CMSC 128

Introduction to Software Engineering
Second Semester AY 2007-2008

jachermocilla@uplb.edu.ph

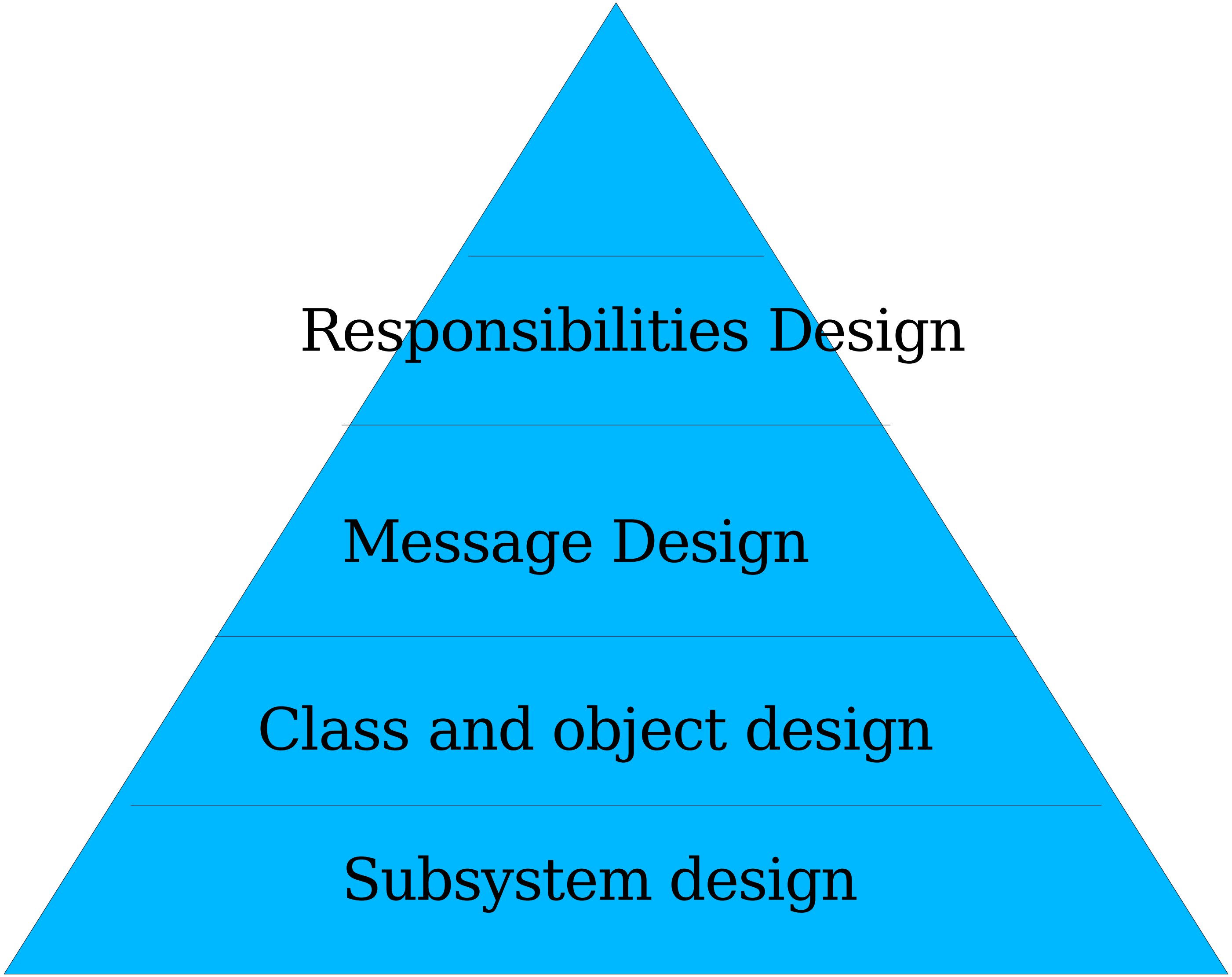
Object-Oriented Design

- OOD transforms the analysis model created using OOA into a design model that serves as a blueprint for software construction
- The unique feature of OOD is in its ability to build upon four important software design concepts: abstraction, information hiding, functional independence, and modularity

Complexity in OOD

- Designing OO software is hard, designing reusable software is even harder
  - Find pertinent objects
  - Factor them into classes at right granularity
  - Define class interfaces and inheritance relationships
  - Design should be problem specific but must be general enough to address future problems and requirements
  - Avoid redesign

