

Software Engineering

Code Metriken

Prof. Dr. Peter Jüttner

Hochschule Deggendorf

Inhalte

- 5.3.1. Überblick über Programmiersprachen (10 Folien)
- 5.3.2. Codegenerierung aus UML (19 Folien)
- 5.3.3. Codierungsregeln (6 Folien)
- 5.3.4. Exkurs: MISRA (15 Folien)
- 5.3.5. Exkurs: C++ vs. C#
- 5.3.6. Code Metriken

Code Metriken



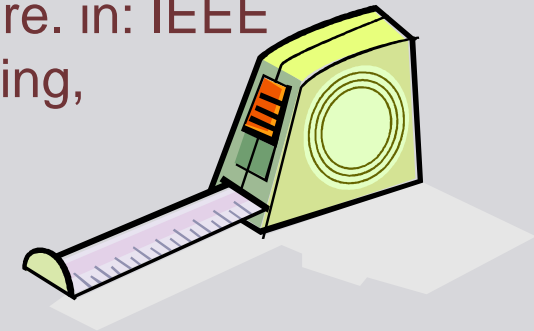
Georg E. Thaller: Software-Metriken einsetzen - bewerten - messen. Verlag Technik, 2000, ISBN 3341012605



Maurice Howard Halstead: Elements of software science. Elsevier, New York u.a. 1977, ISBN 0-444-00205-7 (Operating and programming systems series; 2).



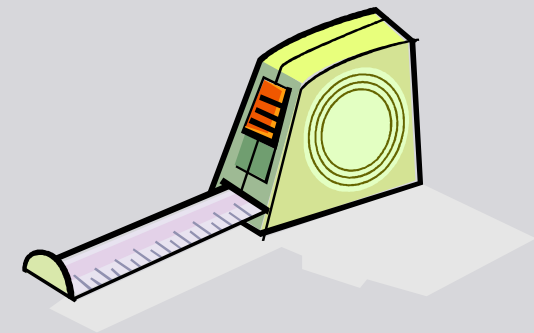
T. J. McCabe: A Complexity Measure. in: IEEE Transactions on Software Engineering, Band SE-2, 308-320. 1976.



Code Metriken

Ziele

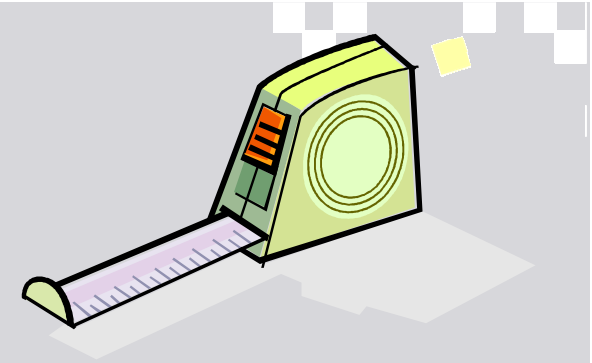
- Messen von Kenngrößen des Source Codes
 - ➔ Rückschlüsse ziehen auf bestimmte Eigenschaften des Codes, z.B.
 - ➔ Komplexität
 - ➔ Lesbarkeit
 - ➔ Testbarkeit



Code Metriken

Lines of Code

- Brutto = mit Kommentarzeilen
 - Netto = ohne Kommentarzeilen
 - relativ einfach zu messen
 - ignoriert Komplexität der Anweisungen und Strukturen
 - abhängig vom persönlichen Stil
- ➔ begrenzte Aussagekraft



```
1  if (spannung < MIN_SPANNUNG)
2    && (strom < MIN_STROM)
3    { unterspannung = TRUE;
4      ....
```

Code Metriken

Halstead (1977)

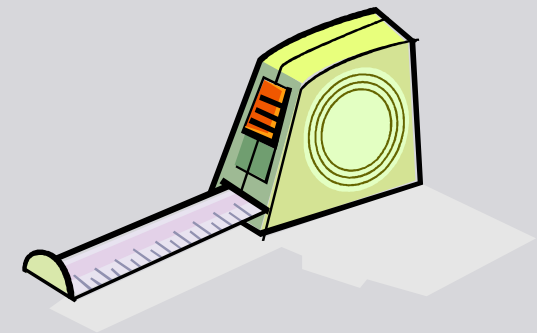
Vokabular des Programms $\mu = \mu_1 + \mu_2$ mit

μ_1 = Anzahl unterschiedlicher Operatoren (if, while, +, -, *, ...)

μ_2 = Anzahl unterschiedlicher Operanden (Variable, Konstante, Funktionen, ...)

