OOSE 課輔

2019/09/11

加分

- 個人:
  - 自願回答問題 +10
- 群體:
  - Bonus Time 全對+50，半對+30
- 登記方式:
  - 手寫 班級 姓名 學號 給助教

MVC

- What is “MVC” for?
- Model, View and Controller
- What is the most important one in MVC?
- Model, so view and controller can be combined together

The Model

- The Model is the part that does the work--it *models* the actual problem being solved
- **The Model should be independent of both the Controller and the View**
  - **But it provides services (methods) for them to use**
- Independence gives flexibility, robustness

Pattern

- Why do we need design patterns?
- It is reusable and provides solutions to deal with the problems.
- What is the character of design patterns in MVC?
- Model, model is the expert to solve the problems.

Gang of Four Pattern

- What are the categories of patterns?
- Creational, Structural, Behavioral

- Creational Patterns focus on creation (creating something).
- Structural Patterns compose complex structures.
- Behavioral Patterns negotiate the responsibilities between objects.

Design Space for GoF Patterns

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter (class)	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter (object) Bridge Composite Decorator Flyweight Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Purpose: reflects what a pattern does

Scope: domain over which a pattern applies

Singleton

- Ensure a class only has one instance, and provide a global point of access to it.

	Lazy Initialization	Eager Initialization
Defined	It will be new when user use it first time.	System will create it at the loading time.
Advantage	Reduce resource that be used, increase system efficiency.	• Reduce the complexity of code, easy to maintain. • Thread-safe. • Can use single –Thread and Multi-Thread
Disadvantage	• It probably creates two instances of the singleton class when two users use it at the same time. • Not thread-safe (only use single-Thread)	It is more a waste of resource than Lazy initialization.

Synchronize

```
public class Singleton {
    private static Singleton instance = null;
    private Singleton(){}
    synchronized static public Singleton getInstance() {
        if (instance == null) {
            instance = new Singleton();
        }
        return instance;
    }
}
```

Solution: Double-checked-locking

```
1  package com.javacodegeeks.patterns.singletonpattern;
2
3  public class SingletonLazyDoubleCheck {
4
5      private volatile static SingletonLazyDoubleCheck sc = null;
6      private SingletonLazyDoubleCheck(){}
7      public static SingletonLazyDoubleCheck getInstance(){
8          if(sc==null){
9              synchronized(SingletonLazyDoubleCheck.class){
10                 if(sc==null){
11                     sc = new SingletonLazyDoubleCheck();
12                 }
13             }
14         }
15         return sc;
16     }
17 }
18
```

- Client call instance method and method will lock the instance at first time

Example

- Train Ticket
- The seat cannot be sold to two or more people.
- So?

Software Quality Assurance

- What is “SQA”?
- Software Quality Assurance
- Talking about the Reliability and Availability, which one is more important?
- Reliability
- Why do we need “SQA”?
- Hint: subjective and objective

Requirements Metrics

$$\text{Requirements Unambiguity Q1} = \frac{\#\text{of uniquely interpreted requirements}}{\#\text{of requirements}}$$

