

ICS 121 Topic 10:
**Object-Oriented
 Analysis and Design**

Object Modeling Technique
 Unified Modeling Language
 Class Modeling
Dynamic Modeling

Dynamic Modeling

- Prepare scenarios
- Identify events between objects
- Prepare interaction diagram(s) for each scenario
- Build a state diagram
- Match events between objects to verify consistency

Topic 10 OOAD 2

Dynamic Model Diagrams

- The dynamic model tracks behavior over time
 - described in terms of change in objects or event sequences between objects
- Interaction Diagrams model how groups of objects interact in some behavior (often a single use case)
 - objects and messages passed within a use case scenario
 - Sequence Diagrams
 - visually ordered trace of events (messages passed) between objects
 - Collaboration Diagrams
 - numbered trace of events (messages passed) between objects laid out spatially
- State Diagrams model events, states, and state transitions
 - a scenario is a path through the state diagram

Topic 10 OOAD 3

Events and Scenarios

- An *event* is something that occurs between objects
 - events have attributes, which are the information transferred from one object to another
- A *scenario* is a specific sequence of events representing a path through a system's states
- Legitimate scenarios
 - common paths (e.g. frequently used functionality)
 - error conditions and known exceptions
- An *event trace* extends a scenario to clarify interactions between objects
- Event traces are modeled in *interaction diagrams*

Topic 10 OOAD 4

Event classes and attributes

<ul style="list-style-type: none"> • Event Classes <ul style="list-style-type: none"> – airplane departs (airline, flight number, city) – mouse button pushed (button, location) – phone receiver lifted – digit dialed (digit) 	<ul style="list-style-type: none"> • Events <ul style="list-style-type: none"> – United Flight 23 departs from Rome – right mouse button pushed at (29, 30) – phone receiver lifted – digit dialed (2)
---	--

Topic 10 OOAD 5

An example scenario

- Scenario for a phone call
 - caller lifts receiver
 - dial tone begins
 - caller dials digit (7)
 - dial tone ends
 - caller dials digit (3)
 - caller dials digit (5)
 - caller dials digit (3)
 - specified phone rings
 - callee answers phone
 - etc.

Topic 10 OOAD 6

UML Sequence Diagrams

- objects are boxes at top of lifelines
- lifelines are dashed vertical lines
- events are horizontal arrows, labeled with message name and possibly a [condition] or * (iteration)
- arrow heads indicate sender/receiver
- time passes from top to bottom

Topic 10 OOAD 7

Sequence Diagram: example

Topic 10 OOAD 8

UML Collaboration Diagrams

- objects are boxes
- events are arrows between objects, labeled with message name and possibly a [condition] or * (iteration)
- arrow heads indicate sender/receiver
- sequence/time indicated by ordering of events

