Metriken (metrics.exe)

Mittwoch, 16. Oktober 2013 12:03

http://www.microsoft.com/en-us/download/details.aspx?id=9422

https://www.fullstackpython.com/code-metrics.html

Metriken_Arbeitsblatt2

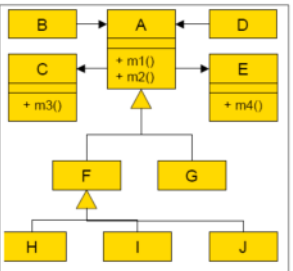
Vorstellung von Metriken und Einbau in Jenkins

Metriken: Zahlenmäßige Abbildung einer Eigenschaft.
Aber wie gut ist denn die Qualität der Software? Wie misst man die Qualität ?

Metrik	Bedeutung	Beurteilung	
Lines Of Code (LOC)	Gibt die Anzahl der Quellcodezeilen zurück	Mit Leerzeilen, ohne Leerzeilen, ohne Kommentar, mit Kommentar,et cetera. Geringe Aussagekraft	
Zyklomatische Komplexität (McCabe)	Ein komplexes System ist schwieriger zu warten und fehleranfälliger als ein einfache Komplexität ist der Grad der Verschachtelungen oder Verzweigungen.	geschachtelte IF-Verzweigungen durch logischen Operatoren verbinden drückt den Wert Schreiben von mehr Methoden drückt die Komplexität der einzelnen Methode Faustregeln: 1–10 Einfaches Programm, geringes Risiko 11–20 komplexeres Programm, erträgliches Risiko 21–50 komplexes Programm, hohes Risiko >50 untestbares Programm, extrem hohes Risiko Beginne bei einer Umstrukturierung mit der Komponente, die die höchste zyklomatische Komplexität hat	
…..		**Alle Metriken messen keine objektorientierte Programmiervorgehensweise**	

```
public int CalcGGT(int a, int b)
{
  int gr = 0;
  int kl = 0;
  int ret = 1;
  if (a == b)
  {
    ret = a;
  }
  else
  {
    gr = a;
    kl = b;
    if (a < b)
    {
      kl = a;
      gr = b;
    }
    int i = gr / 2;
    while (i >= gr / kl)
    {
      if (gr % i == 0 && kl % i == 0)
      {
        ret = i;
        i=1;
      }
      i--;
    } //end while
  } //end else
  return ret;
}
```

OO-Metriken

Metrik	Bedeutung	Beurteilung	
Weighted-Methods-of-Class-Metrik (WMC)	Anzahl der Methoden pro Klasse wird erhoben und gewichtet, z.B. nach LOC oder McCabe	Anzahl und Komplexität = Maß für Entwicklungs- und Wartungsaufwand Fehleranzahl steigt mit Höhe der Metrik Außerdem wird der Einfluss auf die Kindklassen erhöht.	
Depth-of-Inheritance-Tree-Metrik (DIT)	Maximaler Weg von der Wurzel bis zur Kindklasse. DIT(A) = 0, DIT(G) = 1, DIT(H) = 2	Bei Vererbung erbt jede Kindklasse alle Eigenschaften aller Vaterklassen ➔ Fehleranfälligkeit und Aufwand der Fehlerbeseitigung steigt (besonders bei Fehlern der Basisklasse)	
Number of Children (NOC)	Anzahl der direkten Unterklassen NOC(a) = 2, NOC(B) = 0, NOC(F) = 3	Je größer die Anzahl der direkten Kindklassen, desto höher die Wiederverwendung, desto schlechter die Abstraktion der Vaterklasse.	**class A** **{** **public void callB()** **{** **B b = new B();** **b.callC();** **}** **public void doNothingA() {** **;** **}** **}** **class B** **{** **public void callC** **{**
Coupling Between Object Classes (CBO)	Kopplungen zwischen Klassen. Eine Klasse A ist genau dann mit einer Klasse B gekoppelt, wenn A Methoden von B aufruft oder Instanz-variablen von B nutzt. Die Kopplung von A erhöht sich auch dann, wenn eine Klasse C Methoden der Klasse A aufruft oder Instanzvariablen von A nutzt. CBO(A) = 4	Je höher die Kopplung, desto schwieriger die Isolierbarkeit, umso schwieriger der Test umso schwieriger die Wiederverwendung. Änderungen können weitreichende Auswirkungen haben.	

RFC (Response for a Classe	Menge aller Methoden der eigenen Klasse und aller Methoden benachbarter Klassen, die direkt von den eigenen Methoden gerufen werden. Methoden, die wiederum von benachbarten Klassen aufgerufen werden, gehören nicht zum Response Set. **RS(A) := {A.callB();** **A.doNothingA();** **B.callC()}** **RFC(B) = 2** **RFC(C) = 1**	Komplexitätsmaß für die jeweilige Klasse. Gibt eine Obergrenze für notwendige Testfälle an, die durch den Aufruf der Methoden notwendig werden. Je höher der Wert, desto aufwändiger der Test.	**C c = new C();** **c.doNothingC();** **}** **}** **class C** **{** **public doNothingC()** **{** **;** **}** **}**
Lack Of Cohesion in Methods (LCOM)	Anzahl der Methoden ohne gemeinsame Instanzvariable minus der Anzahl Methoden mit gemeinsamen Instanzvariablen.	Hoher Zusammenhalt ist dann gegeben, wenn Instanzvariablen und Methoden zusammenpassen, d.h. Methoden benutzen häufig die Instanzvariablen. Verwenden zwei Methoden dieselben Instanzvariablen, gehören sie zusammen	**class LackOfCohesion** **{** **private int a;** **private int b;** **public IncA() {** **a = a + 1;** **}** **public IncB() {** **b = b + 1;** **}** **}**