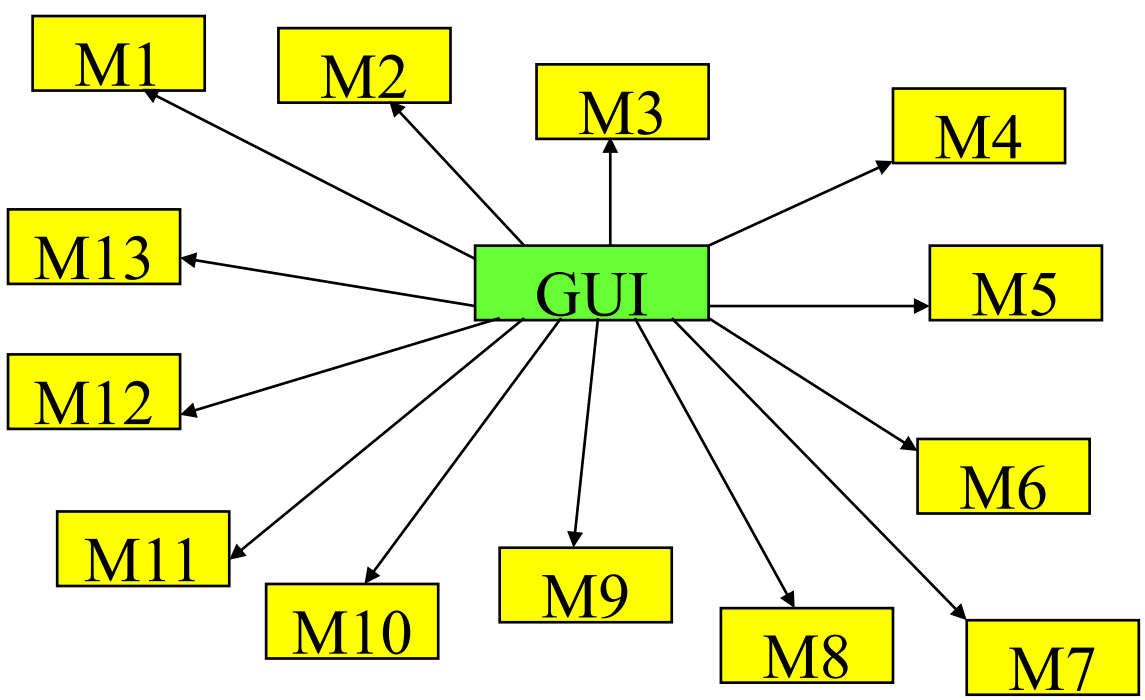
$$\text{Requirements Completeness Q2} = \frac{\#\text{of unique functions}}{\#\text{of combinations of states and stimuli}}$$

Requirements Metrics

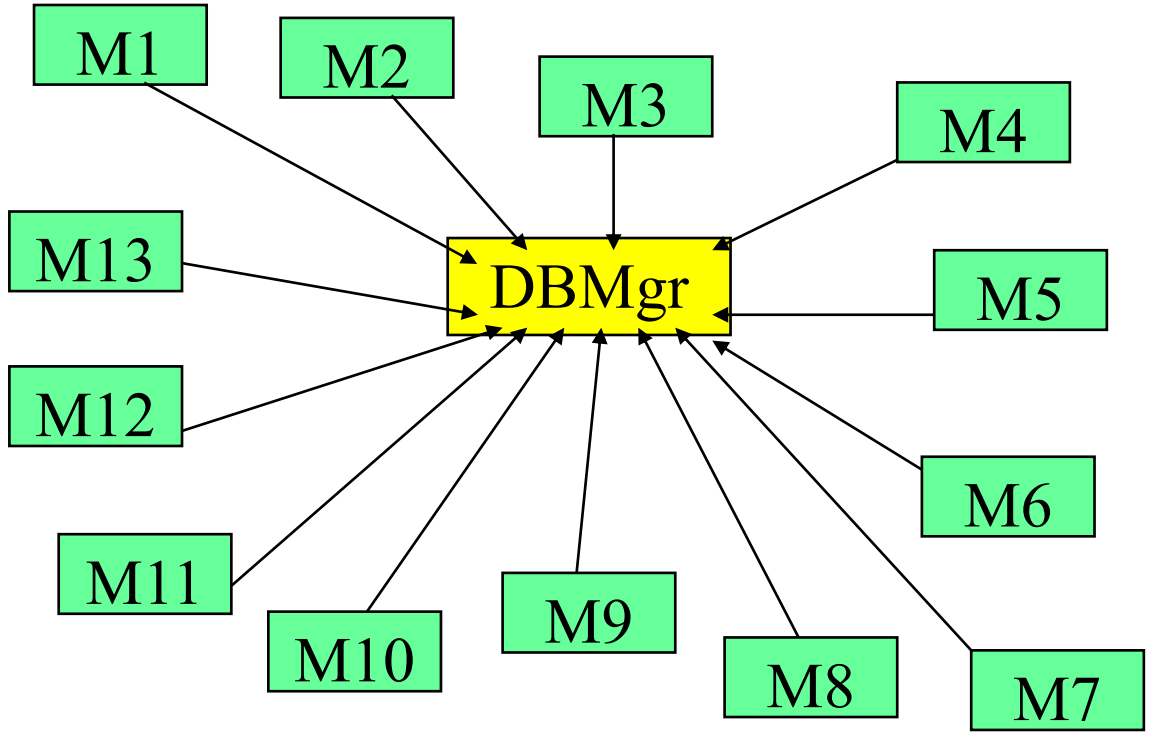
$$\text{Requirements Correctness Q3} = \frac{\#\text{of validated correct requirements}}{\#\text{of requirements}}$$