Topic 10 OOAD 9

Collaboration Diagram: example

Topic 10 OOAD 10

States and Transitions

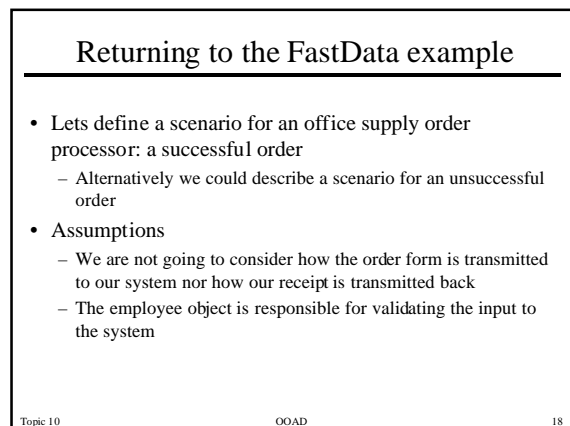
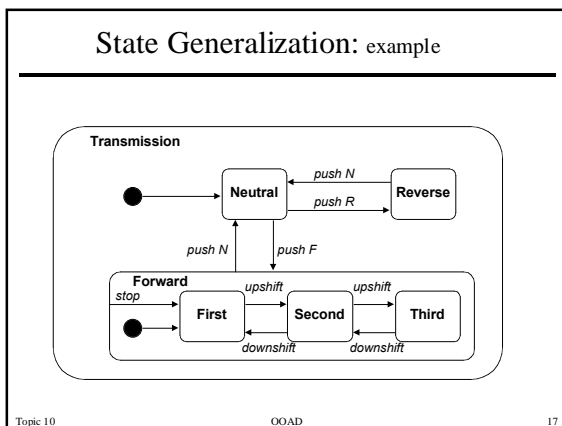
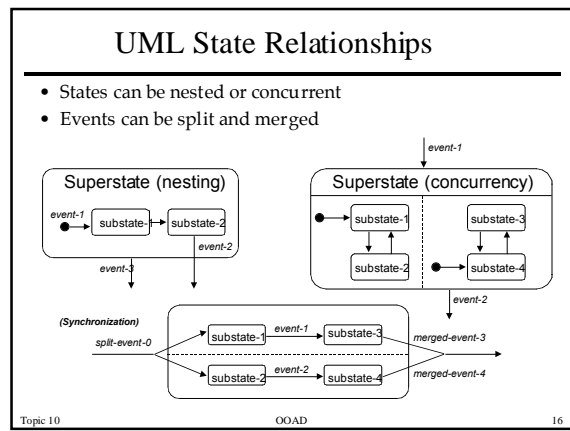
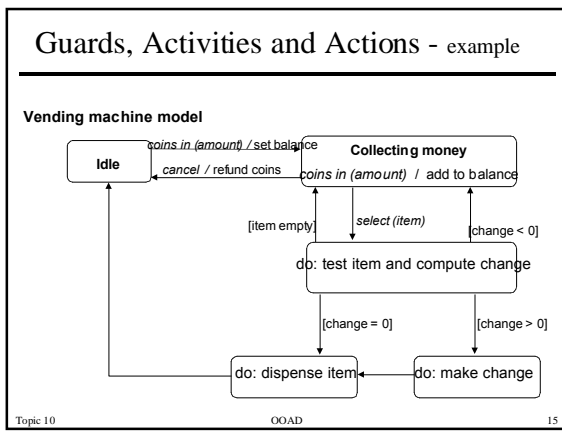
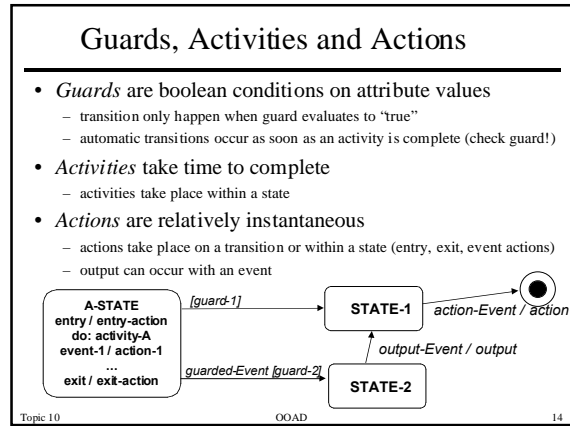
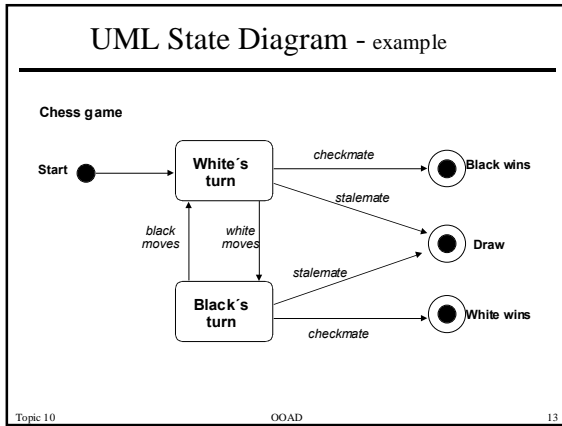
- A *state* is an interval between events
 - it may have an activity that can trigger starting, intermediate and ending events
 - defined in terms of a subset of object attributes and links
- A *state transition* is a change in an object's attributes and links
 - it is the response of an object to an event
 - all transitions leaving a state must correspond to distinct events

