# Seminar "Softwareentwicklung in der Wissenschaft" "Code-Qualität"

Johann Weging
8weging@informatik.uni-hamburg.de

Arbeitsbereich Wissenschaftliches Rechnen
Department Informatik
Universität Hamburg

2011-02-09

# Inhalt

# Code-Qualität

## Definition

Die Sicherstellung der Code-Qualität fördert die Verständlichkeit und Wartbarkeit des Quellcodes. Erreicht wird dies durch formale Sauberkeit und eine verständliche Struktur. [1]

[1] Solid Code [1]

# Ziele von Code-Qualität

- Verbesserung der Lesbarkeit
- Bessere Verständlichkeit
- Vereinfachte Wartbarkeit
- Einfaches ändern

2 3

[2] Solid Code [1]
[3] Wikipedia [12]

- Aufbau des Projekts
- Formale Sauberkeit
- Aussagekräftige Benennung von Variablen und Funktionen
- Sauberes Design
- Gründliche Dokumentation

# Dateilisting ECOHAM

```
[...]
NSriver.inc
NStime.inc
NStopo.inc
NSvelfield.inc
PDIndexL21.asc
River.dat
ZetaM2.dat
biogeo.f90
biogeo.lst
biogeo.o
dep.dat
eco4.f90
eco4.lst
eco4.o
eco4_atm_n_mon_2001.dat
eco4_atm_n_mon_2002.dat
eco4_atm_n_mon_2003.dat
eco4_atm_n_mon_2004.dat
eco4_flu1.com
eco4_flu1.read
eco4_flu2.com
eco4_flud.com
eco4_indh.dat
eco4_neigh.dat
eco4_neigh_sk.dat
[...]
```

# Einfache Ordnerstruktur

| | |
|---|---|
| build | Compilat |
| doc | Dokumentation |
| src | Quellcode |
| model-input | Eingabedaten |
| model-output | Ausgabedaten |

```
/project-root
  /build
    main.o
  /doc
    README.txt
  /src
    main.c
  /model-input
    input.dat
  /model-output
    output.dat
```

# Sauberer Quellcode

```
1  float suml(float* l,int c){
2  float s=0;
3  for(int i=0;i<c;++i){s+=l[i];}
4  return s;}
```

So nicht!

# Extra Zeilen hinzufügen.

```
1  float suml(float* l,int c)
2  {
3
4  float s;
5
6  for(int i=0;i<c;++i)
7  {
8  s+=l[i];
9  }
10
11 return s;
12
13 }
```

## Zusätzliche Leerzeichen verwenden.

```
1  float suml(float* l, int c)
2  {
3
4  float s = 0;
5
6  for(int i = 0; i < c; ++i)
7  {
8   s += l[i];
9  }
10
11 return s;
12 }
```

# Code einrücken.

```
1  float suml(float* l, int c)
2  {
3
4    float s = 0;
5
6    for(int i = 0; i < c; ++i)
7    {
8      s += l[i];
9    }
10
11   return s;
12 }
```

Aussagekräftige Variablennamen (selbsterklärender Code).

```
1  float sumUpList(float* list, int listElementCount)
2  {
3
4    float sum = 0;
5
6    for(int index = 0; index < listElementCount; ++index)
7    {
8      sum = sum + list[index];
9    }
10
11   return sum;
12 }
```

# Kommentare verwenden.

```
1
2   /**
3   *\author Johann Weging
4   *\brief This function sums up all elements in a list.
5   *
6   * This function sums up all elements in a list of floats and returns the sum
7   * as a float. It although needs the count of the elements in the list.
8   *
9   *\param [in] list a list of floats to sum up.
10  *\param [in] listElementCount a integer specify the count of the
11  * list elements.
12  *\param [out] sum a integer containing the sum.
13  */
14  float sumUpList(float* list, int listElementCount)
15  {
16
17    float sum = 0; // Set sum to zero
18
19    /*Iterate over the list elements*/
20    for(int index = 0; index < listElementCount; ++index)
21    {
22      /*Add the current element to the sum*/
23      sum = sum + list[index];
24    }
25
26    return sum; // Return the sum
27  }
```

# Style Guides

| | |
|---|---|
| C | Indian Hill C Style and Coding Standards |
| Fortran | European Standards For Writing and Documenting Exchangeable Fortran 90 Code |

# Genauigkeit

- Explizite Definition der Genauigkeit
- Genauigkeit setzten durch Compiler Flags
- Verwenden einer Genauigkeit

