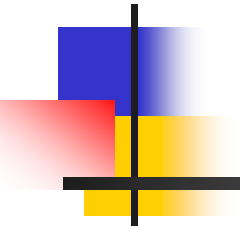


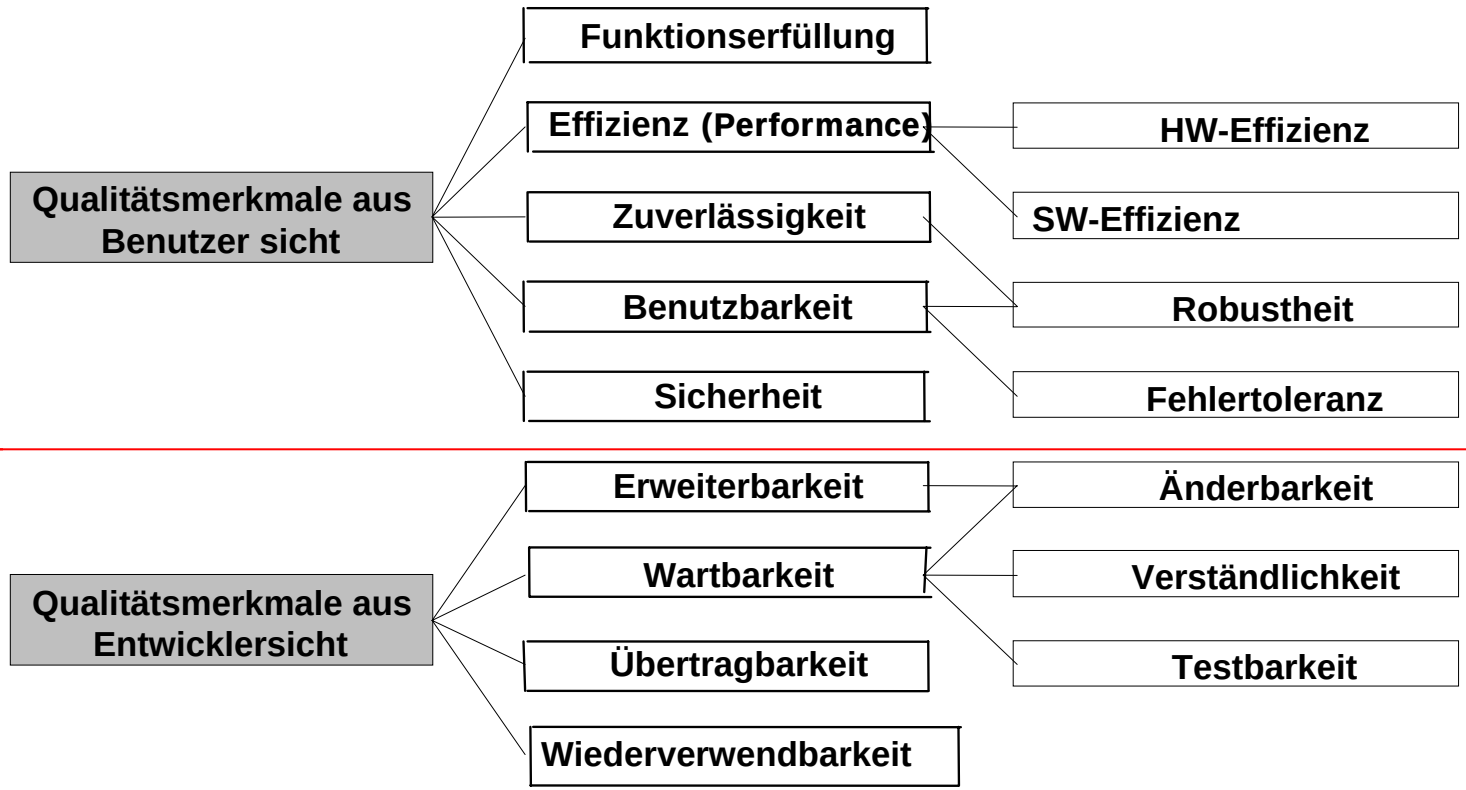
Methoden zur Überprüfung der Ergebnisse
10. Metriken

von
Prof. Dr. Wolfgang Weber

[BGKRSW05]. meine Homepage

Was will man Messen

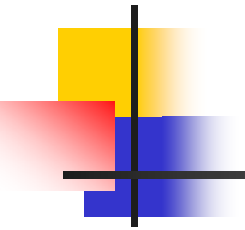
Softwarequalitätseigenschaften:



