Code Metrics from Design Perspective

Life Beyond LOC

-Elango K Sundaram

Method Level Metrics

- Cyclomatic Number
- Halstead Measures

Cyclomatic Number

- Proposed by McCabe
- Defines structural complexity of a program/method
- Cyclomatic number V(G) = Decision Nodes + 1
- The Decision Nodes in Java are as follows

  if..then
  if..then..else
  for
  while
  do..while
  case statements

Halstead Measures

- Program length
- Actual Halstead length
- Halstead length
- Program's vocabulary
- Volume
- Level
- Difficulty
- Effort
- Bug Predicted

Halstead contd..

Halstead parameters

N1 = number of operators

N2 = number of operands

n1 = number of distinct operators

n2 = number of distinct operands

Halstead Contd..

- Program length, N = n1 log n1 + n2 log n2
- Actual Halstead length = N1 + N2
- Program's vocabulary = n1 + n2
- Volume = (N1 + N2) * log (n1 + n2)
- Level = (2/n1)*(n2/N2)
- Difficulty 1/Level
- Effort (Volume/Level)/(18*60) Minutes
- Bug Predicted = Volume/3000

Method Metrics Sample

```
public class Metric {


    public void methodA(String x){
        if(x !=null) {

                for (int i=0;i<10;i++){
                        System.out.println("Metric");
                }
        }

    }

}
```

Method Metric contd..

Method Name	Type	NOE	V(G)	PL	AHL	VOC	VOL	LVL	PD	EFF	BUG
methodA	M	0.00	3.00	61.75	26.00	19.00	110.45	0.16	6.11	0.62	0.04

V(G)	**Cyclometric Number**	**VOL**	**Volume of Code**
PL	**Program Length**	**PD**	**Program Difficulty**
AHL	**Actual Halstead Length**	**EFF**	**Program effort (Minutes)**
VOC	**Program Vocubalry**	**BUG**	**Bugs Predicted (Per Statement)**

Class Level Metrics

DIT (Depth of Inheritance Tree)

- Refers to the place where the class lies with reference to its parents.
- This has reference to the maintenance and dependency
- A high DIT refers to high Reuse Level. But it could also mean difficult understandability/maintenance.
- Example

  Class A extends B {}

  Class B extends C {}

  Class C extends D {}

  DIT <A> = 3

(RFC) Response for a Class

This refers to the all the methods defined in a class along with all the methods called by the methods of the class

This is calculated by summing up the methods of the class, find the methods that each one calls and finding the unique calls

RFC contd..

Compute RFC for this class::

```
public class RFC {
    methodA1() {
        ClassB.methodB1();
    }
    methodA2() {
        ClassC.methodC1();
        ClassB.methodB1();

    }
    methodA3() {
        ///.....
        methodA3();
    }
}
```

RFC Contd..

```
public class RFC {
    methodA1() {
        ClassB.methodB1();
    }
    methodA2() {

        ClassC.methodC1();
        ClassB.methodB1();

    }
    methodA3() {
        ///.....
        methodA1();
    }
}
```

RFC = 5

NOM (No of Defined Methods)

- NOM refers to the number of methods defined in the class.
- This would refer to the methods that act as an interface between the class and the world

NOC (No. Of Children)

- NOC refers to the number of classes extending this class.
- Higher NOC would mean that this class has been re-used heavily and care must be taken during maintenance.
- Class B extends A

NOC <B> = 2

Fan In/Out

- Fan In

  No. of classes Using the class
- Fan Out

  No. Of Classes Used by this class

Fan In

Example

```
class ClassA {
    // Some things getting done here..
}
class ClassB {
    private ClassA a;
    methodB() {
        a = new ClassA();
    }
}
class ClassC {
    private ClassA a;
    methodC() {
        a = new ClassA();
    }
}
```

Fan In <A> ➔ No. of classes using A → 2.

Normally only direct dependents are used for Fain In Calculation

A Large Fain–In means a high re-use

Fan Out

```
class FanOutSample  {
        private SampleClassB sB;
        private SampleClassC sC;

        public FanOutSample() {
            sB = new SampleClassB();
            sC = new SampleClassC();
        }
        public void moreFanOut() {
            SampleClassD sD = new SampleClassD();
        }
        public void someMoreFanOut() {
            SampleClassE sE = new SampleClassE();
        }
}
Fan Out  <FanOutSample>  = 4
```

Lack Of Cohesion Methods

A cohesive class performs one function. Lack of cohesion means that a class performs more than one function, which are disparate. A high LCOM value indicates inappropriate design and high complexity. It also has indicate a high likelihood of errors.

High LCOM class should probably be split into two or more sub-classes.

- LCOM(Chidamber - Kemerer)
- LCOM (Li-Henry)
- LCOM (Henderson - Sellers)

LCOM(Chidamber - Kemerer)

Take each pair of methods in the class. If they access disjoint sets of instance variables, increase P by one. If they share at least one variable access, increase Q by one.

LCOM1 = P - Q , if P > Q

LCOM1 = 0 otherwise

LCOM1 = 0 indicates a cohesive class.

LCOM1 > 0 indicates that the class needs or can be split into two or more classes, since its variables belong in disjoint sets.

LCOM Contd..

```
public class LCOM {
	public String P="X";
	public String Q="X";
	public String R="X";
	public String S="X";
	public String T="X";
	public String U="X";
	public void methodA(){
		P = "Y";
		Q = "Y";
	}
	public void methodB(){
		U="Y";
	}
	public void methodC(){
		P = "Y";
		R = "Y";
	}
	public void methodD(){
		T ="Y";
	}
}
```

LCOM Chidamber- Kemerer

LCOM:

AB – P+

AC – Q +

AD – P +

BC – P +

BD – P +

CD – P +

LCOM = 5 –1 = 4.

Project Level Metrics

- Average Inheritance Depth
  ➔ Sum of Class Depth/Total Number of classes
- Specialization Ratio
  → Number of subclasses/ Number of superclasses
- Reuse Ratio
  → Number of superclasses/ Total number of classes
- Method Size Average

Code Comments

- Code Comments
  - Code Comment Lines (CCL)
  - Code Blank Lines (CBL)
  - Code Source Lines (CSL)
  - Code Total Lines (CTL)
  - Comment Comment Density (CCD)

  - Why do we require Blank Lines when we can use the following formula ?

    CBL = CTL- (CSL+CCL)
  - What is Comment Density ?

Open issues with Comments

- Wrong comments

  Primarily due to maintenance/code update

- Meaningless comments

  -void initialize()//This will initialize

- Auto generated Lines
  - What Do we do with them ?

THANK YOU FOR YOU TIME.

Happy Coding and Metrics.

Author:
Elango Sundaram
www.geocities.com/esundara
Chennai, India.