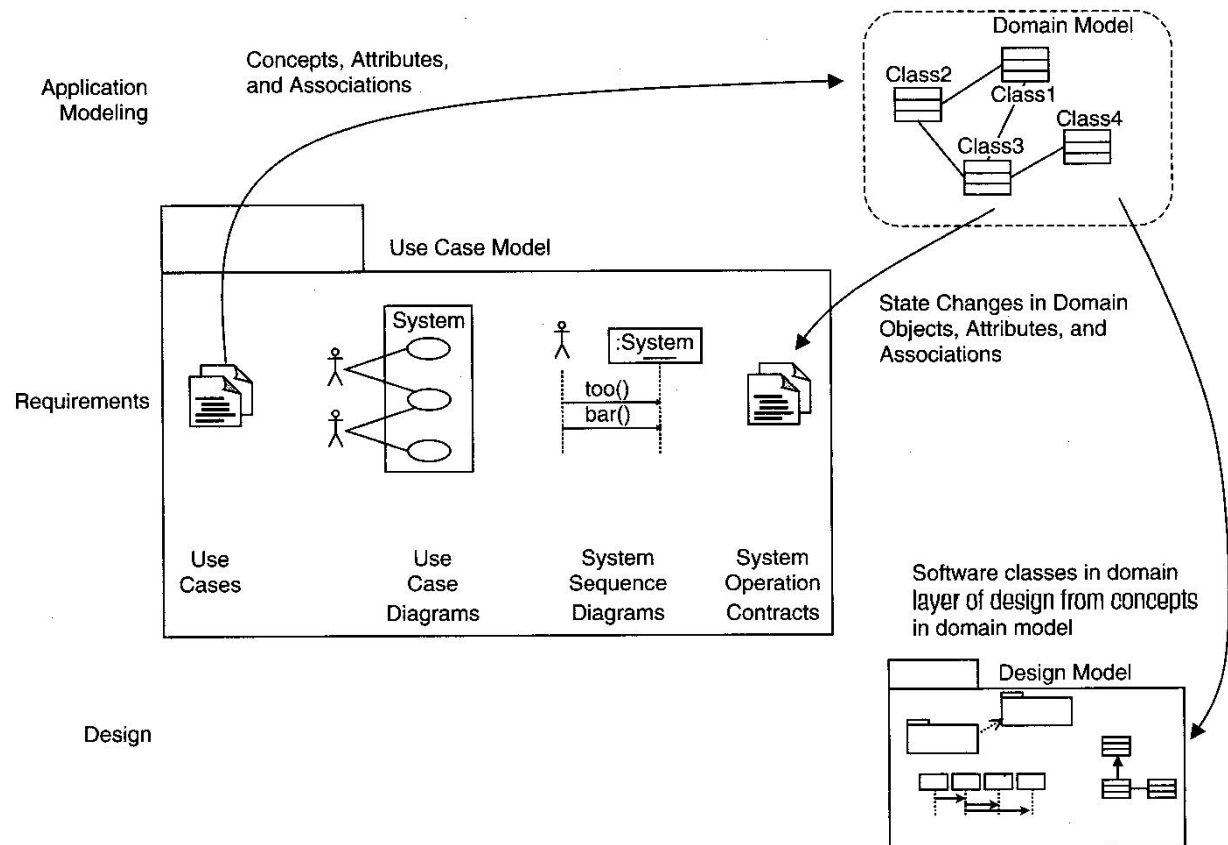
(Subtypes are interchangeable with their base types!)

OO Design Patterns

- GOF: a prototypical solution to a recurring software design problem.
 - » “Design Patterns” book by the ‘Gang-of-Four’
- Typically involves a collection of classes with fragments of behavior
- Categories (GOF)
 - » Creational: Abstract Factory, ...
 - » Behavioral: Observer, ...
 - » Structural: Composite,