$$\text{Requirements Consistency Q4} = \frac{\#\text{of non-conflicting requirements}}{\#\text{of requirements}}$$

Fan-In & Fan-Out



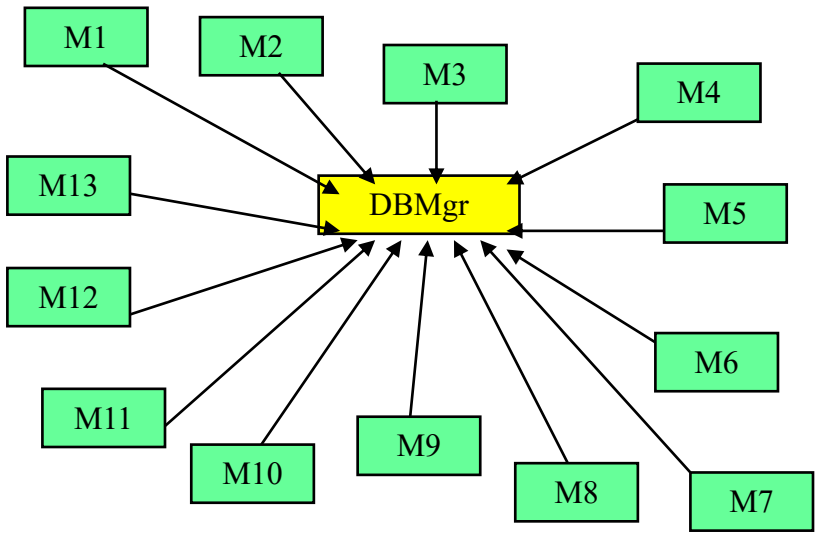
Fan-out



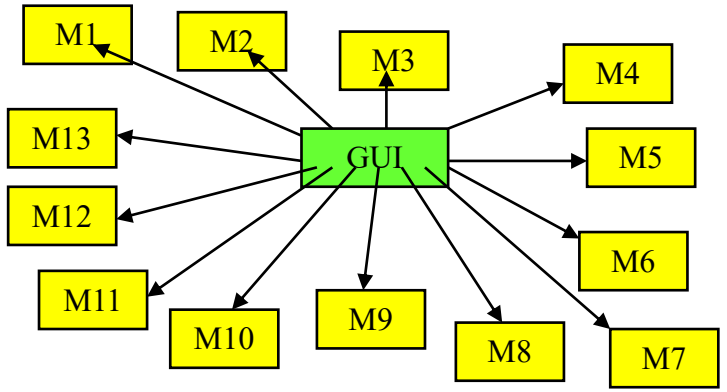
Fan-in

Fan-In & Fan-Out

- What does Fan-in mean?
- High coupling
- DB Mgr. has many requests from others.



- What does Fan-out mean?
- GUI depends others to complete itself, so it is hard to reuse.

