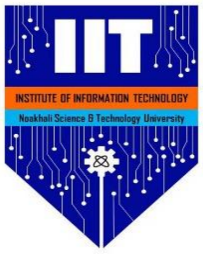

3/5/2023

Object Oriented Structure Measurement

SE 3204 – Software Metrics Lab

SUBMITTED TO

Dipok Chandra Das

Assistant Professor, IIT

Noakhali Science and Technology University

SUBMITTERD BY

Anupa Das Shormi (BFH1925020F)

Md.Rayhan Billah(ASH925028M)

Md.Rokonuzzaman(ASH1925018M)

Table of Contents

1. Measuring Coupling in Object-Oriented System

Coupling is the degree of a measure of how closely connected two routines or modules are; the strength of the relationships between modules. We can measure coupling in object-oriented system by

- Coupling Between Object Class (CBO)
- Response For Class (RFC)
- Message Passing Coupling (MPC)

1.1 Coupling Between Object Class

Coupling between objects (CBO) is a measure of how tightly coupled a group of objects or classes are to each other. It refers to the level of dependence between two or more objects, and the extent to which changes in one object require changes in the other.

High coupling means that objects are highly dependent on each other and that a change in one object is likely to affect many other objects in the system. On the other hand, low coupling means that objects are relatively independent of each other, and a change in one object is unlikely to affect other objects in the system.

Measurement Process:

- Methods Call
- Class Extends

1.2 Response For Class

RFC stands for "Response for a Class" and is a software metric used to measure the number of unique methods that can be executed in response to a message received by an object of a class. In simpler terms, it measures the number of methods that can be invoked by an object in response to a message.

The higher the value of RFC, the more complex the class is, and the more potential paths of execution it has.

Measurement Process:

RFC = number of methods in the class + number of distinct method calls made by the methods

1.3 Message Passing Coupling

MPC stands for "Message Passing Coupling" and is a software metric used to measure the degree of coupling between objects in an object-oriented software system. It measures the number of

messages that are passed between objects in a class, indicating the extent to which objects are dependent on one another to perform their respective functions.

High MPC values indicate that there are many messages being passed between objects, which can result in a tightly coupled system that is difficult to maintain, modify, and test.

Class name	RFC	MPC
Atom	0	
Compound	0	
CompoundManager	0	
Concentration	0	
Converter	0	
Converter.Step	-1	
Database	0	
DatabaseSerializer	-1	
EquationBalancer	0	
Fraction	0	
History	-1	
InsufficientDataException	-1	
InvalidAtomException	-1	
InvalidEquationException	-1	
Matrix	0	
Titration	0	
ConcentrationPanel	0	
ElectronConfigPanel	0	
EquationBalancePanel	0	
Formater	-1	
FxPieChart	0	
HistoryFrame	0	
Home	0.333333	
MolarMassPanel	0	
NeedHelpPanel	0	
PercentOfCompletionPanel	0	
Sidebar	0	
TitrationPanel	0	

2. Measuring Cohesion in Object-Oriented System

Cohesion metrics measure how well the methods of a class are related to each other. cohesion refers to the degree to which the elements inside a module belong together. We can measure Cohesion in Object-Oriented System by LCOM (Lack of Cohesion Metric)

Measurement Process:

LCOM = 1 – (sum(MF) / M*F)

Where

- M is the number of methods in class
- F is the number of instance fields in the class.
- MF is the number of methods of the class accessing a particular instance field.
- Sum(MF) is the sum of MF over all instance fields of the class.

Class name	LCOM
Atom	0
Compound	0
CompoundManager	0
Concentration	0
Converter	0
Converter.Step	-1
Database	0
DatabaseSerializer	-1
EquationBalancer	0
Fraction	0
History	-1
InsufficientDataException	-1
InvalidAtomException	-1
InvalidEquationException	-1
Matrix	0
Titration	0
ConcentrationPanel	0
ElectronConfigPanel	0
EquationBalancePanel	0
Formater	-1
FxPieChart	0
HistoryFrame	0
Home	0.333333
MolarMassPanel	0
NeedHelpPanel	0
PercentOfCompletionPanel	0
Sidebar	0
TitrationPanel	0

3. Object-Oriented Length Measure

Generally, length measures indicate the distance from one element to another. In object-oriented systems, distances depend on the perspective and the model representing an appropriate view of the system. We can measure it by DIT.

DIT stands for "Depth of Inheritance Tree" and is a software metric used to measure the depth of the inheritance hierarchy for a given class in an object-oriented software system. It is a measure of the number of super classes that a class has, either directly or indirectly. A high DIT value indicates that the class has a long inheritance chain, meaning that it inherits many properties and methods from its super classes.

Class name	DIT
Atom	0
Compound	0
CompoundManager	0
Concentration	0
Converter	0
Converter.Step	0
Database	0
DatabaseSerializer	0
EquationBalancer	0
Fraction	0
History	0
InsufficientDataException	0
InvalidAtomException	0
InvalidEquationException	0
Matrix	0
Titration	0
ConcentrationPanel	0
ElectronConfigPanel	0
EquationBalancePanel	0
Formater	0
FxPieChart	0
HistoryFrame	0
Home	0
MolarMassPanel	0
NeedHelpPanel	0
PercentOfCompletionPanel	0
Sidebar	0
TitrationPanel	0

4. Object-Oriented Reuse Measurement

We can measure Object-Oriented reuse measurement by NOC (Number of Children). NOC is computed by counting the number of immediate successors (subclasses or subinterfaces) of a class or interface.

NOC stands for "Number of Children" and is a software metric used to measure the number of immediate subclasses that inherit from a particular class in an object-oriented software system. In other words, it measures the number of classes that are directly derived from a given class. A high NOC value indicates that a class has many direct subclasses, which can make it more complex and difficult to understand, maintain, and modify.

Class name	NOC
Atom	0
Compound	0
CompoundManager	0
Concentration	0
Converter	0
Converter.Step	0
Database	0
DatabaseSerializer	0
EquationBalancer	0
Fraction	0
History	0
InsufficientDataException	0
InvalidAtomException	0
InvalidEquationException	0
Matrix	0
Titration	0
ConcentrationPanel	0
ElectronConfigPanel	0
EquationBalancePanel	0
Formater	0
FxPieChart	0
HistoryFrame	0
Home	0
MolarMassPanel	0
NeedHelpPanel	0
PercentOfCompletionPanel	0
Sidebar	0
TitrationPanel	0

5. Weighted Method Per Class

WMC stands for "Weighted Methods per Class" and is a software metric used to measure the complexity of a class in an object-oriented software system. It is a measure of the total number of methods in a class, weighted by their complexity. A high WMC value indicates that a class has many methods, and these methods are complex, making it more difficult to understand, maintain, and modify. On the other hand, a low WMC value indicates that a class has fewer methods, and these methods are less complex, which can lead to simpler, more manageable code.

Class name	WMC
Atom	15
Compound	12
CompoundManager	23
Concentration	4
Converter	6
Converter.Step	0
Database	2
DatabaseSerializer	1
EquationBalancer	13
Fraction	32
History	3
InsufficientDataException	1
InvalidAtomException	1
InvalidEquationException	1
Matrix	35
Titration	11
ConcentrationPanel	14
ElectronConfigPanel	6
EquationBalancePanel	7
Formater	4
FxPieChart	5
HistoryFrame	7
Home	8
MolarMassPanel	8
NeedHelpPanel	3
PercentOfCompletionPanel	9
Sidebar	11
TitrationPanel	35