SW-Metrikem

Man redet oft von Qualität

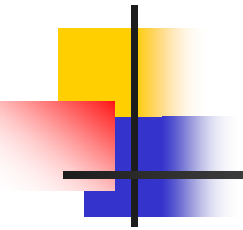
- aber: Wie kann man feststellen, wie hoch die Qualität eines SW-Produktes ist?
  - -> durch Messen
- aber: Wie kann ich SW-Qualitätseigenschaften wie z. B. Verständlichkeit messen?
  - -> durch den Einsatz von SW-Metriken



Metriken

- Leiten aus Code Programmeigenschaften ab

  wie z. B.
  - Programmgröße
  - Schwierigkeit der Implementierung
  - Aufwand für die Implementierung und spätere Wartung etc.
- Durch Kombination von elementaren Programmeigenschaften kann ich Messzahlen für SW-Qualitätseigenschaften wie Verständlichkeit, Wartbarkeit etc. ableiten.



Metriken

Frage:

Wie messe ich Größe eines Programms (einer Funktion)?

- Anzahl Zeilen (LOC)
  - -> Durch viele Anweisungen auf einer Zeile wird Programm nicht kleiner!
- Anzahl Anweisungen
  - -> Durch Zusammenfassung von Einzelanweisungen zu komplizierteren Anweisungen wird ein Programm nicht besser!
- Anzahl Bytes
  - Durch Sparen von Code (z. B. kurze Variablennamen) wird Programm nicht besser!

Besseres Maß: Halstead-Metriken

Halstead-Metriken

- Anzahl der unterschiedlichen Operatoren: n1
- Anzahl der unterschiedlichen Operanden: n2
- Anzahl des Auftretens von Operatoren: N1
- Anzahl des Auftretens von Operanden: N2

Primitives Beispiel zur Veranschaulichung der Maße:

```
void tausch (float &x, float &y)
{
      float z;
      z=x;
      x=y;
      y=z;
}tausch
```

Operatoren	Anz. Auftreten	Operanden	Anz. Auftreten
void	1	x	3
( )	1	y	3
&	2	z	3
float	3		
=	3		
{ }	1		
,	1		
;	4		
n1 = 8	N1 = 16	n2 = 3	N2 = 9

Halstead-Metriken

Abgeleitete Maße:

- Größe des Vokabulars: n = n1 + n2, bei unserem Beispiel ?
  - 8+3 = 11
- Länge der Implementierung: N = N1 + N2, bei unserem Beispiel?
  - 16 + 9 = 25
- Berechnete Länge: **N^** = n1*ld(n1) + n2*ld(n2)
  (Es gilt empirisch: N^ ist etwa gleich N), bei unserem Beispiel?
  N^ = 8*ld(8) + 3*ld(3) = 24 + 4.75 = 28.75
  also etwa N^ = N.
- Programmgröße (Volumen): **V** = N*ld(n), bei unserem Beispiel?
  V = 25*ld(11) = 25 * 3,5 = 87,5
- ...

Mc-Cabe-Maß

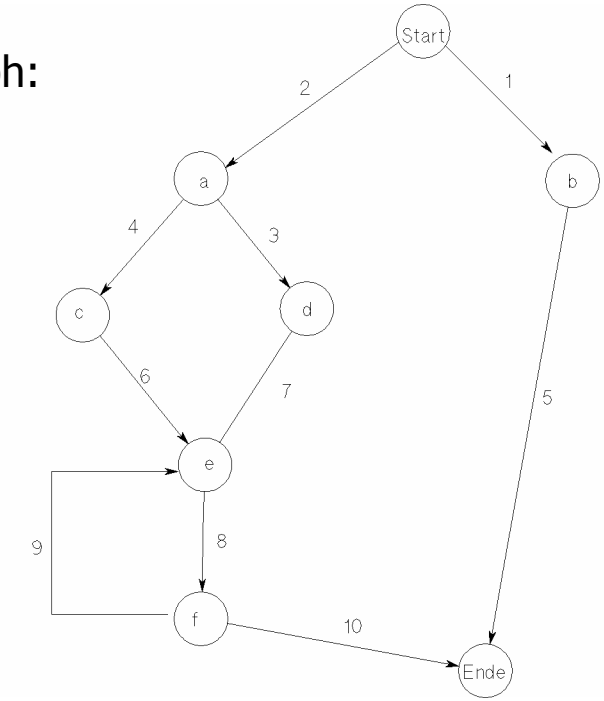
- Gibt Auskunft über Komplexität der Kontrollstruktur einer Funktion

zyklomatische Zahl (McCabe-Zahl) z(G):

