Copyright © 2005  
John Wiley & Sons, Inc.

- All rights reserved. Reproduction or translation of this work beyond that permitted in Section 117 of the 1976 United States Copyright Act without the express written permission of the copyright owner is unlawful.
- Request for further information should be addressed to the Permissions Department, John Wiley & Sons, Inc.
- The purchaser may make back-up copies for his/her own use only and not for redistribution or resale.
- The Publisher assumes no responsibility for errors, omissions, or damages, caused by the use of these programs or from the use of the information contained herein.
Objectives

- Understand the basic characteristics of object-oriented systems.
- Be familiar with the Unified Modeling Language (UML), Version 2.0.
- Be familiar with the Unified Process.
- Understand a minimalist approach to object-oriented systems analysis and design.
Basic Characteristics of Object Oriented Systems

- Classes and Objects
- Methods and Messages
- Encapsulation and Information Hiding
- Inheritance
- Polymorphism and Dynamic Binding
Classes and Objects

- **Class** – Template to define specific instances or objects
- **Object** – Instantiation of a class
- **Attributes** – Describes the object
- **Behaviors** – specify what object can do
Classes and Objects

**Classes**

- **Patient**
  - Name
  - Birthdate
  - Address
  - Phone Number
  - + Insert()
  - + Delete()

- **Appointment**
  - Patient name
  - Doctor name
  - Date
  - time
  - + Insert()
  - + Delete()

**Objects**

An instance of the Patient class

- **aPatient : Patient**
  - Name = Theresa Marks
  - Birthdate = March 26, 1965
  - Address = 50 Winds Way, Ocean City, NJ 09009
  - Phone Number = (804) 555-7889

An instance of the Appointment class

- **anAppointment : Appointment**
  - Patient name = John Smith
  - Doctor name = Dr. David Broussesau
  - Date = September 17, 2002
  - time = 9:30 A.M.

*Figure 2-1: Classes and Objects*
Methods and Messages

- Methods implement an object’s behavior
  - Analogous to a function or procedure
- Messages are sent to trigger methods
  - Procedure call from one object to the next
FIGURE 2-2
Messages and Methods

A message is sent to the application. The object’s insert method will respond to the message and insert a new patient instance.
Encapsulation and Information Hiding

- Encapsulation
  - combination of data and process into an entity

- Information Hiding
  - Only the information required to use a software module is published to the user

- Reusability Key
  - Use an object by calling methods
Inheritance

- Superclasses or general classes are at the top of a hierarchy of classes
- Subclasses or specific classes are at the bottom
- Subclasses inherit attributes and methods from classes higher in the hierarchy
Class Hierarchy

FIGURE 2-3
Class Hierarchy
Inheritance

**FIGURE 2-4**
Inheritance
Polymorphism and Dynamic Binding

- Polymorphism
  - A message can be interpreted differently by different classes of objects

- Dynamic Binding
  - Sometimes called late binding
  - Delays typing or choosing a method for an object until run-time

- Static Binding
  - Type of object determined at compile time
Polymorphism & Encapsulation

1. An insert message is sent to the patient object.
2. The object’s method responds to the message.
3. The application responds appropriately.

1. An insert message is sent to the appointment object.
2. The object’s method responds to the message.
3. The application responds appropriately.

**Figure 2-5** Polymorphism and Encapsulation
The Unified Modeling Language, Version 2.0

- Structure Diagrams
- Behavior Diagrams
- Extension Mechanisms
- Developers
  - Grady Booch
  - Ivar Jacobson
  - James Rumbaugh
Structure Diagrams include:

- Class
- Object
- package
- Deployment
- Component
- Composite structure diagrams
<table>
<thead>
<tr>
<th>Diagram Name</th>
<th>Used to</th>
<th>Primary Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structure Diagrams</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Illustrate the relationships between classes modeled in the system.</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td>Object</td>
<td>Illustrate the relationships between objects modeled in the system.</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>Used when actual instances of the classes will better communicate the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>model.</td>
<td></td>
</tr>
<tr>
<td>Package</td>
<td>Group other UML elements together to form higher level constructs.</td>
<td>Analysis, Design,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>Deployment</td>
<td>Show the physical architecture of the system. Can also be used to show</td>
<td>Physical Design,</td>
</tr>
<tr>
<td></td>
<td>software components being deployed onto the physical architecture.</td>
<td>Implementation</td>
</tr>
<tr>
<td>Component</td>
<td>illustrate the physical relationships among the software components.</td>
<td>Physical Design,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>Composite Structure</td>
<td>Illustrate the internal structure of a class, i.e., the relationships</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>among the parts of a class.</td>
<td></td>
</tr>
<tr>
<td><strong>Behavioral Diagrams</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Activity</td>
<td>Illustrate business workflows independent of classes, the flow of</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>activities in a use case, or detailed design of a method.</td>
<td></td>
</tr>
<tr>
<td>Sequence</td>
<td>Model the behavior of objects within a use case. Focuses on the time-</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>based ordering of an activity.</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>Model the behavior of objects within a use case. Focuses on the</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>communication among a set of collaborating objects of an activity.</td>
<td></td>
</tr>
<tr>
<td>Interaction Overview</td>
<td>Illustrate an overview of the flow of control of a process.</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td>Timing</td>
<td>Illustrate the interaction that takes place among a set of objects and</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td></td>
<td>the state changes in which they go through along a time axis.</td>
<td></td>
</tr>
<tr>
<td>Behavioral State Machine</td>
<td>Examine the behavior of one class.</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td>Protocol State Machine</td>
<td>Illustrates the dependencies among the different interfaces of a class.</td>
<td>Analysis, Design</td>
</tr>
<tr>
<td>Use-Case</td>
<td>Capture business requirements for the system and to illustrate the</td>
<td>Analysis</td>
</tr>
<tr>
<td></td>
<td>interaction between the system and its environment.</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 2-6** UML 2.0 Diagram Summary
Structure Diagrams

- Class
  - relationship between classes
- Object
  - Relationships between objects
- Package
  - Group UML elements together to form higher level constructs
Structure Diagrams Cont.

- **Deployment**
  - Shows the physical architecture and software components of system

- **Component**
  - Physical relationships among software components

- **Composite Structure**
  - Illustrates internal structure of a class
Activity Diagrams

- **Activity**
  - Illustrates business workflows
- **Sequence**
  - Time-based ordering Behavior of objects activities in a use case
- **Communication**
  - Communication among a set of collaborating objects of an activity
- **Interaction Overview Timing**
  - Overview of flow of control of a process
State Machines

- Behavioral State Machine
  - Examines behavior of one class
- Protocol State Machine
  - Shows dependencies of different interfaces of a class
- Use-Case
  - Captures business requirements
  - Illustrates interaction between system and environment
Use Case Diagrams

- Captures Business requirements
- Illustrates interaction between a system and its environment
  - Includes end user
  - Any external system that interacts with its information system
- Documents and clarifies requirements of system being modeled
Extension Mechanisms

- Stereotypes
  - Gives ability to incrementally extend UML
- Tagged Values
  - Add new properties to base elements
- Constraints
  - Place restrictions on use of model elements
- Profiles
  - Group model elements into a package
Object Oriented Systems Analysis and Design

- Use-case driven
- Architecture Centric
- Iterative and Incremental
- The Unified Process
# Engineering Workflows

<table>
<thead>
<tr>
<th>Phases</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Modeling</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analysis</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Supporting Workflows

<table>
<thead>
<tr>
<th>Phases</th>
<th>Inception</th>
<th>Elaboration</th>
<th>Construction</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration and Change Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Environment</td>
<td>Iter 1</td>
<td>...</td>
<td>Iter i + 1</td>
<td>Iter j + 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>Iter k</td>
<td>...</td>
<td>Iter k + 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>Iter m</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Minimalist Approach

- Benefits of Object-Oriented Systems Analysis and Design
- Extensions of the Unified Process
- The Minimalist Object-Oriented Systems Analysis and Design Approach
## Benefits of the Object Approach

