Introduction

Object Orientated Analysis and Design

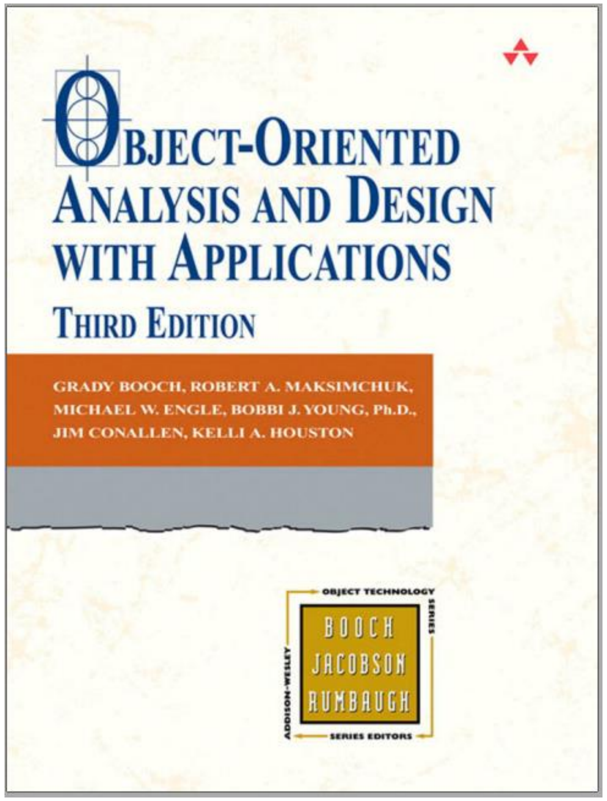
Benjamin Kenwright

Outline

- What do we mean by Object Orientated?
- Why do we need to analyze and design our solutions?
  - Give examples
- What analysis and design tools are available?
- Course structure
- Grading/assessment

Recommended Book

- Object Oriented Analysis and Design with Applications 3rd Edition by Booch
  - Ebook Available: https://zjnu2017.github.io/OOAD
  - Complete Reading Chapter 1 Before Next Week


OBJECT-ORIENTED ANALYSIS AND DESIGN WITH APPLICATIONS
THIRD EDITION
GRADY BOOCH, ROBERT A. MAKSIMCHUK, MICHAEL W. ENGLE, BOBBI J. YOUNG, Ph.D., JIM CONALLEN, KELLI A. HOUSTON
OBJECT TECHNOLOGY SERIES
BOOCH JACOBSON RUMBAUGH
SERIES EDITORS
ADDISON-WESLEY

Recommended

- Also read around the subject to gain a broad/comprehensive understanding of the topic
  - Articles, books, online-tutorials, …

Why use Object-Orientation Concepts?

Why use Object-Orientation Concepts?

- Sooner rather than later, you'll have to work with object-oriented code
- Modularity
- Scalability
- Frameworks
- Contributing to open source software
- Gives you various ways to think and solve problems
- Easier to translate your programming skills into other Object-Oriented languages
- Become a more valuable developer

Revision

- Review Software Development Methodologies
- (Core OO Programming Principles)

- What is an object?
- What is a class?
- What is abstraction?
- What is encapsulation?
- What is inheritance?
- What is an Interface?
- What is polymorphism?

Question

- What is a Great Software Solution?

Answer

- A great software must satisfy the customer
- The software must do what the customer wants it to do!
- Great software is also
  - well-designed
  - well-coded
  - easy to maintain, reuse, and extend