1. z(G) = V + 1,
   V: Anzahl der Verzweigungen über boolschen Bedingungen im Programm-Graphen G
2. z(G) = e – n + 2,
   e: Anzahl Kanten von G,
   n: Anzahl Knoten von G
3. z(G) = P (Originalsdefinition),
   P: Zahl der linearunabhängigen Pfade durch den Programm-Graphen G

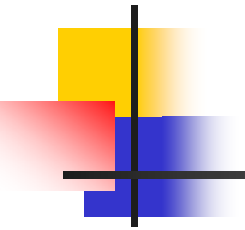
Mc-Cabe-Maß

Programm-Graph:



1. z(G) = V + 1 = 3 + 1 = 4
2. z(G) = e – n + 2 = 10 – 8 + 2 = 4
3. z(G) = 4 (1.<2,3,7,8,10>, 2.<2,4,6,8,10>, 3.<2,4,6,8,9,8,10>, 4.<1.5>)

   <2,3,7,8,9,8,10> ist linear abhängig (= 3. – 2. + 1.)



Mc-Cabe-Maß

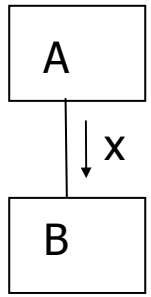
Empfehlung:

- Komplizierte Programme: zerlegen in Teilprogramme, wobei für jedes Programm möglichst z(G) < 10

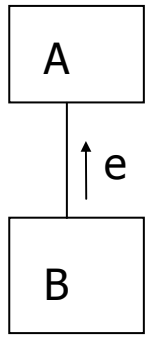
Maße für Informationsfluss

Darstellung des Flusses: (Quelle, Fluß, Ziel)

- Lokaler Datenfluß (A, x, B)
  A ruft die Funktion B auf und übergibt Wertparameter x

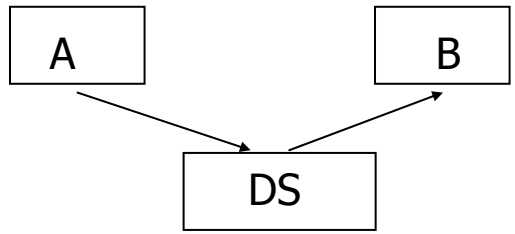


- Lokaler Datenfluß (B, e, A)
  A ruft die Funktion B auf und erhält Ergebnis e



Maße für Informationsfluss

- Globaler Datenfluß (A, DS, B)
  A verändert
  eine globale Datenstruktur DS
  und B benutzt diese



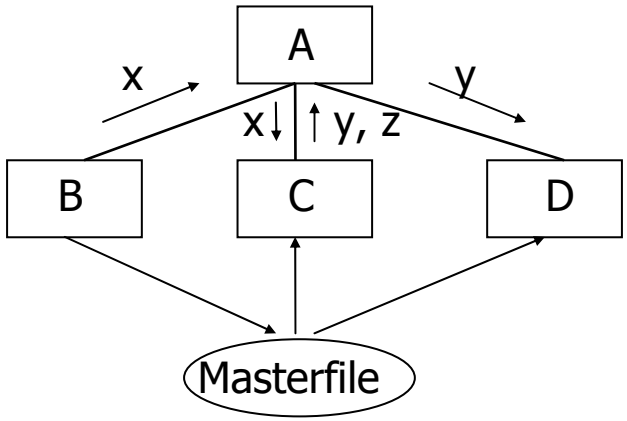
Maße für Informationsfluss

Beurteilung der Software-Architektur durch Informationsfluß-Analyse:

- "**fan-in**" Zahl der Datenflüsse, die zu einem Modul m führen: FI_m
- "**fan-out**" Zahl der Datenflüsse, die von einem Modul wegführen: FO_m
- $IF4_m(S) = (FI_m * FO_m)^2$ Maßzahl für den Modul m im System S
- $IF4(S) = \sum_{i=1}^{n} (FI_i * FO_i)^2$ Maßzahl für die Architektur des Systems S

Maße für Informationsfluss

Beispiel:



Globale Datenflüsse:
(B, Masterfile, C)
(B, Masterfile, D)

- Frage: fan-in, fan-out und IF4 der Module und IF4(S) des Gesamtsystems?