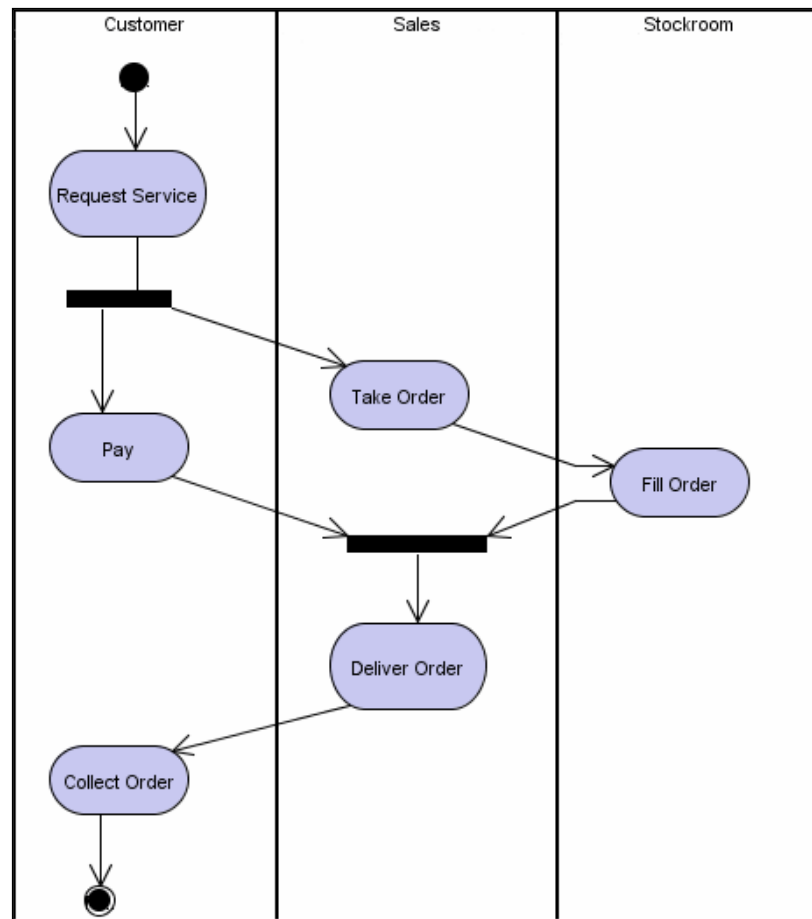
OO Design using UML

- UML: a collection of graphical languages



UML Diagrams for Real-time Systems

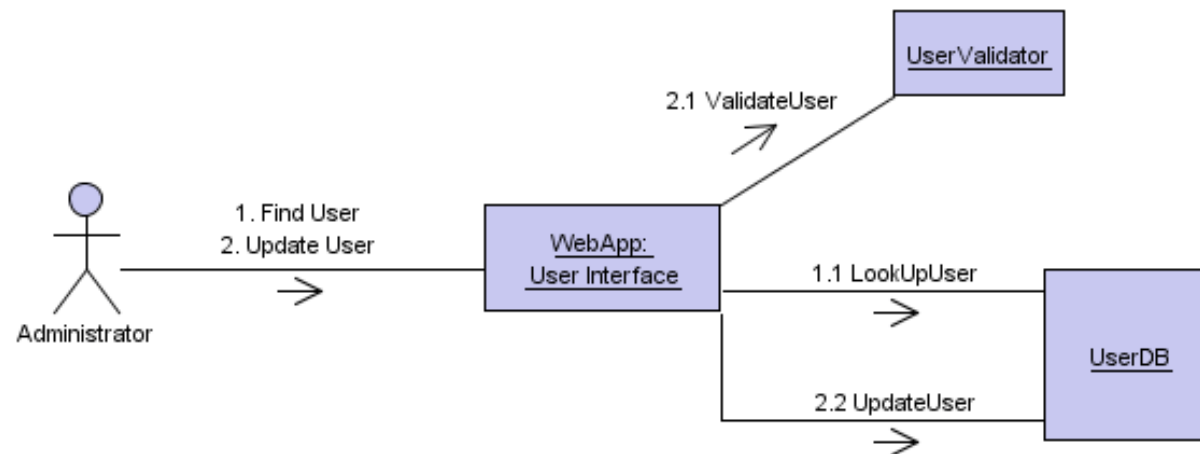
- Activity diagrams
Like flowcharts, but
can represent
concurrency
- Class diagram
(seen before)