Question

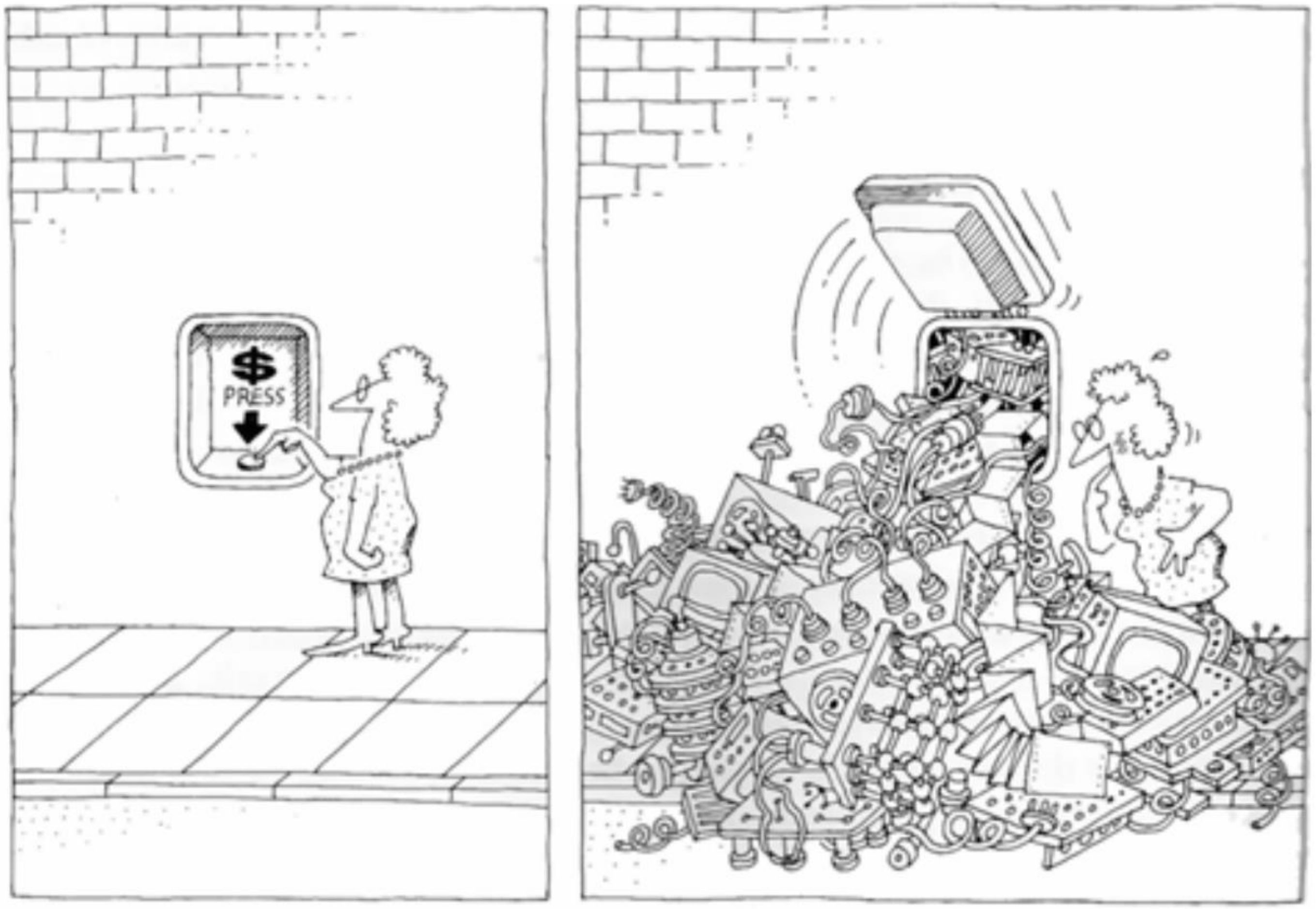
- How do we make Great Software Solutions?

Answer

- Apply Object Orientated Analysis and Design Processes
  - e.g., Make sure your software does what the customer wants it to do
  - Use OO principles to add flexibility
  - Strive for a maintainable, reusable design
  - …

Why do we need Object Orientated Analysis and Design (OOAD)?

Why we need OOAD?



Why we need OOAD?

- Software is inherently complex
  - Complexity of software systems often exceeds the human intellectual capacity
- Helps create illusion of simplicity
- Order to Chaos
  - Add meaningful logic

Object Orientated Analysis and Design

- Techniques that allow us to decompose problems/tasks into manageable components
- Employs object-oriented methods for organising the complexity of the system
- Provides a rich set of models to understand the different aspects of the system under consideration

The primary tasks for the object-oriented `analysis’ (OOA)

- Find the objects
- Organize the objects
- Describe how the objects interact
- Define the behavior of the objects
- Define the internals of the objects

Common models used in OOA are use-cases and object models

Class Participation

- Welcome participation by students
- Feel free to interrupt me during lectures to ask questions!
- Stupid Questions — No such thing!
- No participation leads to “silent tomb” - Boring!
- If I speak too fast or you are unsure of something, stop me and ask/tell me to slow down

Class Participation

- Quizzes
- Discussion/Project
- Homeworks
  - Reading Chapters
  - Researching Topics
  - Case studies

Question

- Which of the following is the functionality of ‘Data Abstraction’?

a) Reduce Complexity

b) Binds together code and data

c) Parallelism

d) None of the mentioned

Answer

- Answer: a

Explanation: An essential element of Object Oriented Programming is ‘Data Abstraction’ which means hiding things. Complexity is managed through abstraction.