Topic 10 OOAD 11

UML State Notation

- nodes represent states: rounded rectangles with state name
 - initial state represented as solid circle
 - final state represented as bull's eye
- edges represent transitions between states and are labeled with an event name (the trigger)

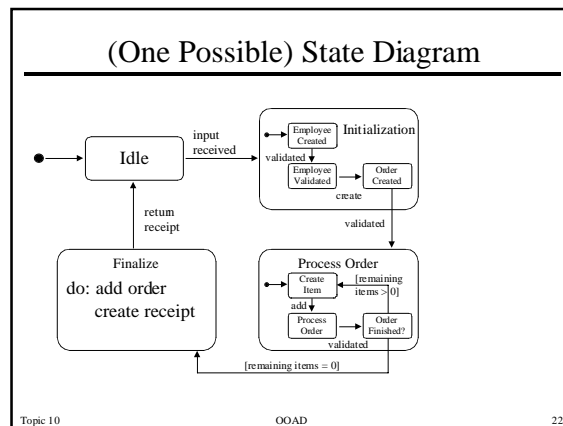
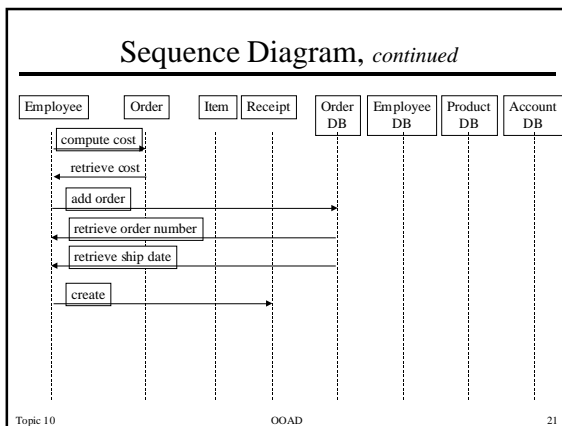
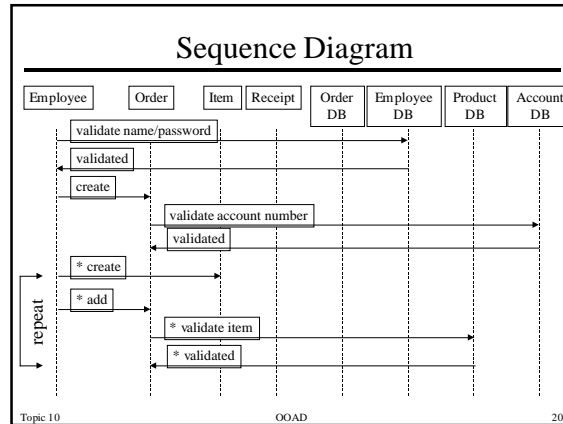
Topic 10 OOAD 12



One Scenario: a successful order

- input received (we don't care how)
- create employee object
- pass input to employee
- validate name and pass word
- create order object
- validate account number
- for each item
 - create item
 - add item to order and validate item
- compute total cost
- add order to order DB and retrieve order number and ship date
- generate receipt
- return receipt (we don't care how)