Modul	fan-in (FI_i)	fan-out (FO_i)	$FI_i * FO_i$	$IF4_i = (FI_i * FO_i)^2$
A	3	2	6	36
B	0	2	0	0
C	2	2	4	16
D	2	0	0	0

IF4(S): 52

Maße für Informationsfluss

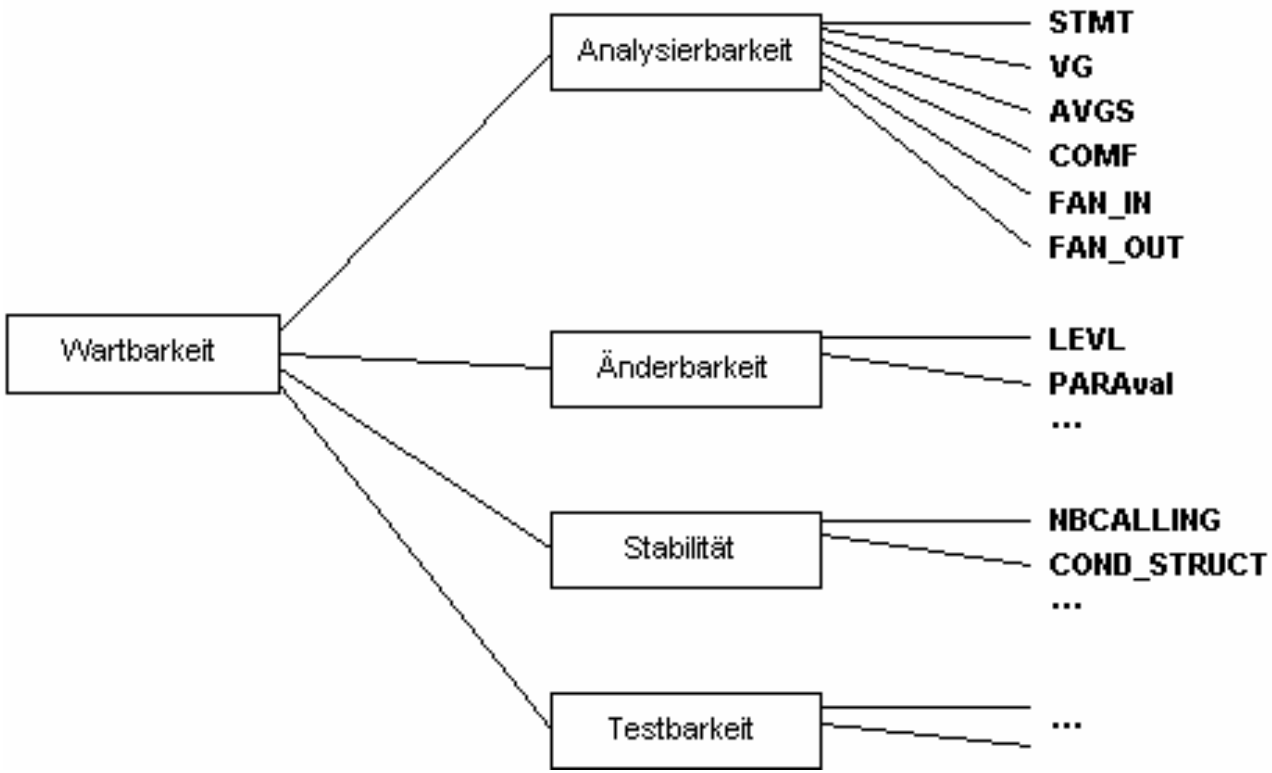
- $IF4_i$ drückt die Kopplung der Module mit ihrer Umwelt aus.
- Geringe Kopplung = gutes Design
- $IF4_D = IF4_B < IF4_C < IF4_A$
- A hat die stärkste **Kopplung** mit der Umgebung.
- IF4 kann benutzt werden, um Architekturen miteinander zu vergleichen.

  Deutlich geringeres IF4 -> weniger starke Kopplung unter den Komponenten.
- Beim "Altern" der Software entartet sehr oft die Struktur.