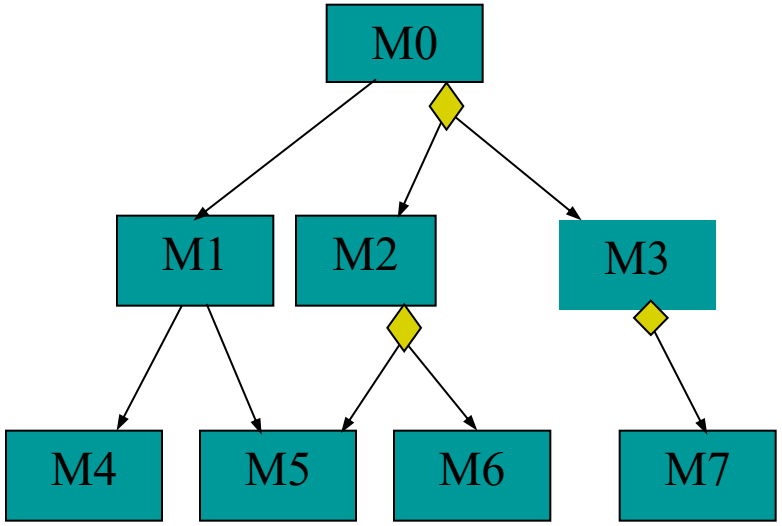


Modularity

- Modularity – Measured by cohesion and coupling metrics.
- Modularity = (a * Cohesion + b * Coupling)/(a + b), where a and b are weights on cohesion and coupling.

Module Design Complexity mdc

- The number of integration tests required to integrate a module with its subordinate modules.
- It is the number of decisions to call a subordinate module (d) plus one.

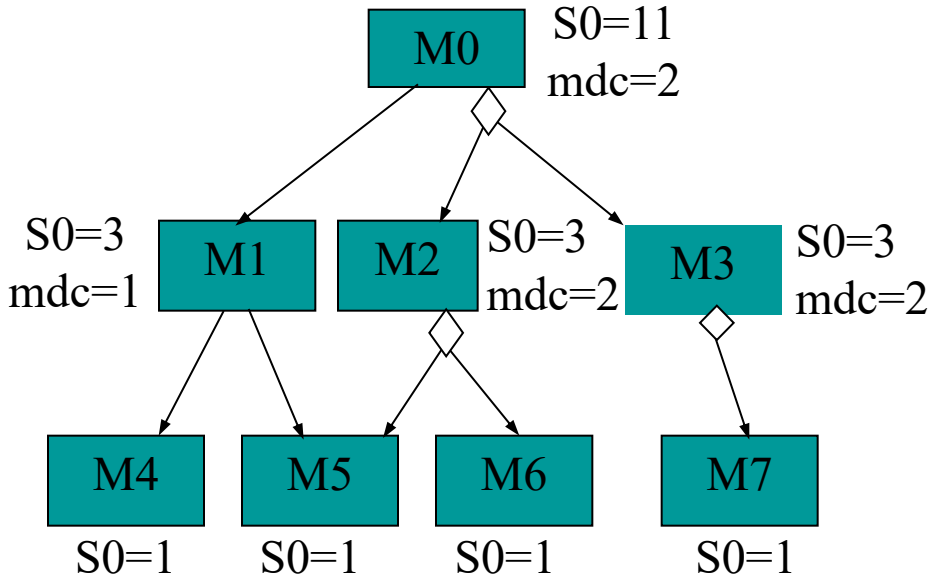


Integration Test:

Module Design Complexity mdc=4

Design Complexity S0

- Number of subtrees in the structure chart with module M as the root.
- S0(leaf) = 1
- S0(M) = mdc+S0(child1 of M)+...+S0(childn of M)