Topic 10 OOAD 19



OOAD: Steps

- Functional Modeling
- Class Modeling
- Dynamic Modeling

- *Refine the Class Model (add Operations)*

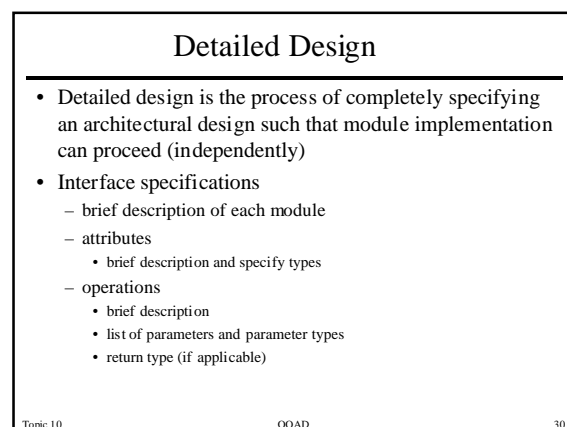
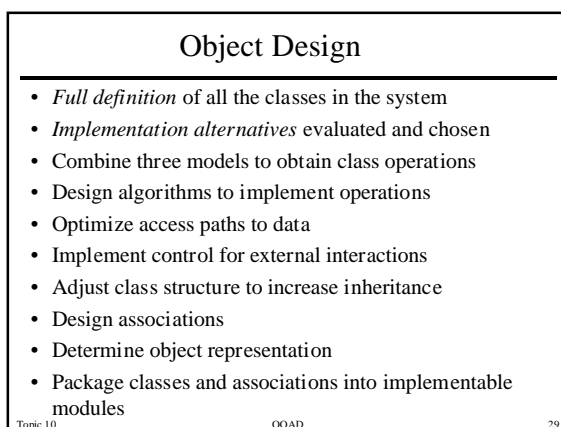
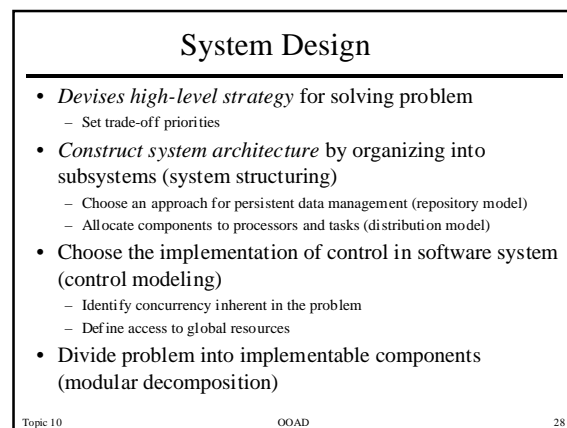
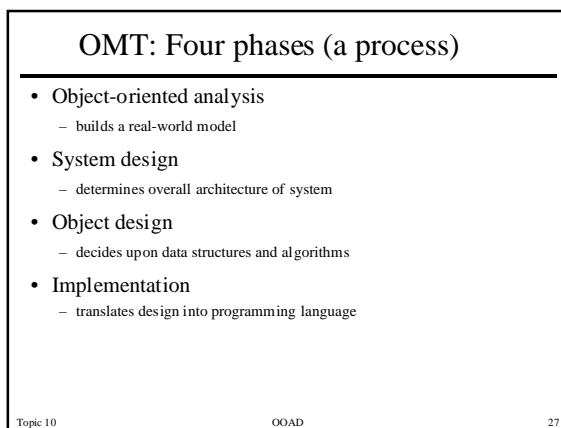
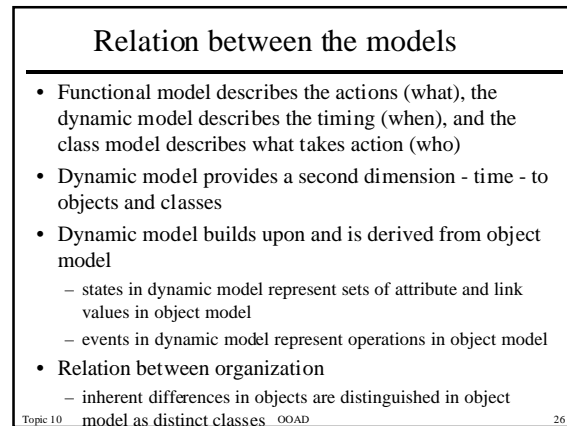
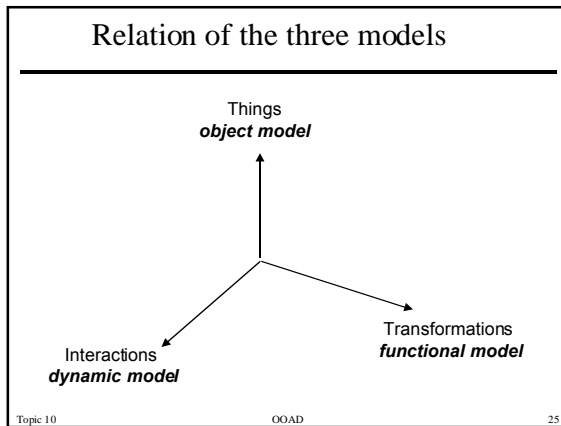
- Iterate and refine all models
 - After the first iteration, steps may occur in parallel or out of order
 - All models must be kept in synch as changes are made