OOD Pyramid



Responsibilities Design

Message Design

Class and object design

Subsystem design

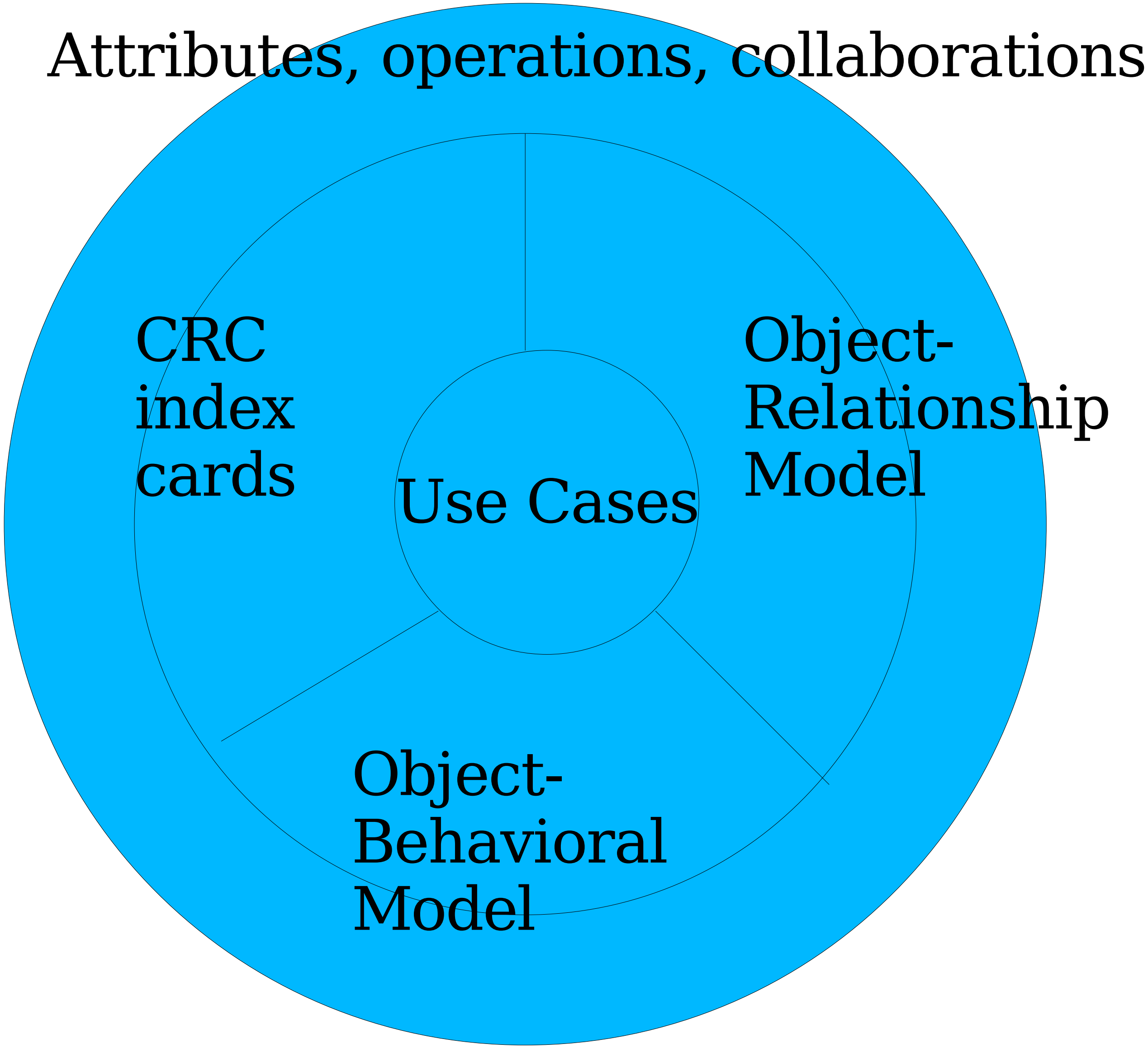
OOD Pyramid

- Subsystem Layer
  - Representations of the subsystems that enable the software to achieve customer requirements and implement technical infrastructure
- Class and Object design
  - Class hierarchies and design representations of objects

OOD Pyramid

- Message Layer
  - Details that enable each object to communicate with its collaborators, both internal and external interfaces
- Responsibilities Layer
  - Contains the data structures and algorithmic design for all attributes and operations for each object