UML Diagrams for Real-time Systems

- Collaboration diagrams

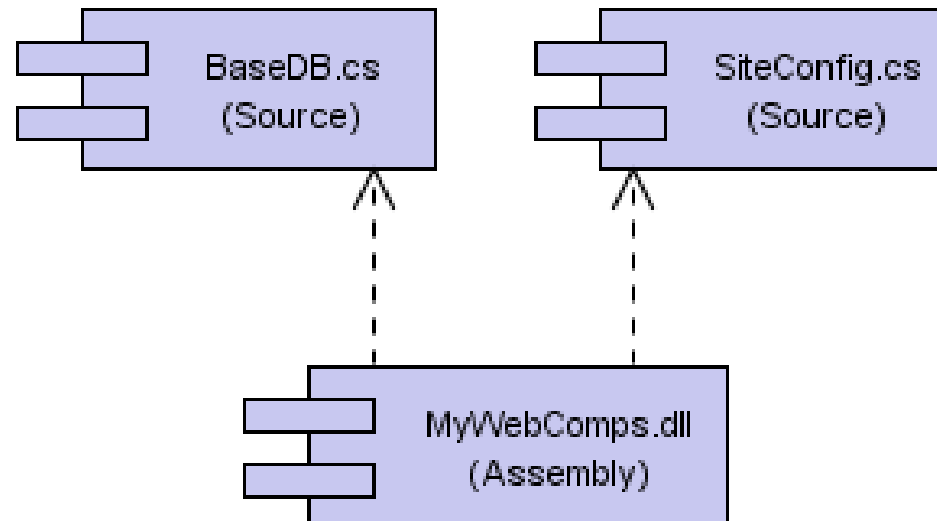
Messages passed between classes along associations



UML Diagrams for Real-time Systems

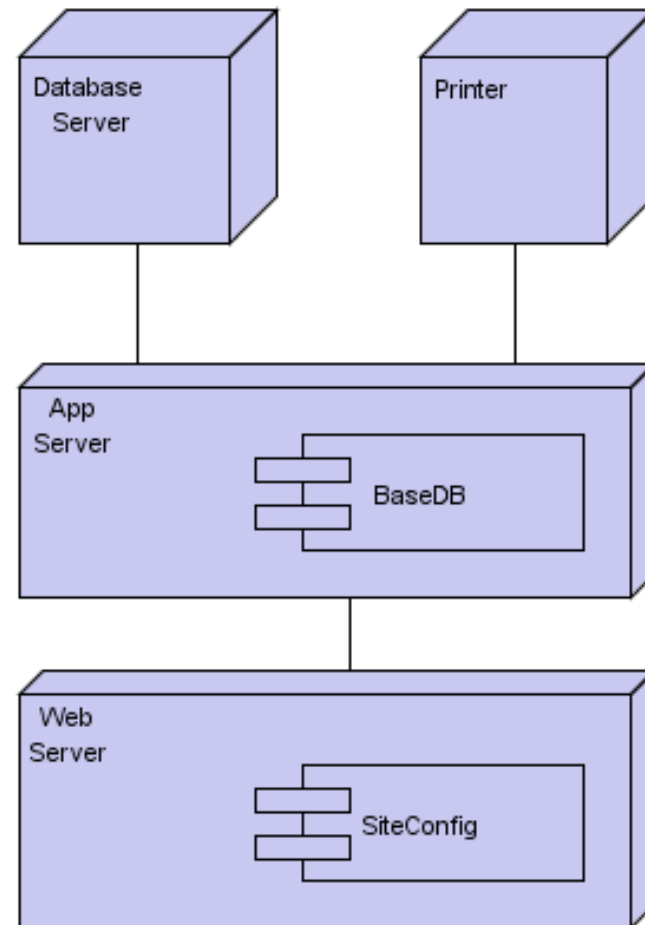
- Component diagrams

Components and their interfaces and dependencies



UML Diagrams for Real-time Systems

- **Deployment diagrams**
How the components will be deployed (on processing nodes)

