

Software-Metriken

10 Metriken

Messen und Maße
Softwaremetriken

Bedeutung des Messens

"To measure is to know."

Clerk Maxwell, 1831-1879

"A science is as mature as its measurement tools."

Louis Pasteur, 1822-1895

"Miss alles, was sich messen lässt, und mach alles messbar, was sich nicht messen lässt. "

Galileo Galilei, 1564-1642

Bedeutung des Messens

"To measure is to know."

Clerk Maxwell, 1831-1879

"A science is as mature as its measurement tools."

Louis Pasteur, 1822-1895

"Miss alles, was sich messen lässt, und mach alles messbar, was sich nicht messen lässt. "

Galileo Galilei, 1564-1642

"Messen können Sie vieles, aber das Angemessene erkennen können Sie nicht."

Hans Gadamer

Definitionen

Maß: Abbildung von einem beobachteten (empirischen) Beziehungssystem auf ein numerisches Beziehungssystem

Maß: Abbildung von Eigenschaften von Objekten der realen Welt auf Zahlen oder Symbole

Metrik: Abstandsmaß (math.)

Messen: Anwendung eines Maßes auf ein Objekt

Definitionen für Software-Metriken

“A quantitative measure of the degree to which a system, component, or process possesses a given variable.”

– IEEE Standard Glossary

“A software metric is any type of measurement which relates to a software system, process or related documentation.”

– Ian Sommerville

Messen und Softwaretechnik

- Beschreibung: kompakt und objektiv
- Beurteilung: Qualität, Wartbarkeit, ...
- Vorhersage: geordnete Planung

Probleme bei Maßen

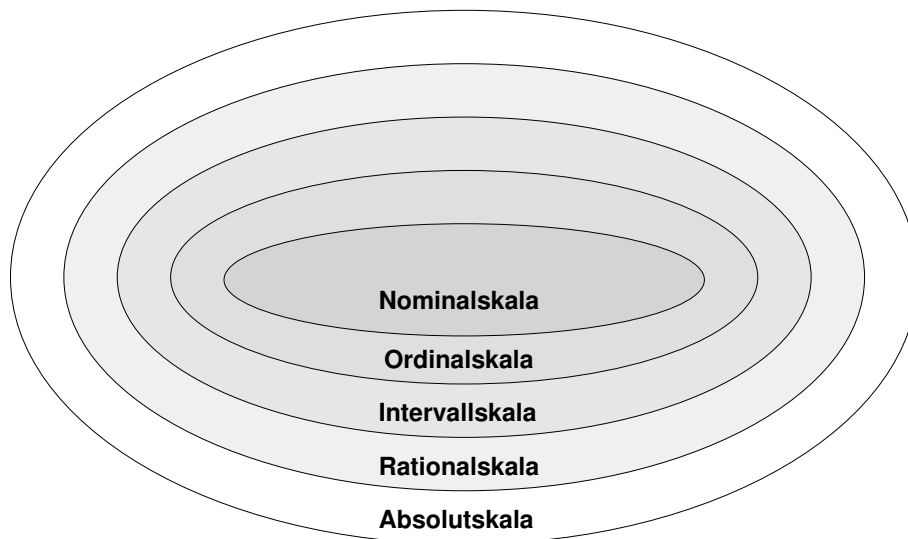
- Repräsentanz
 - Eindeutigkeit
 - Bedeutung
 - Skalierung
- Darstellung als Zahl sinnvoll möglich?
viele Abbildungen möglich
erhalten bei Transformationen
welche?

Skalen

- „20 Prozent Verbesserung der Qualität“
- „Dieser Kunde ist doppelt so zufrieden wie jener“
- „Heute doppelt so warm wie gestern“

→ Skala?

Skalenhierarchie



Skalenhierarchie – Nominalskala

1. Nominalskala

- ungeordnete 1:1 Abbildung
- Transformationen: beliebige 1:1
- Operationen: =, \neq
- Statistiken: Häufigkeit
- Beispiel: Programmiersprachen

ADA C C++ Java ...

Skalenhierarchie – Ordinalskala

2. Ordinalskala

- dazu: vollständige Ordnung
- Transformationen: streng monoton steigend
- Operationen: $<$, $>$
- Statistiken: Median
- Beispiel: Prioritäten

niedrig $<$ mittel $<$ hoch

Skalenhierarchie – Intervallskala

3. Intervallskala

- dazu: Distanzfunktion
- Transformationen: $M' = aM + b$ ($a > 0$)
- Operationen: +, −
- Statistiken: Mittelwert, Standardabweichung
- Beispiel: Temperatur

$$T_{Celsius} = \frac{5}{9} \cdot (T_{Fahrenheit} - 32)$$

Definition Metrik

Metrik: Distanzfunktion $d : A \times A \rightarrow \mathbb{R}$, mit:

- $d(a, b) \geq 0 \quad \forall a, b \in A, \quad d(a, b) = 0 \Leftrightarrow a = b$
- $d(a, b) = d(b, a) \quad \forall a, b \in A$
- $d(a, c) \leq d(a, b) + d(b, c) \quad \forall a, b, c \in A$

Skalenhierarchie – Rationalskala

4. Rationalskala

- dazu: Maßeinheit, Nullpunkt
- Transformationen: $M' = aM$ ($a > 0$)
- Operationen: /
- Statistiken: geom. Mittel, Korrelation
- Beispiel: Länge (Kelvin)

$$L_{Meter} = L_{Meilen} \cdot 1609$$

Skalenhierarchie – Absolutskala

5. Absolutskala

- Metrik steht für sich selbst, kann nicht anders ausgedrückt werden
- Transformationen: nur die Identität $M' = M$
- Operationen: absoluter Vergleich
- Beispiele:
 - Zähler: Anzahl Personen in einem Projekt
 - Wahrscheinlichkeit eines Fehlers
 - LOC für Anzahl Codezeilen
 - **nicht**: LOC für Programmlänge

Durchführung von Messungen

- Maße definieren
- Messwerte sammeln
- Messwerte analysieren, interpretieren
- Gefahr von Zahlenfriedhöfen

Zielorientiertes Messen

GQM (Goal-Question-Metric; Basili et al., 1984)

Nicht das messen, was einfach zu bekommen ist,
sondern das, was benötigt wird

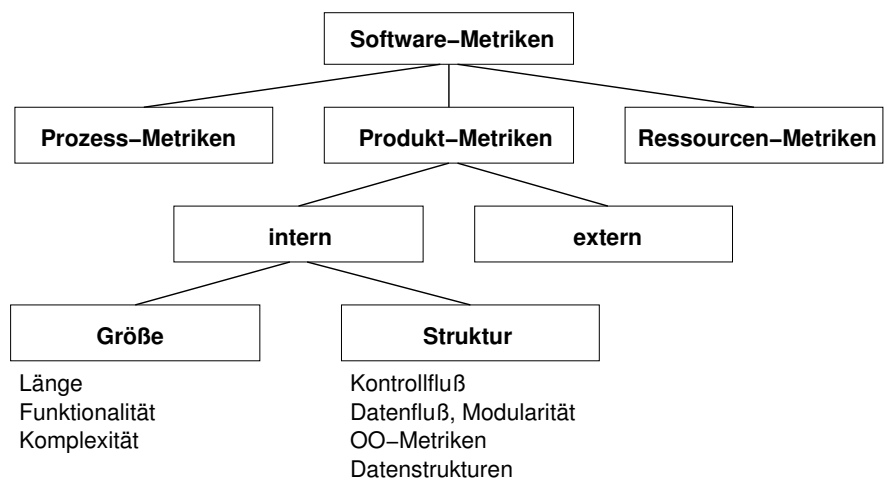
- ① Ziele erfassen
- ② zur Prüfung der Zielerreichung notwendige Fragen ableiten
- ③ was muss gemessen werden, um diese Fragen zu beantworten

Gütekriterien für Metriken

- Objektivität
- Validität
- Zuverlässigkeit
- Nützlichkeit
- Normiertheit
- Vergleichbarkeit
- Ökonomie

(nach Balzert)

Klassifikation nach Fenton (1996)



Größenmetriken – LOC

Lines of code (LOC)

- + relativ einfach messbar
- + starke Korrelation mit anderen Maßen
- ignoriert Komplexität von Anweisungen und Strukturen
- schlecht vergleichbar

Größenmetriken – LOC

```
int main(int argc, char **argv) {  
    printf("Hello World."); }  
}
```

Größenmetriken – LOC

```
/*
 * This program prints the message
 * "Hello World." to stdout.
 */

int main(int argc,
         char **argv)
{
    printf("Hello World.");
}
```

Größenmetriken – LOC

```
/*
 * This function should be documented.
 *
 * Author:
 * Date created:
 * Date modified:
 * Version:
 *
 */

int main(int argc, char **argv)
{
    printf("Hello World.");
}
```

Größenmetriken – Halstead

Halstead (1977)

Länge	$N = N_1 + N_2$
Vokabular	$\mu = \mu_1 + \mu_2$
Volumen	$V = N \cdot \log_2 \mu$
Program Level	$L_{est} = (2/\mu_1) \cdot (\mu_2/N_2)$
Programmieraufwand	$E_{est} = V/L_{est}$

mit μ_1, μ_2 = Anzahl unterschiedlicher Operatoren, Operanden
 N_1, N_2 = Gesamtzahl verwendeter Operatoren, Operanden

- + komplexe Ausdrücke und viele Variablen berücksichtigt
- Ablaufstrukturen unberücksichtigt