OOA Model



Attributes, operations, collaborations

CRC index cards

Object-Relationship Model

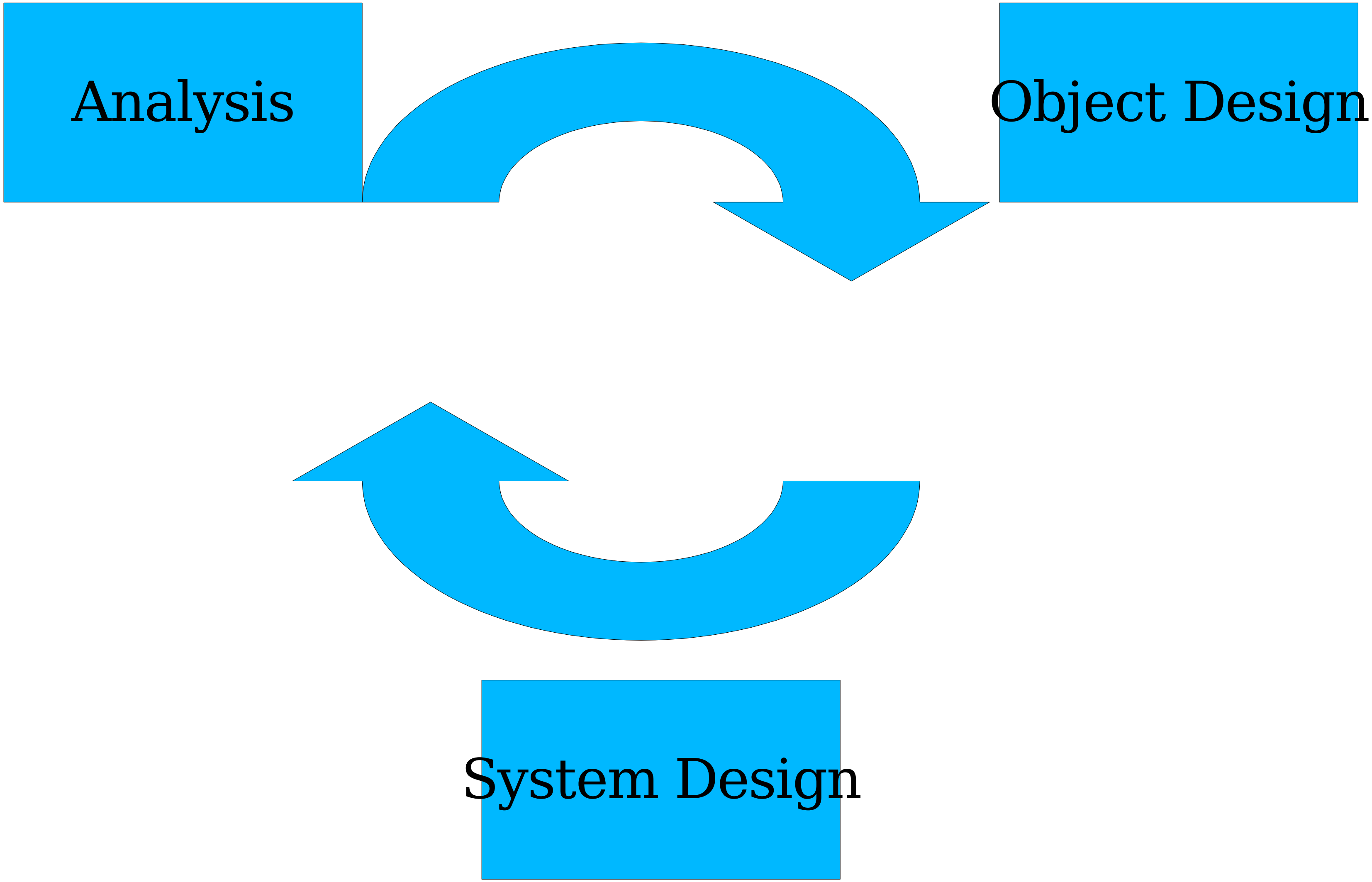
Use Cases

Object-Behavioral Model

Generic Components of OOD

- Difficult to differentiate OOA from OOD
- In essence OOA is a classification activity
    - Determine the classes of objects as solution is developed
    - Determines object relationships and behavior
- OOD
    - Indicate the objects derived from the classes and how these objects interrelate with one another

Process Flow for OOD



Generic Components of OOD

- After a sufficient OOA model is developed, concentrate on the design
- Start by describing the characteristics of the subsystem required to implement both customer requirements and the support environment necessary
  - Subsystem Design

Generic Components of OOD

- Basic subsystem design components
  - Problem domain
    - For directly implementing customer requirements
  - Human interaction
    - UI, including reusable UI components
  - Task management
    - Controlling and coordinating concurrent tasks
  - Data management
    - Storage and retrieval of objects

Generic Components of OOD

- After subsystem design, object design comes next
- Elements of the CRC model are translated into design realization

Object Design

- OOA Model
  - Classes
  - Attributes
  - Methods
  - Relationships
  - Behavior

- OOD Model
  - Objects
  - Data structures
  - Algorithms
  - Messaging
  - Control

System Design Steps

- Partition OOA model into subsystems
- Identify concurrency dictated by the problem
- Allocate the subsystems to processors and tasks
- Choose a basic strategy for implementing data management
- Identify global resources and control mechanisms required to access them

