LECTURE-13

- Object Oriented Design
- Covered concepts
  - Classes and objects
  - Encapsulation
  - State, behavior, identity
  - Relationships among objects
  - Inheritance and polymorphism
- Covered constraints
  - Coupling
  - Cohesion
- Covered tools
  - Class diagrams
  - Sequence diagrams

Metrics

- Weighted Methods per Class (WMC)
  - Complexity of a class depends on number of classes and their complexity
    - Suppose class C has methods M_1, M_2, …, M_n
    - Suppose complexity of methods is c_1, c_2… determined by some functional complexity metric
    - WMC = Σc_i
    - If the complexity of each method is considered 1, WMC gives the total number of methods in the class
  - Large WMC might mean that the class is more fault-prone

Metrics…

- The deeper a class is in a class hierarchy
  - More methods to reuse – larger reuse potential
  - Increased coupling – harder to make change
- Depth of Inheritance (DIT) Tree
  - DIT of class C in an inheritance hierarchy tree is depth from the root class
    - Shortest path from root to node C
  - DIT is significant in predicting fault proneness

Metrics…

- Number of Children (NOC)
  - Number of immediate subclasses of C
  - Evaluates the degree of reuse
  - Higher NOC indicates reuse of definitions in superclass by a larger number of subclasses
  - Indicates influence of a class on other elements
    - Larger influence, more important to get design correct
  - Higher NOC classes are less defect-prone

NOC is only measuring structure, not inheritance

Metrics…

- Coupling Between Classes (CBC)
  - Reduces modularity and makes module modification harder
  - CBC = Number of classes to which this class is coupled
    - Two classes are coupled if methods of one use methods or instance variables of other
  - Can be determined from code
    - There are indirect forms of coupling that cannot be statically determined (e.g., pointers)
  - Can predict fault proneness of classes, particular user interface classes

Metrics…

- Response for a Class (RFC)
  - The total number of methods that can be invoked from an object of this class
  - RFC of C is cardinality of the response set for a class
    - Set of all methods that can be invoked if a message is sent to an object of this class
      - All methods of C as well as other classes the methods of C send messages
    - Even if CBC of a class is 1, RBC may be high
  - Captures the strength of connections
  - Harder to test classes with high RFC

Metrics…

- Lack of Cohesion in Methods (LCOM)
  - Cohesion captures how close are different methods of a class bound
    - Two methods form a cohesive pair if they access some common variables
    - Form a non-cohesive pair if no common variables
    - High cohesion is highly desirable
  - LCOM is the number of method pairs that are non-cohesive minus the number of cohesive pairs
  - Not significant in predicting fault tolerance of a class

Metrics Studies Show

- Weighted Methods per Class (WMC)
  - Classes tend to have only small number of methods
    - Classes are simple and provide some specific abstraction and operations
    - Only few classes have many methods defined in them
  - Has a reasonable correlation with fault-proneness of a class
- Depth of Inheritance (DIT)
  - Classes tend to be close to the root
    - Max DIT around 10
    - Most classes have DIT of 0 (they are the root)
  - Designers tend to keep the number of abstraction levels small, i.e., they give up reusability in favor of comprehensibility
- Number of Children (NOC)
  - Classes generally had a smaller NOC value with most having 0
  - Inheritance was not used very heavily

Metrics Studies Show…

- Coupling Between Classes (CBC)
  - Most classes are self contained with CBC = 0
    - Not coupled with any other class
  - Interface objects tend to have higher CBC
- Response for a Class (RFC)
  - Most classes tend to invoke a small number of methods of other classes
  - Classes for interface objects tend to have higher RFC
- Lack of Cohesion in Methods (LCOM)
  - Not very good at predicting fault-proneness

Summary

- OO is a newer paradigm, slowly replacing the functional approach
- OO models both data and functions
- UML is a notation that is used often to model systems in an OO manner
- UML provides various diagrams for modeling the structure, dynamic behavior, etc.
- Through UML modeling, design for the system can be developed
- Metrics can help predict fault proneness of design

Example OO Design – PIMS
Personal Investment System

- Help investors keep track of their investments
- Determine rate of return
  - On individual investments
  - On overall portfolio
- Determine net worth of portfolios