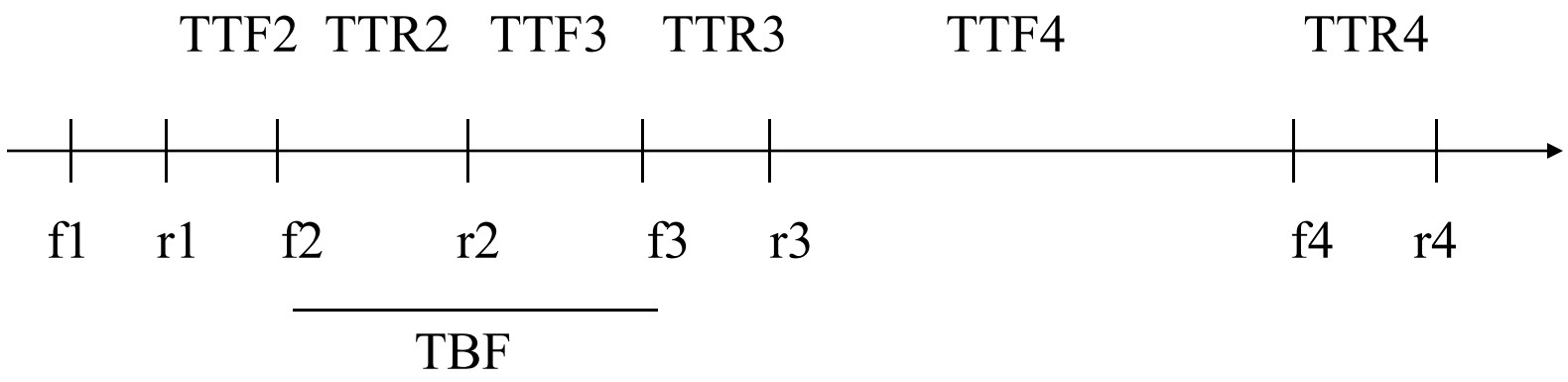


Integration Complexity S1

- Minimal number of integration test cases required to integrate the modules of a structure chart.
- Integration complexity S1(M) = S0 (M) – n + 1, where n is the number of modules in the structure chart.
- pp. 478-479

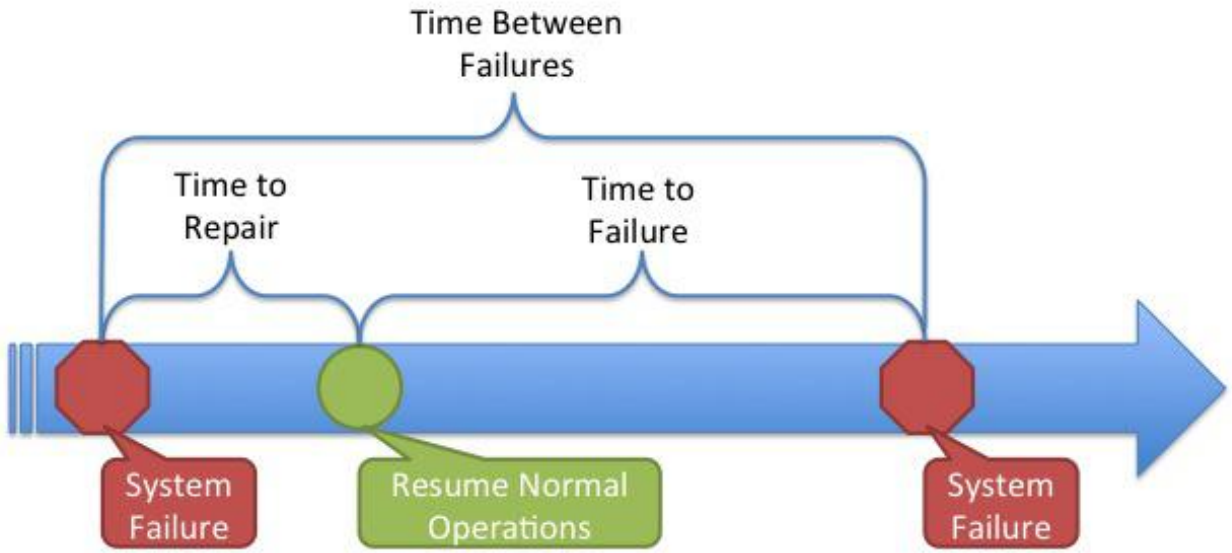
Reliability and Availability

- Mean time to failure (MTTF)
- Mean time to repair (MTTR)
- Mean time between failure (MTBF) = MTTF + MTTR
- Availability = MTTF/MTBF X 100%



MTBF, MTTR, MTTF

Differentiating Between Failure Metrics



MTBF, MTTR, MTTF

- Suppose MTTR(A)=0 and MTTR(B)=0,
- System A: during a 150-day period 5 failures
- System B: during a 500-day period 10 failures

- Questions:
- What is the MTBF(A) and MTBF(B)?
- What does the value mean? Which one has a higher reliability?

MTBF, MTTR, MTTF

- Suppose MTTR(A)=5 and MTTR(B)=2,
- System A: during a 150-day period 5 failures
  - 5 days for repairing
- System B: during a 500-day period 10 failures
  - 2 days for repairing

- Questions:
- What is the MTTF(A) and MTTF(B)?