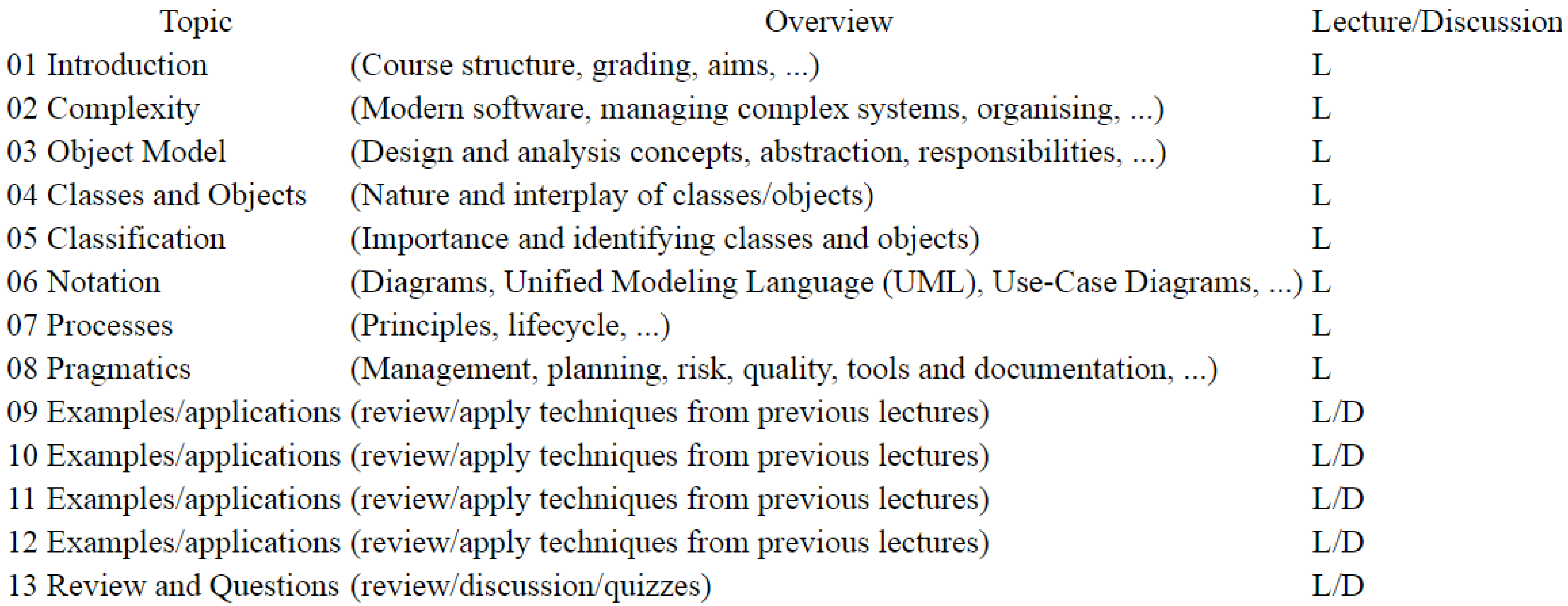
Goals

- Provide students with knowledge and skills in:
  - Object-oriented concepts
  - OO analysis, design, and implementation techniques
  - Object-oriented design methods
    - (aka software development life cycles)
- Students should view OO software development as a software engineering <u>process</u> that has well-defined stages with each stage requiring specific <u>tools</u> and techniques

Grading

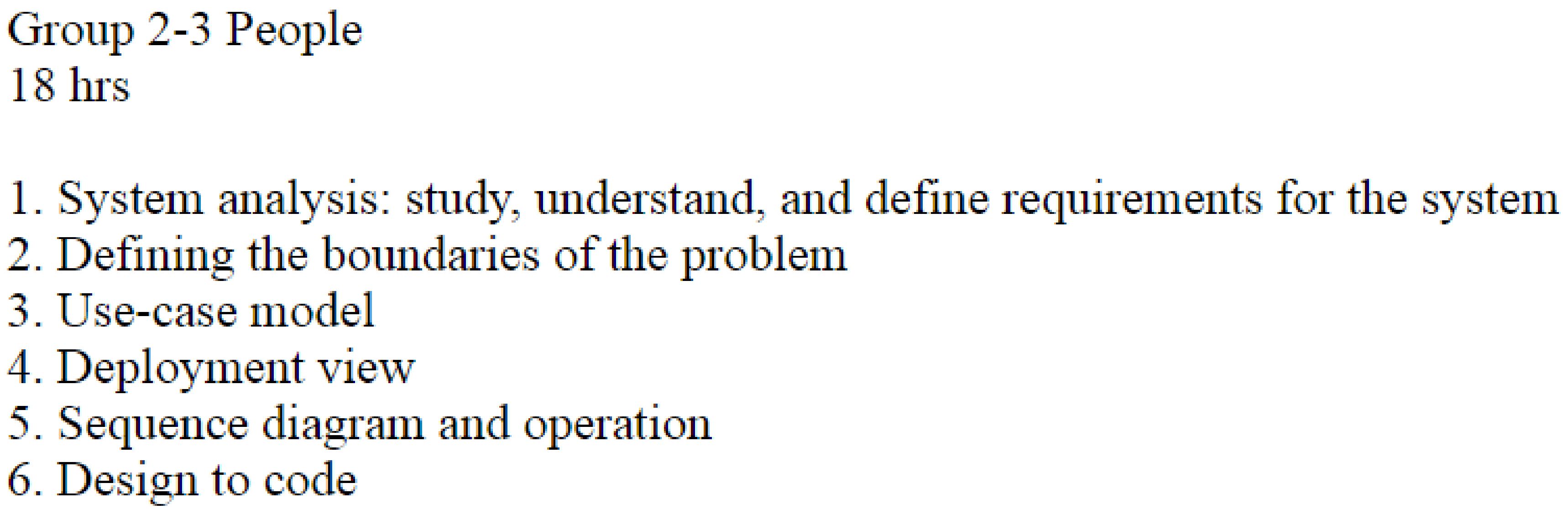
- Attendance 10%
- Experiments & Discussion 40%
- Final Exam 50%

