Object-Oriented Design III
Modeling Dynamic Aspects of Systems

*Interaction diagrams*: set of objects and their relationships including messages that may be dispatched among them

- *Sequence diagrams*: time ordering of messages
- *Collaboration diagrams*: structural organization of objects that send and receive messages

*Activity diagram*: flow chart showing flow of control from activity to activity

*Statechart diagram*: models a state machine
Bouncing Ball Diagrams

Example: http://www.cs.cornell.edu/

domain name

TCP connection

HTTP get

Client                                Servers
Actions on Objects

- call
- return
- send
- create
- destroy

returnCopy(c)

okToBorrow()

status

notifyReturn(b)

<<create>>

<<destroy>>

local

asynchronous

signal

stereotypes
Links

**class**

- LibraryMember
  - +borrowCopy()
  - +returnCopy()

**object**

- libMem:LibraryMember
  - link: borrowCopy(c)
    - message: borrowCopy(c)

**association**

- 1: on loan
- 0..*: Copy

**link**

- c:Copy
Sequence Diagram: Change in Cornell Program

Cornellian

1 : getName()  
  1.1 : name

2: new PhDStudent(name)

3: <<destroy>>

:MEngStudent

:PhDStudent

sequence numbers added to messages
Sequence Diagram: Borrow copy of a Book

borrow(theCopy)

okToBorrow

borrow

borrow
Sequence Diagram: Painting Mechanism

:Thread

:Toolkit

:ComponentPeer

target: HelloWorld

run

run

run

callbackLoop

handleExpose

paint
Activity Diagram: Process Modeling

1. Release work order
2. [materials not ready] → Reschedule
3. [materials ready] → Assign tasks

Branch

Guard expression
ActivityDiagram: Parallel Activities

start state

fork

Decompress

stream video

stream audio

join

stop state
State Diagram for class Book

State diagram for class Book
Implementation Modeling

Subsystem

A grouping of elements that specifies what a part of a system should do.

Component (UML definition)

"A distributable piece of implementation of a system, including software code (source, binary, or executable) but also including business documents, etc., in a human system."

A component can be thought of as an implementation of a subsystem.
Component Diagram

hello.hml

HelloWorld.class

executable
component

hello.jpg

hello.java
Components and Classes

agent.dll

AgentAction

Policy

PatternSearch
Components and Classes

agent.dll

Realizes
AgentAction
PatternSearch
Policy

extended component
Components and Classes

**Classes** represent logical abstractions. **Components** represent physical things.

**Components** may live on **nodes**.

**Classes** have attributes and operations directly. **Components** have operations that are reachable only through **interfaces**.
Interfaces
Application Programming Interface (API)

API is an interface that is realized by one or more components.

```
simulation.exe

IModels  ILighting
```

IRender
Components allow system to be assembled from binary replaceable elements.

- A component is physical -- bits not concepts
- A component can be replaced by any other component(s) that conforms to the interfaces.
- A component is part of a system.
- A component provides the realization of a set of interfaces.