Object-Oriented Quality Metrics

- Weighted Methods per Class (WMC).
- Depth of Inheritance Tree (DIT).
- Number of Children (NOC).
- Coupling Between Object Classes (CBO).
- Response for a Class (RFC).
- Lack of Cohesion in Methods (LCOM).

Weighted Methods per Class

- WMC(C) = Cm1 + Cm2 + ⋯ + Cmn

Cmi=complexity metrics of methods of C.

- Significance: Time and effort required to understand, test, and maintain class C increases exponentially with WMC.

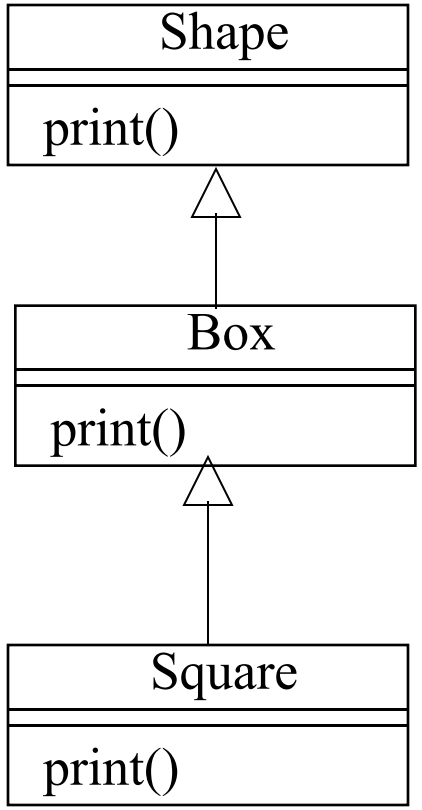
Depth of Inheritance Tree (DIT)

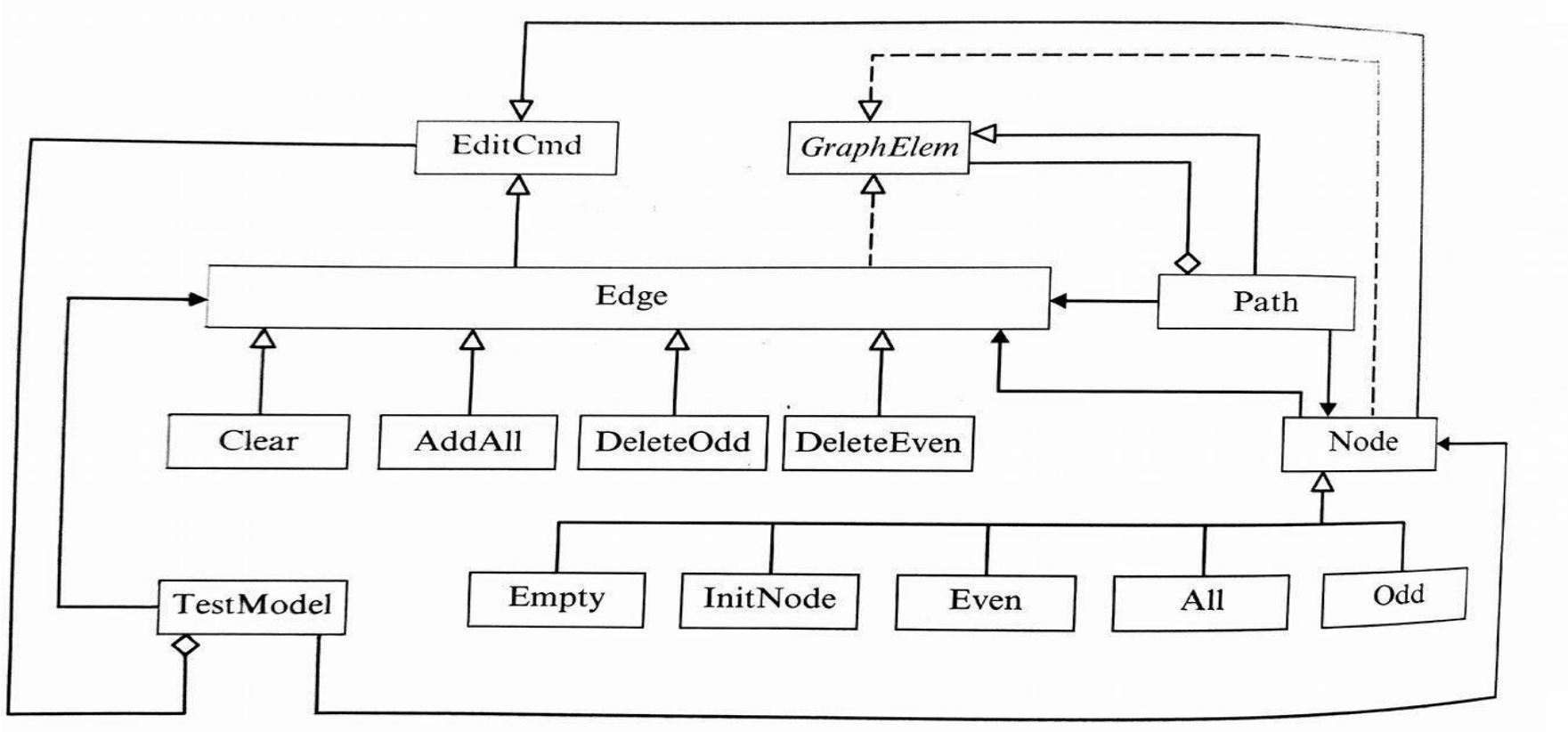
- Distance from a derived class to the root class in the inheritance hierarchy
- Measures
  - the degree of reuse through inheritance
  - the difficulty to predict the behavior of a class
  - costs associated with regression testing due to change impact of a parent class to descendant classes