Structure Topics



Topic	Overview	Lecture/Discussion
01 Introduction	(Course structure, grading, aims, ...)	L
02 Complexity	(Modern software, managing complex systems, organising, ...)	L
03 Object Model	(Design and analysis concepts, abstraction, responsibilities, ...)	L
04 Classes and Objects	(Nature and interplay of classes/objects)	L
05 Classification	(Importance and identifying classes and objects)	L
06 Notation	(Diagrams, Unified Modeling Language (UML), Use-Case Diagrams, ...)	L
07 Processes	(Principles, lifecycle, ...)	L
08 Pragmatics	(Management, planning, risk, quality, tools and documentation, ...)	L
09 Examples/applications	(review/apply techniques from previous lectures)	L/D
10 Examples/applications	(review/apply techniques from previous lectures)	L/D
11 Examples/applications	(review/apply techniques from previous lectures)	L/D
12 Examples/applications	(review/apply techniques from previous lectures)	L/D
13 Review and Questions	(review/discussion/quizzes)	L/D

Experiments/Discussion



Group 2-3 People
18 hrs

1. System analysis: study, understand, and define requirements for the system
2. Defining the boundaries of the problem
3. Use-case model
4. Deployment view
5. Sequence diagram and operation
6. Design to code

Contact Details

- Questions/Issues

  Benjamin Kenwright
  email: bkenwright@ieee.org

- Open Door Policy
  - Problems/Help
  - Within Reason

Question

- Which of the following mechanisms is/are provided by Object Oriented Language to implement Object Oriented Model?

a) Encapsulation

b) Inheritance

c) Polymorphism

d) All of the mentioned

Answer

- d) All of the mentioned

Question

- An object is a combination of data and logic; the representation of some real-world entity

a) True

b) False

Answer

- a) True

Summary

- Clear idea of the goal of this course/topic
- Structure of the course
- Assessment/grading

This Week

- Review Slides
- Read Chapter 1
- Challenging so Start Early

Questions/Discussion

- Optional Revision/Review Slides

Revision

- Review Software Development Methodologies
- (Core OO Programming Principles)

- Why use object-orientation?
- What is a class?
- What is an object?
- What is abstraction?
- What is encapsulation?
- What is inheritance?
- What is an Interface?
- What is polymorphism?

What Is a Class?

- A class is a blueprint or prototype from which objects are created. This section defines a class that models the state and behavior of a real-world object. It intentionally focuses on the basics, showing how even a simple class can cleanly model state and behavior.

What Is an Object?

- An object is a software bundle of related state and behavior. Software objects are often used to model the real-world objects that you find in everyday life.

What is Abstraction?

- Abstraction is a process where you show only “relevant” data and “hide” unnecessary details of an object from the user. Consider your mobile phone, you just need to know what buttons are to be pressed to send a message or make a call, What happens when you press a button, how your messages are sent, how your calls are connected is all abstracted away from the user.

What is Encapsulation?

- Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.

What Is Inheritance?

- Inheritance provides a powerful and natural mechanism for organizing and structuring your software. Classes inherit state and behavior from their superclasses.

What Is an Interface?

- An interface is a contract between a class and the outside world. When a class implements an interface, it promises to provide the behavior published by that interface.

What does Polymorphism mean?

- Polymorphism is an object-oriented programming concept that refers to the ability of a variable, function or object to take on multiple forms. A language that features polymorphism allows developers to program in the general rather than program in the specific.