CMSC 128

Introduction to Software Engineering
Second Semester AY 2007-2008

jachermocilla@uplb.edu.ph

Object Oriented Concepts

- We live in a world of objects
- Why not use objects as abstraction for software?
- OO began in 1960's, wide adoption only after 20 years
- OO is slowly replacing classical software development approaches
  - why?

Advantages of OO

- Reuse of software *components*
  - Leads to faster development and higher quality programs
- Easier to maintain because its structure is highly decoupled - less dependencies
  - Fewer side effects, less frustration
- OO systems are easier to adapt and easier to scale

OO Paradigm

- Term 'object-oriented' was used to denote development of software using an OOPL
- Best paradigm is evolutionary and component assembly
  - Classes are looked up in a library
  - If classes not found, following steps are done
    - Object-oriented analysis (OOA)
    - Object-oriented design (OOD)
    - Object-oriented programming (OOP)
    - Object-oriented testing (OOT)

OO Concepts

- Start with a real world object: Rose
  - Rose is an instance of a class Flower
  - A set of generic attributes can be associated with every object in the class Flower
    - name,color, scent, shape, length
    - cost
  - Since Rose is a member of Flower, it has all the attributes of flower
  - Once the class has been defined, new instances/objects can be created, each having the attributes of the original class

OO Concepts

- Start with a real world object: Rose
  - Operations/actions can be performed on every object in the class – modify attributes
    - water, display, arrange, cut
    - trash
  - Each object is said to encapsulate both attributes and operations
    - Packaged under one name, can be reused as one *specification* or program component

Classes and Objects

- A good design should exhibit data and procedural abstractions that lead to effective modularity
- A class encapsulates the *data and procedural abstractions* that are required to describe the *content and behavior* of some real world entity
- A class is a generalized description (template, pattern, blueprint) that describes a collection of similar objects

Classes and Objects

- A new class can be defined from an existing class through inheritance
- A class hierarchy shows the relationships between classes

Attributes

- A binary relation between a class and a certain domain
  - An attribute can take on a value defined by an enumerated domain
  - Domain is a set of specific values
    - color : {white, red, blue, pink}
    - May be a set of classes

Operations

- Methods, Services
- Algorithms that process data
- Modules in the conventional sense
- Each operation provides a representation of one of the behaviors of the object
  - GetColor() - returns the color of flower
  - Implies that object can respond to stimulus (message)

Messages

- Means by which object interacts
- Stimulates some behavior on the receiving object
  - Behavior is accomplished when an operation is executed
- Format
  - message: {destination, operation, parameters}

Encapsulation

- Data and operation bundled together
- Internal implementation details of data and procedures are hidden from the outside world (information hiding)
- Facilitates reuse
- Interfaces (set of operations) among encapsulated objects are simplified
  - Reduced coupling

Inheritance

- A subclass Y inherits all of the attributes and operations of superclass X
  - Reuse of data and operations of X in Y
- Change in superclass is propagated to subclasses
  - Class hierarchy becomes a mechanism for *change propagation*
- At each level of the class hierarchy, new attributes and operations may be added
- Multiple inheritance

Adding a new class

- Options
  - Design and build from scratch
  - Search the hierarchy for a suitable ancestor class to inherit from
  - Restructure/refactor class hierarchy
  - Characteristics of an existing class can be overridden (modified in a subclass)

Polymorphism

- Literally: “many forms”
- Reduces the effort required to extend an existing OO system
  - enables plug-in mechanism
- Enables a number of different operations to have the same name: <u>overloading</u>

Object Model

- Identifying classes and objects
  - Perform grammatical parse of processing narrative (scope document) for the system
  - Objects are the nouns or noun clauses
    - Categorized as object in *solution space* or *problem space*
  - Objects can be classified as *external entities, things, occurrences/events, roles, organizational units, places, structures*

Object Model

- Six selection characteristics for inclusion of a potential object to the analysis model
  1. Retained information
     - Must be remembered for system to function
  2. Needed services
     - Must have a set of identifiable operations
  3. Multiple attributes
     - Must have many attributes
  4. Common attributes
     - Can be defined/applied to all instances/objects

Object Model

- Six selection characteristics for inclusion of a potential object to the analysis model

5. Common operations
    - Can be defined/applied to all instances/objects
6. Essential requirements
    - External entities that produce/consume information needed for the system to function

Object Model

- Specifying attributes
  - Attributes describe an object that has been selected for inclusion in the analysis model
  - In essence, the attributes define the object
  - Depends on the problem
  - 'What data items fully define this object in the context of the problem at hand?'

Object Model

- Defining operations
  - Operations define the behavior of an object
  - Changes one or more attribute values
  - Categories of operations
    - Operations that manipulate data
    - Operations that perform computation
    - Operations that monitor an object for the occurrence of a controlling event
  - Identify verbs in the grammatical parse

Managing OO Projects

- Common Process Framework
  - Software engineering paradigm, tasks, milestones, deliverables
  - Iterative development, not linear
  - Recursive/Parallel model
    - Decompose into independent components
    - Reapply decomposition to independent components
    - Conduct reapplication of decomposition in parallel
    - Continue as desired (if in class repo, stop)

OO Project Metrics

- Lorenz and Kidd(1994) Metrics
  - Number of scenario scripts
    - (initiator, action, participant)
  - Number of key classes
    - Problem domain specific
  - Number of support classes
    - GUI, Data Access Objects
  - Average number of support classes per key class
  - Number of subsystems (group of related classes)

OO Estimating

- Conventional methods apply
- Lorenz and Kidd (1994)
  1. Use conventional methods (Effort, FP)
  2. Using OOA, develop scenario scripts and determine a count
  3. Using OOA, determine number of key classes
  4. Categorize type of interface and develop multiplier for support classes. Multiply the number of key classes with the multiplier
  5. Multiply number of classes with average effort/class

OO Scheduling

- Complicated by the iterative nature
- Metrics that may be used
  - Number of major iterations
  - Number of completed contracts
    - A contract is a 'group of related public responsibilities that are provided by subsystems and classes to their clients'

Tracking

- Major milestones
  - OOA completed
    - Classes and class hierarchy defined
  - OOD completed
    - Subsystems defined and reviewed
  - OOP completed
    - Code has been written from design model
  - OOT completed
    - Class level test has been conducted

Summary

- Object technologies reflect a natural view of the world
- Objects are grouped into classes and hierarchies are created
- A class contains a set of attributes that describe it and operations that define its behavior. New objects can be created from once a class has been defined
- Encapsulation, Inheritance, Polymorphism differentiates OO from conventional methods

Reference

- Roger S. Pressman.Software Engineering: A Practitioner's Approach, 4th Ed.McGraw-Hill,1997. Chapter 19