High DIT Means Hard to Predict Behavior

- All three classes include print()
- It is difficult to determine which “print()” is used:

```
public static void main (...) {
   Shape p; ….
   p.print();    // which print()?
   …
}
```





- DIT = 0 : EditCmd, GraphElem, TestModel.

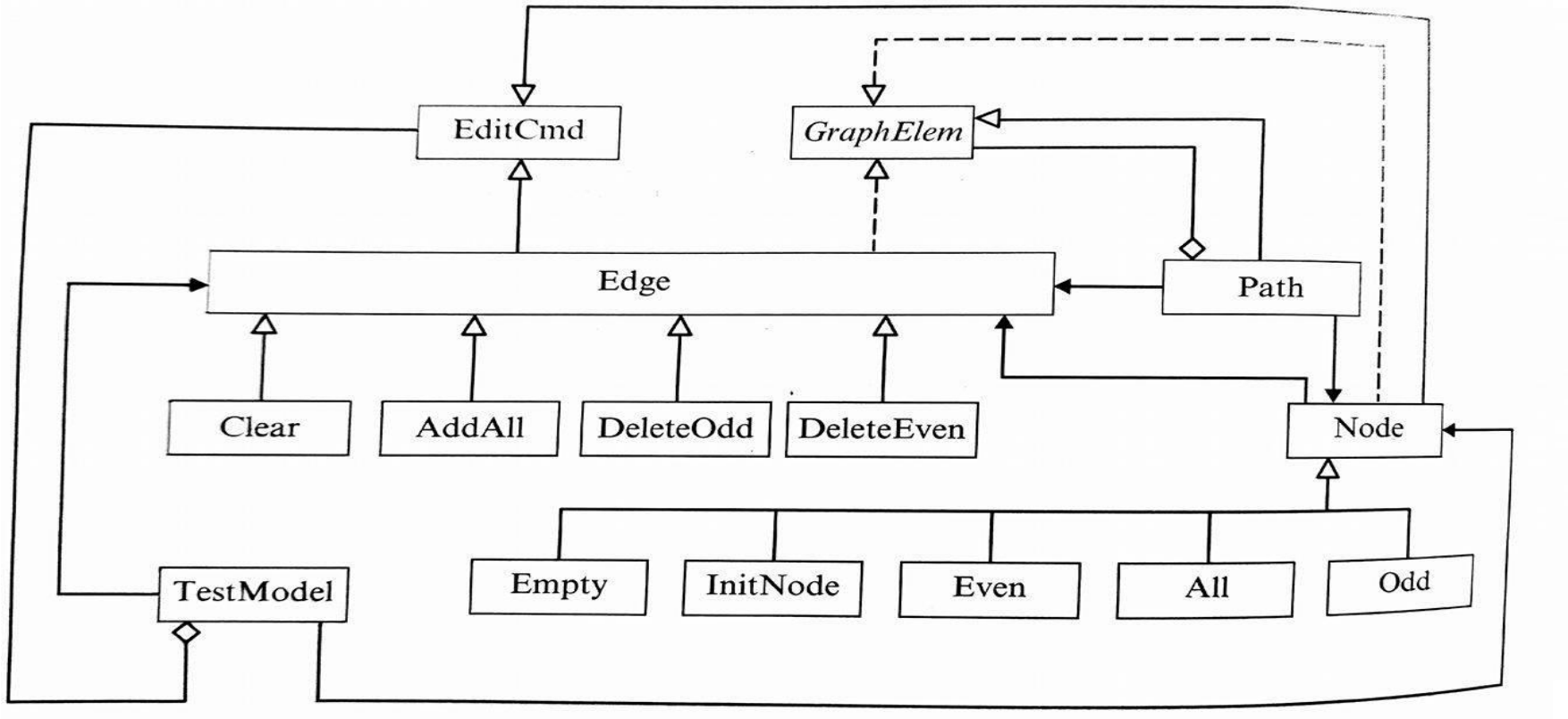
  DIT = 1 : Edge, Path, Node

  DIT = 2 : Clear, AddAll, DeleteOdd, DeleteEven, Empty, InitNode, Even, All, Odd

Number of Children

- NOC(C)

  = | { C' : C' is an immediate child of C }|
- The dependencies of child classes on class C increases proportionally with NOC.