<table>
<thead>
<tr>
<th>Concept</th>
<th>Supports</th>
<th>Leads to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class, objects, methods, and messages</td>
<td>A more realistic way for people to think about their business</td>
<td>Better communication between user and analyst developer</td>
</tr>
<tr>
<td></td>
<td>Highly cohesive units that contain both data and processes</td>
<td>Reusable objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Benefits from having a highly cohesive system (see cohesion in Chapter 13)</td>
</tr>
<tr>
<td>Encapsulation and information hiding</td>
<td>Loosely coupled units</td>
<td>Reusable objects</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fewer ripple effects from changes within an object or in the system itself</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Benefits from having a loosely coupled system design (see coupling in Chapter 13)</td>
</tr>
<tr>
<td>Inheritance</td>
<td>Allows us to use classes as standard templates from which other classes can be built</td>
<td>Less redundancy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Faster creation of new classes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standards and consistency within and across development efforts</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ease in supporting exceptions</td>
</tr>
<tr>
<td>Polymorphism and Dynamic Binding</td>
<td>Minimal messaging that is interpreted by objects themselves</td>
<td>Simpler programming of events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ease in replacing or changing objects in a system</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fewer ripple effects from changes within an object or in the system itself</td>
</tr>
<tr>
<td>Use-case driven and use cases</td>
<td>Allows users and analysts to focus on how a user will interact with the system to perform a single activity</td>
<td>Better understanding and gathering of user needs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Better communication between user and analyst</td>
</tr>
<tr>
<td>Architecture centric and functional, static, and dynamic views</td>
<td>Viewing the evolving system from multiple points of view</td>
<td>Better understanding and modeling of user needs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>More complete depiction of information system</td>
</tr>
<tr>
<td>Iterative and incremental development</td>
<td>Continuous testing and refinement of the evolving system</td>
<td>Meeting real needs of users</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Higher quality systems</td>
</tr>
</tbody>
</table>

**FIGURE 2-8** Benefits of the Object Approach
MOOSAD Approach

Figure 2-10: The Minimalist Object-Oriented Systems Analysis and Design (MOOSAD) Approach
Basic Characteristics of Object Oriented Systems

- Identifying business value
- Analyze feasibility
- Develop workplan
- Staff the project
- Control and direct project
- Requirements determination
- Functional modeling
- Structural modeling
- Behavioral modeling
- Moving on to design
UML Summary

- Class and method design
- Data management layer design
- Human computer interaction layer design
- Physical architecture layer design
- Construction
- Installation
- Operations and support
Summary

- Basic characteristics of an object oriented system
- Unified modeling system
- Object oriented Systems Analysis and Design
- Minimalist approach to Object oriented systems analysis and design with UML
EOC Question Chapter 2

1. Describe the major elements and issues with an object-oriented approach to developing information systems.
2. What is the difference between classes and objects?
3. What are methods and messages?
4. Why are encapsulation and information hiding important characteristics of object-oriented systems?
5. What is meant by polymorphism when applied to object-oriented systems?
6. Compare and contrast dynamic and static binding.
7. What is the Unified Modeling Language?
8. Who is the Object Management Group?
9. What is the primary purpose of structure diagrams?
10. For what are behavior diagrams used?
11. What is a use case?
12. What is meant by use-case driven?
EOC Question Chapter 2

13. Why is it important for an OOSAD approach to be architecture centric?
14. What does it mean for an OOSAD approach to be incremental and iterative?
15. What are the phases and workflows of the Unified Process?
16. Compare the phases of Unified Process with phases of waterfall model described in Chapter 1.
17. What are the benefits of an object-oriented approach to systems analysis and design?
18. Compare and contrast the typical phased delivery RAD SDLC with the modified-phased delivery RAD SDLC associated with the OOSAD approach SDLC associated with the OOSAD approach.
19. What are the steps or phases of the minimalist OOSAD approach described in this chapter?
20. What are the different views supported by the minimalist OOSAD approach described in this chapter?
21. What diagrams and models support the different views identified in the previous question?
22. What is a build?
23. What is a pattern?
24. How do the Unified Process’s phases and workflows map into the steps of the minimalist OOSAD approach described in this chapter?