Größenmetriken – Halstead

```
void sort( int *a, int n )
{
    int i, j, t;
    if ( n < 2 ) return;
    for ( i = 0; i < n-1; i ++ ) {
        for ( j = i + 1; j < n; j ++ ) {
            if ( a[i] > a[j] ) {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }
    }
}
```

Größenmetriken – Halstead

3	<	3	{	1	0
5	=	3	}	2	1
1	>	1	+	1	2
1	-	2	++	6	a
2	,	2	for	8	i
9	;	2	if	7	j
4	(1	int	3	n
4)	1	return	3	t
6	[]				

$$\mu_1 = 17, \mu_2 = 7, N_1 = 50, N_2 = 30$$

$$\Rightarrow V = N \cdot \log_2(\mu) = 80 * \log_2(24) = 391,9$$

Größenmetriken – Function Points

Function Points (Albrecht, 1979)

- Messung der Funktionalität anhand der Spezifikation
- Zählen von externen Eingaben/Ausgaben/Anfragen/Schnittstellen
- Gewichtung anhand Komplexitätsfaktor
- ergibt sich aus 14 Einzelkomponenten

Anwendung: vor allem Aufwandsabschätzung

Größenmetriken – weitere

weitere:

- Anzahl Module
- Anzahl Operatoren, Operanden, Schlüsselworte
- Anzahl Parameter
- Anzahl/Umfang von Klonen
- durchschnittliche Länge von Bezeichnern
- ...

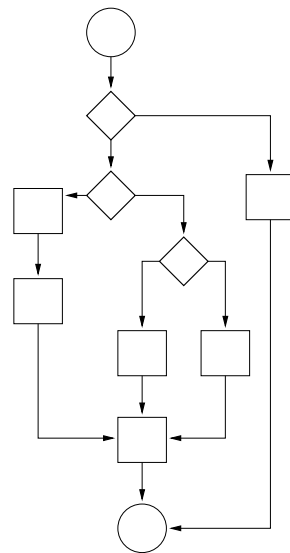
Strukturmetriken

- Eigenschaften des Kontrollflussgraphen
- Eigenschaften des Aufrufgraphen (Größe, Tiefe, Breite)
- Modulkohäsion, Modulkopplung (Abhängigkeiten)
- OO-Metriken
- Daten, Datenstrukturen

Strukturmetriken – Kontrollflussgraph

Eigenschaften des Kontrollflussgraphen

- Anzahl Knoten
- Anzahl Kanten
- maximale Tiefe
- abgeleitete Maße



Komplexitätsmetriken – McCabe

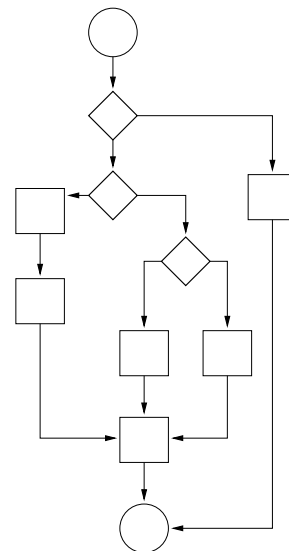
McCabe's cyclomatic complexity (1976)

$$V(g) = \#Kanten - \#Knoten + 2$$

(im Kontrollflussgraph), oder einfacher:

$$V(g) = \#Binärverzweigungen + 1$$

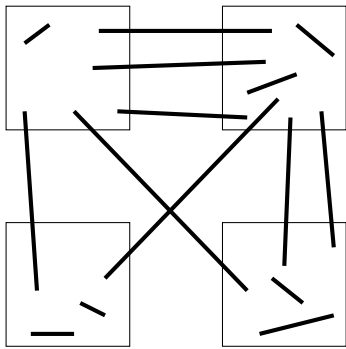
- + einfach zu berechnen
- Komplexität von Anweisungen unberücksichtigt



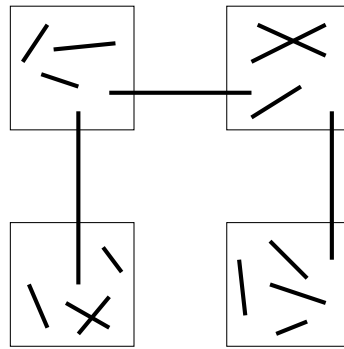
Komplexitätsmetriken – McCabe

```
void sort( int *a, int n )
{
    int i, j, t;
    if ( n < 2 ) return;
    for ( i = 0; i < n-1; i ++ ) {
        for ( j = i + 1; j < n; j ++ ) {
            if ( a[i] > a[j] ) {
                t = a[i];
                a[i] = a[j];
                a[j] = t;
            }
        }
    }
}
```