Topic 10 OOAD 23

Add Operations to the Class Model

- From the Class Model:
 - Reading/writing object attributes (e.g., get_width, get_height of Rectangle)
- From Events, State Actions, and Activities in the Dynamic Model:
 - Each event sent to an object => operation (e.g., Vending machine: set_balance)
 - Actions/activities may be operations (e.g., Vending machine: do: test item and compute change)

Topic 10 OOAD 24



Detailed Design, continued

- Algorithm and data structure specification
 - the designer can give hints as to what algorithms or data structures might be most useful for a particular module
 - also, the client may have specified a particular algorithm or data structure that must be used
 - in addition, constraints in the requirements may require one approach over another
 - for instance, implementing a data structure so that it uses the minimum amount of memory possible vs. keeping everything in memory for speed

Topic 10

OOAD

31

Mapping design into code

- Most programming languages provide very similar sets of features
 - user-defined types
 - control structures
 - if...then...else...
 - while x do y
 - for i = 1 to x
 - etc
 - etc.
- This means that, in general, operations can be expressed in many different languages

Topic 10

OOAD

32

Mapping design into code, continued

- Major differences between languages usually fall into these categories
 - compiled vs. interpreted
 - procedural vs. object-oriented
 - general purpose vs. application/domain specific
 - e.g. C++ vs. FileMaker Pro's scripting language
- If a design takes advantage of, or depends on, one or more of these features then certain programming languages have to be excluded from implementation

Topic 10

OOAD

33

Modularity Mechanisms

- One import feature of any programming language is how it can represent modules directly
 - C and C++ have separate header and body files
 - Java has package names and class files
 - Ada has a construct called a package with a specification and body (implementation)
 - etc.
- These features are important since it makes it easier to map the design into code and to trace a code module back to its design counterpart

Topic 10

OOAD

34

Integration Test Plan

- Developed as part of (architectural) design
- Test plan to exercise module interactions
 - actual test data and expected results for each potential module interaction
 - order of test executions
 - completion criteria
 - simple coverage (one test per interaction)
 - input/output coverage (range of values)
 - data flows (flows from user to used and back)
- Basic goal is to test how modules interact with each other and with data under the assumption that they have passed module testing

Topic 10

OOAD

35

Integration Testing

- Testing based on integration test plan after module testing
- Integration is the process by which modules are aggregated to create larger components
- Integration may be determined by uses hierarchy
- Integration testing examines each combination to determine whether it is also correct or to find defects in the interaction between "correct" components

Ensures modules make compatible assumptions

Topic 10

OOAD

36

Integration Test Plan Process

- For a given interaction:
 - Design test cases to test that interaction
 - design typical test case
 - design test cases specific to this interaction
 - design special and boundary value test cases
 - For each test case, provide the "values" of parameters and any environment (e.g., persistent data) required
 - Plan the order of the test cases for this interaction
 - initialize, set-up, process, wrap-up
 - Describe any stubs or drivers required for this interaction
- Plan the order of integration testing
 - top-down
 - bottom-up

Topic 10

combination

OOAD

37

Homeworks 5 and 6

- Builds on Homework 4 (and 1) using OOAD: UML
- Objective
 - Produce an object-oriented design of VirtualMallOnline in two phases
- Techniques
 - Object/Class Diagrams
 - Collaboration Diagrams
 - State Diagrams
 - Detailed Designs

Topic 10

OOAD

38

Something to think about: Structured vs. O-O Design

- What are the major differences between object-oriented design and structured design?
- How does each relate to software engineering principles?
- How does each relate to desirable characteristics of a design?

Topic 10

OOAD

39