- IEEE 754-2008

Explizit:

```
1  a = sqrt(2.0)      !single Precision
2  b = sqrt(2.0d0)    !double Precision
3  c = sqrt(dble(2)) !double Precision
4
5  a => 1.41421353816986
6  b => 1.41421356237310
7  c => 1.41421356237310
```

# Messwerte

- Anzahl Codezeilen
- Anzahl Klassen
- Anzahl Felder
- Anzahl Methoden
- Anzahl Methoden pro Klasse

# Codezeilen

- Codezeilen mit Withespace: 12
- Codezeilen ohne Withespace: 8

```
1   /*This Function calculates the absolute value of an integer*/
2   int abs(int value)
3   {
4
5     if(value < 0)
6     {
7       /*change the algebraic sign*/
8       value = value * -1;
9     }
10
11    return value;
12  }
```

# Metriken

- Klassen Kopplung
- Zyklomatische Komplexität
- Halstead Metriken
- Wartbarkeitsindex
- Testabdeckung

# Klassen Kopplung

- Geringe Kopplung
- Hohe Kohäsion

=> Aussagen über Wiederverwendbarkeit des Codes [4]

[4] Solid Code [1]

# Zyklomatische Komplexität

- Misst die Komplexität eines Programms
- Definiert über den Kontrollflussgraphen