- Increase in dependencies increases the change impact, and behavior impact of C on its child classes.
- These make the program more difficult to understand, test, and maintain.



- NOC = 0 : Empty, InitNode, Even, TestModel, etc. → Because these are the leaf nodes.

  NOC = 2 : EmitCmd

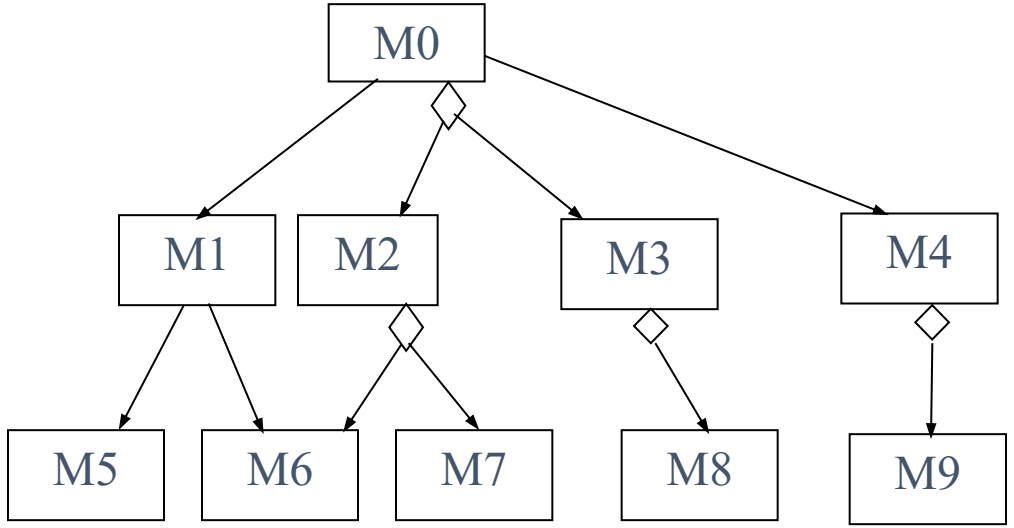
  NOC = 3 : GraphElem

  NOC = 4 : Edge

  NOC = 5 : Node

Bonus Time

- mdc(M1)
- mdc(M2)
- mdc(M3)
- mdc(M4)



Bonus Time

- S0(M9)
- S0(M8)
- S0(M7)
- S0(M6)
- S0(M5)
- S0(M4)
- S0(M3)
- S0(M2)
- S0(M1)