Volumen (Umfang) des Programms $V = N * \log_2 \mu$ mit

- Länge des Programms $N = N_1 + N_2$ mit
- N_1 = Gesamtzahl verwendeter Operatoren
- N_2 = Gesamtzahl verwendeter Operanden)



Code Metriken

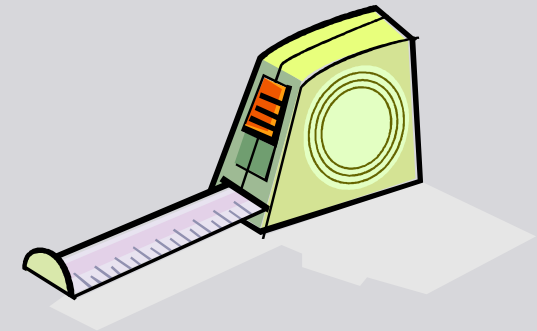
M.H. Halstead (1977)

$D = (\mu_1 / 2) * (N_2 / \mu_2)$ Schwierigkeit (Difficulty) ein Programm zu verstehen

$E = D * V$ Aufwand (Effort) das Programm zu verstehen

Bewertung

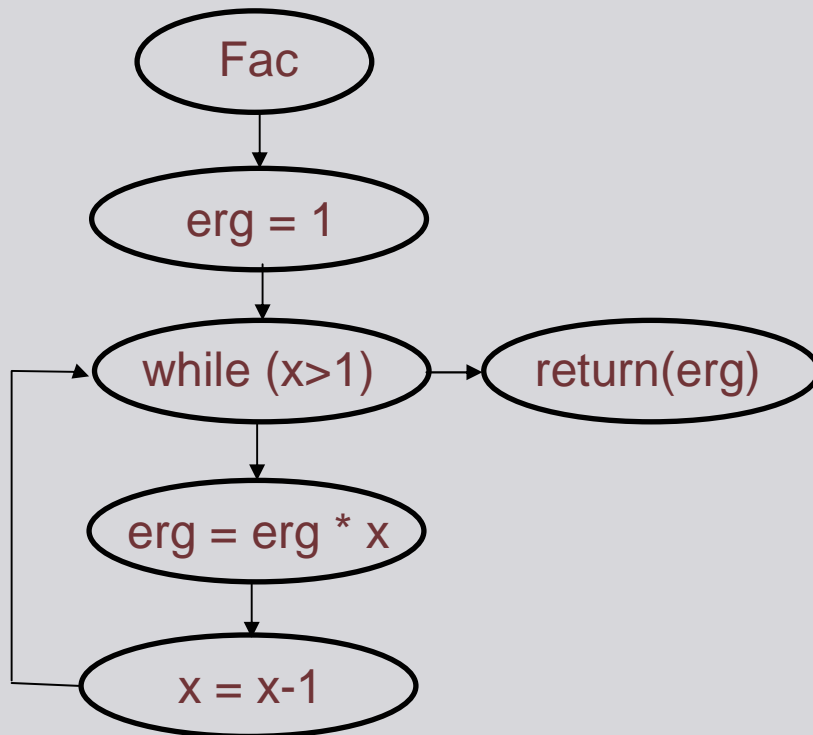
- + einfach zu berechnen
- + komplexe Ausdrücke und viele Variablen berücksichtigt
- + für alle Programmiersprachen anwendbar
- Ablaufstrukturen nicht berücksichtigt
- Namensräume, Sichtbarkeit nicht berücksichtigt



Code Metriken

Mc Cabe (1976)

- Basiert auf Kontrollflussgraph G
- Berücksichtigt Komplexität



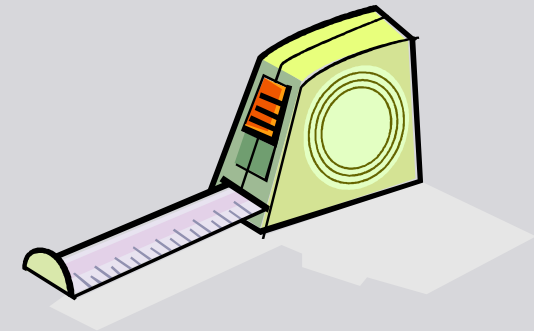
$V(G) = e - n + 2$ mit
 e = Anzahl der Kanten von G
 n = Anzahl der Knoten von G
Hier $V(G) = 2$

Code Metriken

Mc Cabe (1976)

Bewertung

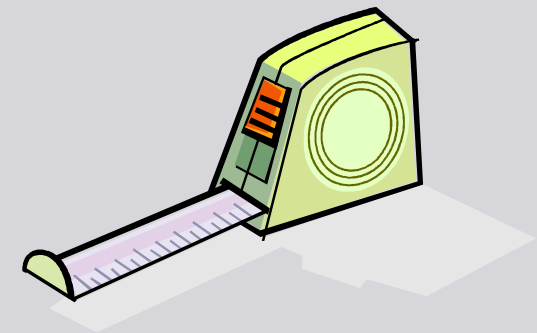
- + einfach zu berechnen
- + gibt Indikation für White Box Testaufwand (C1 Überdeckung)
- Komplexität der Anweisungen bleibt unberücksichtigt
- Schlecht geeignet für Objektorientierte Programmierung
 - kurze Methoden
 - Komplexität liegt im Zusammenspiel von Klassen bzw. Objekten



Code Metriken

Objekt – orientierte Metriken (Beispiele)

- DIT (Depth of Inheritance Tree) — Anzahl Oberklassen einer Klasse
- NOC (Number of Children of a Class) — Anzahl direkter Unterklassen
- RFC (Response For a Class) — Anzahl der Methoden, die potentiell ausgeführt werden können, wenn ein Objekt der betrachteten Klasse auf eine eingegangene Nachricht reagiert
- WMC (Weighed Methods per Class) — Anzahl definierter Methoden
- CRO (Coupling Between Object Classes) — Anzahl Klassen, auf deren Dienste eine Klasse zugreift



Zum Schluß dieses Abschnitts ...

Noch Fragen ??