M Zyklomatische Komplexität

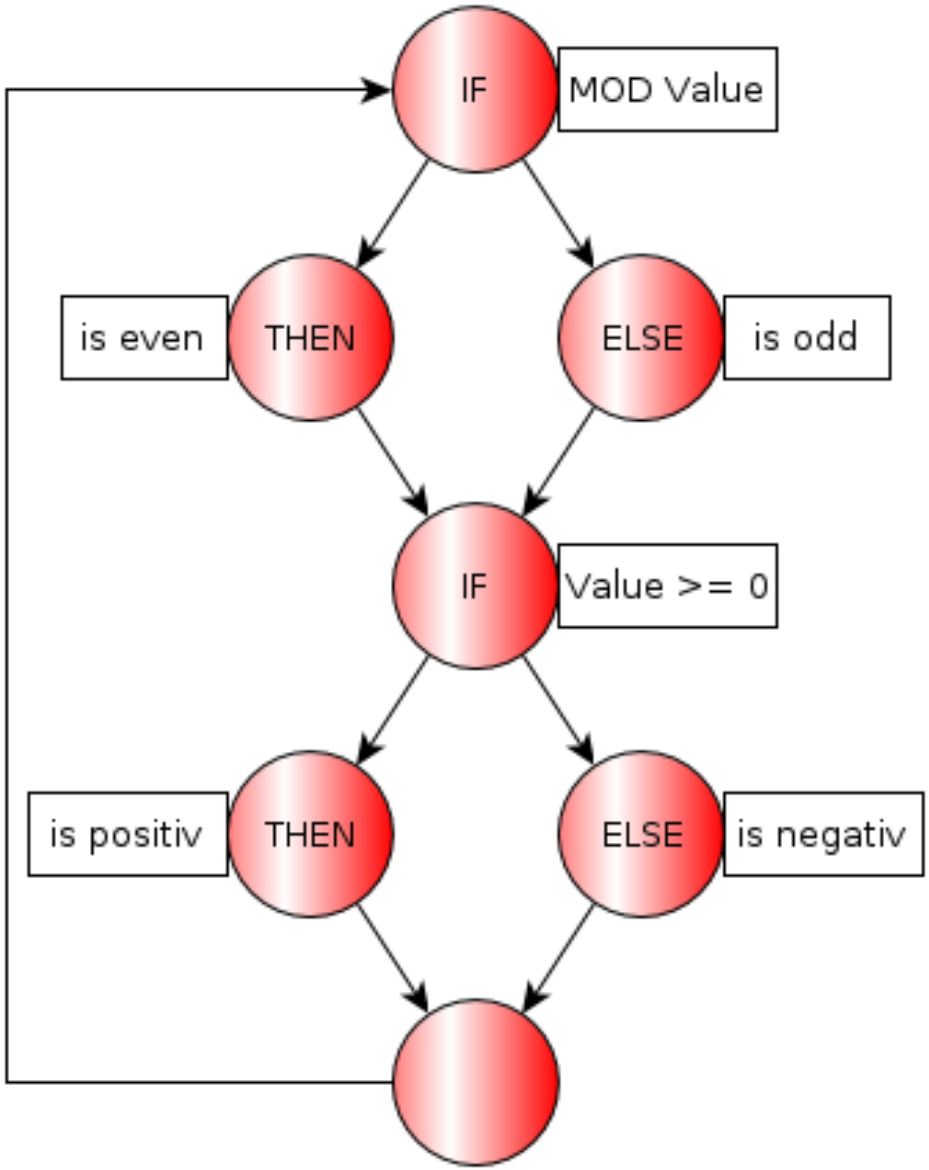
E Anzahl Kanten

N Anzahl Knoten

P Anzahl verbundener Komponenten

$$M = E - N + 2P$$

```
1  INTEGER :: Value
2
3  READ(*,*)  Value !read the Value
4
5  IF (MODULO(Vlaue, 2) == 0) THEN !modulo of Value = 0
6    WRITE(*,*)  Value , ' is even'
7
8  ELSE !modulo of Value != 0
9    WRITE(*,*)  Value, ' is odd'
10 END IF
11
12 IF(Value >= 0) THEN !Value is positive
13   WRITE(*,*) Value, ' is positive'
14
15 ELSE ! Value is negative
16   WRITE(*,*) Value, ' is negative'
17 END IF
```




$$E = 9$$
$$N = 7$$
$$P = 1$$

$$M = E - N + 2P$$
$$M = 9 - 7 + 2 \cdot 1$$
$$M = 4$$

# Halstead Metriken

- Misst die Komplexität des Quellcodes
- Definiert über die Operatoren und Operanden

| | |
|---|---|
| OP | Anzahl Operatoren |
| UOP | Anzahl einmaliger Operatoren |
| OD | Anzahl Operanden |
| UOD | Anzahl einmaliger Operanden |

| | |
|---|---|
| LTH | Länge |
| VOC | Vokabular |
| DIF | Schwierigkeit |
| VOL | Volumen |
| EFF | Aufwand |
| BUG | Bugs |

$$LTH = OP + OD$$

$$VOC = UOP + UOD$$

$$DIF = \frac{UOP}{2} \cdot \frac{OD}{UOD}$$

$$VOL = LTH \cdot \log_2(VOC)$$

$$EFF = DIF \cdot VOL$$

$$BUG = \frac{VOL}{3000}$$

```
1       int x = 0;
2       x = x + 1;
```

$$OP = 5$$
$$UOP = 3$$
$$OD = 6$$
$$UOD = 4$$
$$LTH = OP + OD = 5 + 6 = 11$$
$$VOC = UOP + UOD = 3 + 4 = 7$$
$$DIF = \frac{UOP}{2} \cdot \frac{OD}{UOD} = \frac{3}{2} \cdot \frac{6}{4} = 2.25$$
$$VOL = LTH \cdot \log_2(VOC) = 11 \cdot \log_2(7) = 30.88$$
$$EFF = DIF \cdot VOL = 2.25 \cdot 30.88 = 69.48$$
$$BUG = \frac{VOL}{3000} = \frac{30.88}{3000} = 0.01$$

# Wartbarkeitsindex

| | |
|---|---|
| MI | Wartbarkeitsindex |
| MIwoc | MI ohne Kommentare |
| MIwc | MI mit Kommentare |
| aveVOL | Durchschnittliches Volumen des Codes |
| aveM | Durchschnittliche zyklomatische Komplexität |
| aveLOC | Durchschnittliche Zeilenanzahl |
| perCM | Prozentualer Kommentaranteil |

$$MIwoc = 171 - 5.2 \cdot \ln(aveVOL) - 0.23 \cdot aveM - 16.2 \cdot \ln(aveLOC)$$
$$MIcw = 50 \cdot \sin(\sqrt{2.4 \cdot perCM})$$
$$MI = MIwoc + MIwc$$

# Dokumentationsqualität

## Definition

Die Softwaredokumentation soll zu dem Verständnis der Software beitragen und den Anwender informieren, so dass dieser in der Lage ist, die ihm gestellten Aufgaben zu erfüllen.[5]

[5] software-kommunikation.net [6]

# Qualität

- Korrektheit
- Vollständigkeit
- Angemessenheit
- Konsistenz / Einheitlichkeit

6

[6] software-kommunikation.net [6]

# Dokumentieren mit Doxygen

## Doxygen

Doxygen ist ein Dokumentationssystem für C, C++, Java, Fortran [...]. [7]

[7] www.stack.nl/ dimitri/doxygen/ [10]