System Design Steps

- Design an appropriate control mechanism for the system
- Consider how boundary conditions should be handled
- Review and consider trade-offs

Partition Analysis Model

- Create subsytems which define a cohesive collection of classes, relationships, and behavior
- All elements of a subsystem share a common property
  - Accomplish same function
  - Reside within the same hardware
  - Manage same class of resources

Partition Analysis Model

- Subsystems are characterized by their responsibilities – services it provides
- A service is a collection of operations that performs a specific function
- Guidelines for designing subsystems
  - A subsystem should have a well defined interface
  - Classes within subsystem should communicate only with classes within the subsystem

Partition Analysis Model

- Guidelines for designing subsystems
  - Number of subsystems should be kept small
  - Subsystems can be partitioned internally to reduce complexity
- Types of communication between subsystems
  - client/server
    - Distinct client or server role
  - peer-to-peer
    - No distinction between client or server

Concurrency and Tasks

- Dynamic aspect of object-behavioral analysis model provide an indication of concurrency among objects
- Options for allocating concurrent subsystems
  - Allocate each subsystem to an independent processor
  - Provide concurrency support through operating system features
- Choice depends on performance, costs, etc

Concurrency and Tasks

- Concurrent tasks are defined by examining the STD for each object
  - A thread of control is present when an object is in active state
  - It is possible to have multiple threads of control when several objects are in the active state
- Tasks can be event driven or clock driven
- Think of tasks as interrupt or event handlers

Data Management

- Concerns
  1. Managing data critical to the application itself
  2. Creation of an infrastructure for storage and retrieval of objects
- Layered to isolate low-level requirements
- A DBMS is used normally
  - Objects to manipulate database are members of reusable classes from domain analysis
    - ex. JDBC, Hibernate, Torque

Data Management

- Includes design of attributes and operations required to manage objects
  - ex. serial id in java, annotations in Hibernate

Resource Management

- Define resource manager objects for handling and controlling system resources
- Resources are usually external entities
  - Disk drive, processor, communication line
- Define a “guardian” object
  - ex. Toolkit Object in Java

HCI Component

- Inputs come from use case scenarios
- A command hierarchy is identified
  - Menu categories
  - Refined iteratively
- A variety of HCI development environment exists - reusable
  - AWT, Swing, SWT, MS Windows, Qt, GTK

Intersubsystem Communication

- Extension of object-object communication
  - Use of messages
- Specify a contract that exists between two subsystems
  - Contract provides an indication of the ways in which one subsystem can interact with another

Intersubsystem Communication

- Steps

1. List requests that can be made by the collaborators and define them as contracts
2. For each contract note the operations required to implement the contract. Associate the operation with a specific class that reside within a subsystem
3. Create a table that describes each contract
4. Subsystem collaboration diagram can be created

Object Design Process

- Develop a detailed design of the attributes and operations that comprise each class, and a thorough specification of the messages that connect the class with their collaborators
- Design description of an object
  - Protocol Description
  - Implementation Description

Object Description

- Protocol Description (What)
  - Establishes the interface of an object by defining each message that the object can receive and the related operation that the object performs when it receives the message
  - ex. Given a “Hard Disk” object, possible messages:
    - Format
    - Reset
    - Seek
    - Read

Object Description

- Implementation Description (How)
  - Provides the internal (“hidden”) details required for implementation but not necessarily for invocation
    - Object name and reference to class (rose object instance of Flower class)
    - Private data structures: data items and types
    - Procedural description of each operation

Algorithms/Data Structures

- An algorithm is created to implement the specification of each operation
- Data structures are designed concurrently with algorithms
  - Operations manipulate attributes
- Types of operations
  - Operations that manipulate data
  - Operations that perform computation
  - Operation that monitors an object

OOD Optimization

- Review object-relationship model to ensure implemented design leads to efficient utilization of resources and ease of implementation
- Revise attribute data structures and corresponding operation algorithms to enhance efficient processing
- Create new attributes to save derived information, avoiding recomputation

Components and Interfaces

- Interfaces should be represented in the context of the programming language for use in the implementation
- Makes use of a Program Design Language (PDL)
  - ex. interface in Java

Design Patterns

- The best designers in a fields have an uncanny ability to see patterns that characterize a problem and corresponding patterns that can be combined to create a solution
- Examples: MVC, Singleton, Iterator, Visitor, Factory, Abstract Factory, Bridge, etc
- Read “Gang of Four” book by Gamma et al

Summary

- Object-oriented design can be described using four layers: subsystem design,class and object design, message design, responsibilities design
- OOD looks at two levels of abstractions: subsystem design and class/object design
- OOD provides us with the means for breaking down the “partitions” between data and process

Reference

- Roger S. Pressman.Software Engineering: A Practitioner's Approach, 4th Ed.McGraw-Hill,1997. Chapter 21