Strukturmetriken – Kopplung und Kohäsion



starke Kopplung
schwache Kohäsion



schwache Kopplung
starke Kohäsion

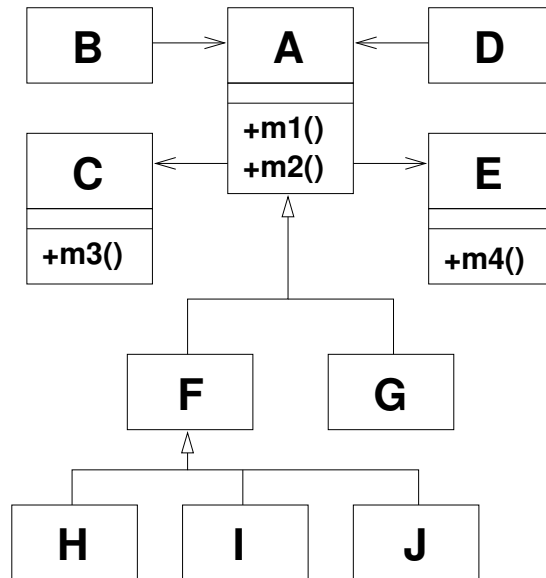
Strukturmetriken – OO

OO-Metriken (Chidamber und Kemerer, 1994):

- WMC - weighted methods per class
- DIT - depth of inheritance tree
- NOC - number of children
- CBO - coupling between objects (uses, used-by)
- RFC - response for a class (#own + #called methods)
- LCOM - lack of cohesion in methods

Metriken pro Klasse

Strukturmetriken – OO



Strukturmetriken – OO

Metrics for OO Design (MOOD, Abreu 1995):

- MHF - method hiding factor
- AHF - attribute hiding factor
- MIF - method inheritance factor
- AIF - attribute inheritance factor
- PF - polymorphism factor
- CF - coupling factor

Metriken pro Projekt (nicht Klasse)

Empirische Studien

Wie kann Qualität (Wartbarkeit, Testbarkeit, ...) gemessen werden?

- Bewertung der Qualität durch Entwickler
- Anwendung verschiedener Metriken
- Bestimmung der Korrelation
- Kombination der Metriken (orthogonal)

Empirische Studien

Maintainability Index (Coleman/Oman, 1994):

$$MI_1 = 171 - 5.2 \cdot \ln(V) - 0.23 \cdot V(g') - 16.2 \cdot \ln(LOC)$$
$$MI_2 = MI_1 + 50 \cdot \sin \sqrt{2.46 \cdot perCM}$$

- V = average Halstead Volume per module
- $V(g')$ = average extended cyclomatic complexity per module
- LOC = average LOC per module
- $perCM$ = average percent of lines of comment per module

- MI_2 nur bei sinnvoller Kommentierung
- $MI < 65 \Rightarrow$ schlechte / $MI \geq 85 \Rightarrow$ gute Wartbarkeit

Empirische Studien

Maintainability model (Muthanna et al., 2000):

$$SMI = 125 - 3.989 \cdot FAN - 0.954 \cdot DF - 1.123 \cdot MC$$

- *FAN*: average number of external calls from the module
- *DF*: total number of outgoing and incoming data flow for the module (0-45)
- *MC*: average McCabe for the module (0-14)

Empirische Studien – OO

Wartbarkeit korreliert mit (Dagpinar und Jahnke, 2003)

- TNOS - total number of statements
- NIM - number of instance methods
- FOUT - fan out, number of classes directly used

- **nicht** Vererbungshierarchie
- **nicht** Kohäsion
- **nicht** indirekte Kopplung
- **nicht** Kopplung über used-by Beziehungen

Empirische Studien – OO

Bruntink und van Deursen, 2004:

Testbarkeit korreliert vor allem mit

- LOCC - lines of code per class
- FOUT - fan out, number of classes directly used
- RFC - response for class

Weitere Anwendungen

- Hinweise auf Designfehler
- Klonerkennung (Mayrand)
- Mustersuche (Reduzierung des Suchraums)
- Suche nach Aspekten (FIN)
- ...

Tools

Freie Tools:

Tool	Sprachen	Metriken
clc (Perl)	C/C++	LOC, Comments, #Statements
cccc	C++, Java	LOC, McCabe, OO, ...
Metrics	C	LOC, Comments, Halstead, McCabe
MAS-C4	C	LOC, Comments, Halstead, McCabe, Nesting, Fan-In/-Out, Data Flow, ...
JMT	Java	OO-Metriken
Eclipse Plugins	Java	LOC, McCabe, OO, ...

Kommerzielle: z.B. McCabeQA, CMT, TAU/Logiscope, SDMetrics, CodeCheck, Krakatau Metrics, RSM, Together, ...

Literaturreferenzen

- 1 N. Fenton, S. Pfleeger. Software Metrics: A Rigorous & Practical Approach (2nd ed.) International Thomson Computer Press, London, 1996.