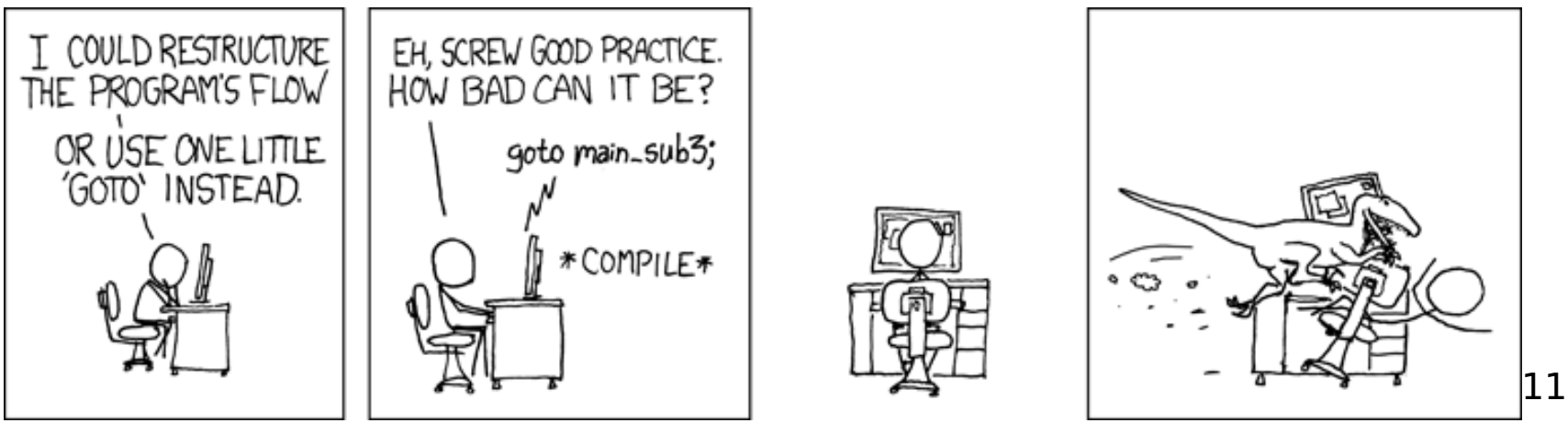
```
1  /**
2  *\author Johann Weging
3  *\brief This function sums up all elements in a list.
4  *
5  * This function sums up all elements in a list of floats and
       returns the sum
6  * as a float. It although needs the count of the elements in the
       list.
7  *
8  *\param [in] list a list of floats to sum up.
9  *\param [in] listElementCount a integer specify the count of the
10 * list elements.
11 *\param [out] sum a integer containing the sum.
12 */
13 float sumUpList(float* list, int listElementCount){[...]}
```

# Zusammenfassung

- Auf Code-Qualität achten
- Style Guides befolgen
- Checkliste häufiger Fehler durchgehen [8]
- Tools zum analysieren der Code-Qualität verwenden [9] [10]
- Es gibt nicht ßu viel"Dokumentation

[8] microsoft.com [7]
[9] Tool Support for Inspecting the Code Quality of HPC Applications [13]
[10] metrics.sourceforge.net [5]

# Danke für Ihre Aufmerksamkeit



[11]

[11] xkcd.com [14]

# Quellen

1. Donis Marschall & John Bruno, Soid Code, Microsoft Press Deutschland, 2009
2. Komplexität und Qualität von Software, In: MSCoder 1/2007, Seite 36-43
3. IEEE 754-2008
4. http://www.cs.arizona.edu/ mccann/cstyle.html
5. http://metrics.sourceforge.net/
6. http://software-kommunikation.net/qm_qualitaet.html

7 http://msdn.microsoft.com/en-us/library/ms182021%28v=vs.90%29.aspx

8 http://www.virtualmachinery.com/sidebar2.htm

9 http://en.wikipedia.org/wiki/Halstead_complexity_measures

10 http://www.stack.nl/~dimitri/doxygen/

10 http://research.metoffice.gov.uk/research/nwp/numerical/fortran90/f90_standards.html

11 https://srv.rz.uni-bayreuth.de/lehre/fortran90/vorlesung/V10/V10.html

12 http://en.wikipedia.org/wiki/Software_quality#Source_code_quality

13 T. Panas, D. Quinlan, R. Vuduc, Tool Support for Inspecting the Code Quality of HPC Applications 2007

14 http://xkcd.com/292/

Alle Internetquellen wurden am 2011-02-09 geprüft.