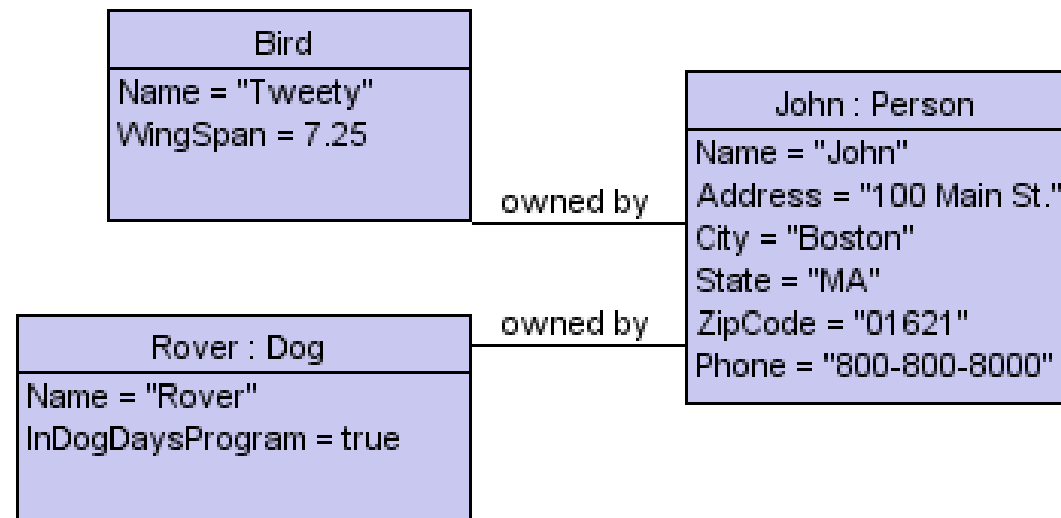


UML Diagrams for Real-time Systems

- Object diagrams

Static objects of the system

Follows class diagram



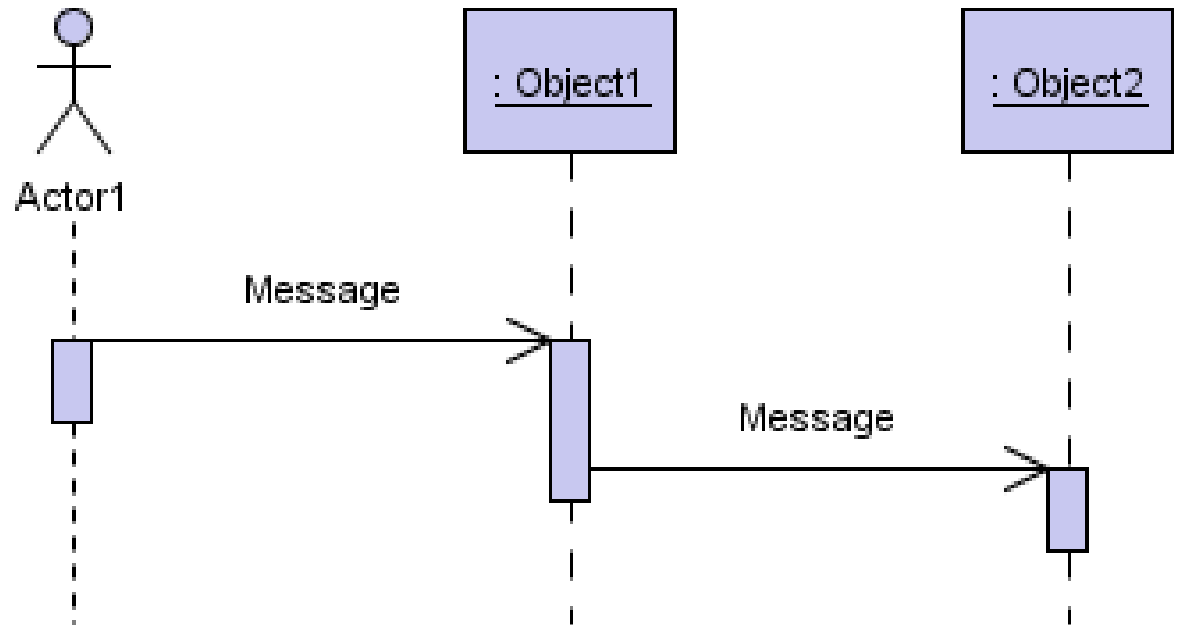
UML Diagrams for Real-time Systems

- Sequence diagrams

Objects, links, messages

Message sequences shown in *time*

UML does not provide good facilities expressing real-time properties!
(see ROOM, HRT-HOOD)



SA vs. OO design

	SA	OOD
System components	Functions	Objects
Data processors Control Processes Data stores	Separated through internal decompositions	Encapsulated within objects
Characteristics	Hierarchical structure Classification of functions Encapsulation of knowledge within functions	Inheritance Classification of objects Encapsulation of knowledge within objects