  IF4 und die verschiedenen IF4i können aufzeigen, ob und wo ein Re-Engineering notwendig bzw. sinnvoll wäre.
- Empirische Studien haben gezeigt: Es besteht eine starke Abhängigkeit zwischen Informationsfluss-Maßen und dem Wartungsaufwand.
- Es gibt aber auch Abweichungen von dieser empirischen Gesetzmäßigkeit, z.B. verursacht durch ungewöhnlich große Moduln.
- Eine wichtige Frage: Welches ist die optimale Größe von Moduln?
- Zu diesem Zweck können kombinierte Maße wie (LOC_i , $IF4_i$) oder ($McCabe_i$,$IF4_i$) benutzt werden, um ausgeartete Module zu lokalisieren.

Komplexe zusammengesetzte Qualitätsmaße

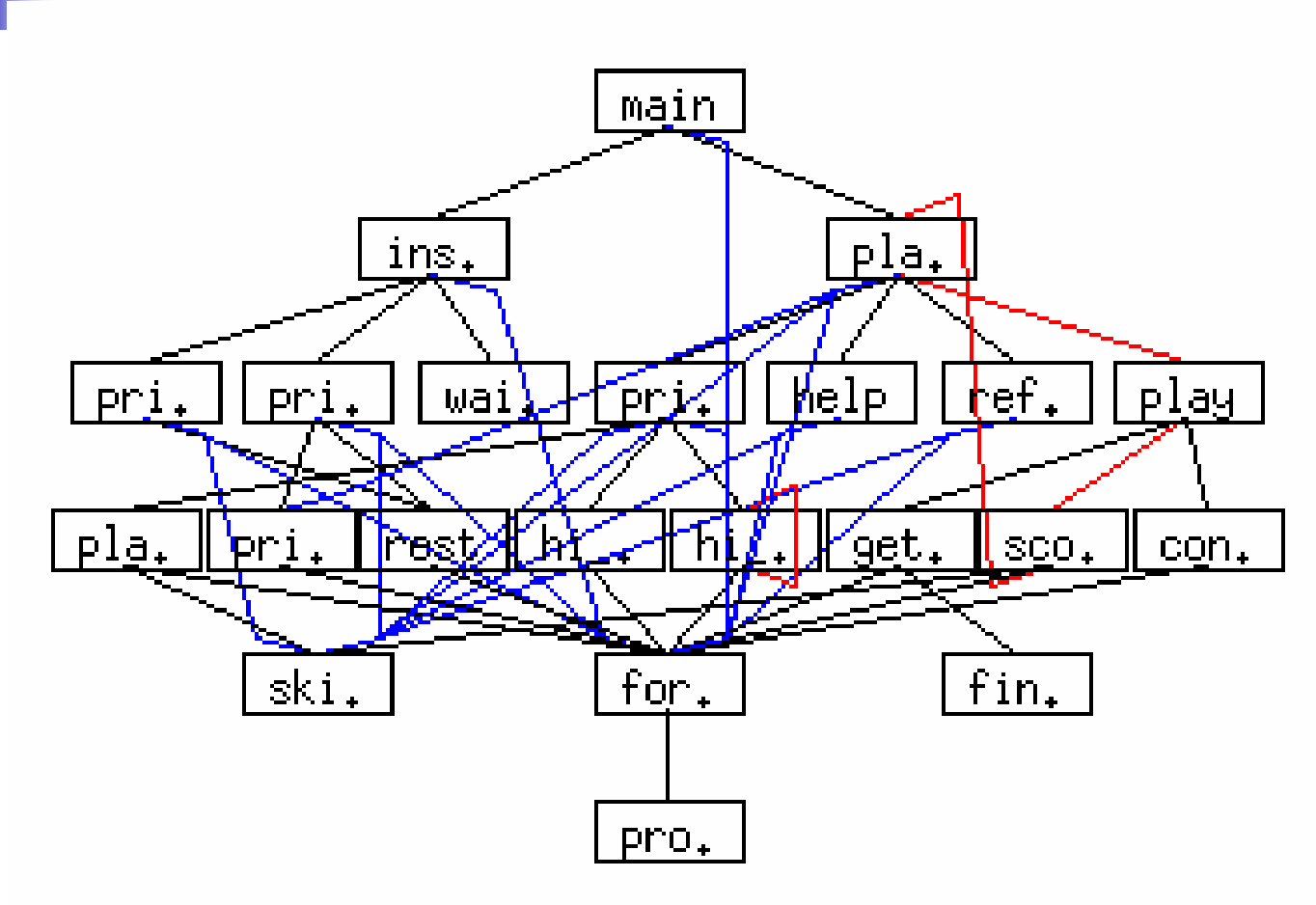
- Teil des Qualitätsmodells des Testtools Logiscope der Fa. Verilog:



Komplexe zusammengesetzte Qualitätsmaße

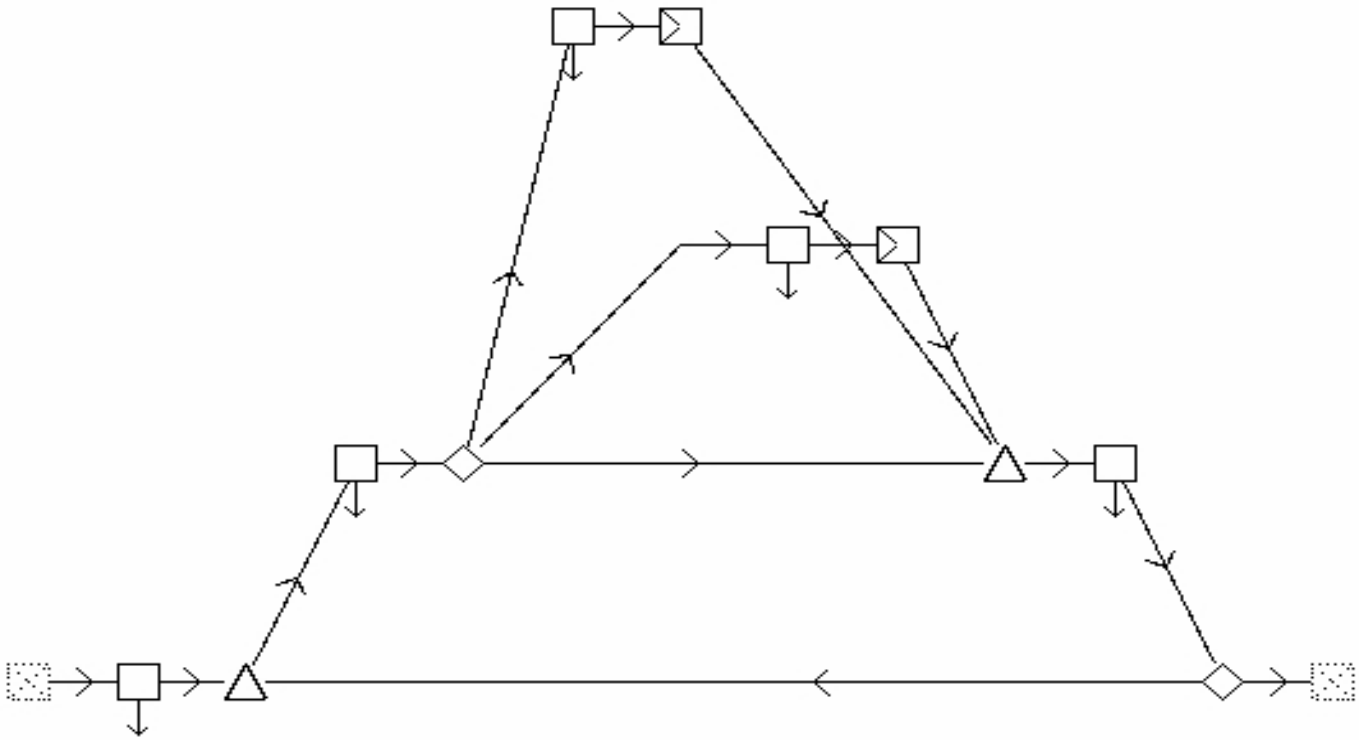
STMT	**Anzahl der Statements einer Komponente.** Je größer die Anzahl der Statements in einem Programm, um so schwerer ist es zu verstehen, d. h. um so größer ist der Aufwand zur Fehlersuche oder bei Änderungen in der Wartung. Es ist deswegen sinnvoll die Anzahl Statements in jeder Funktion eines Programms zu beschränken.
VG	**zyklomatisches Maß nach McCabe**
AVGS	**durchschnittliche Länge der Statements.** Dieses Maß ist ein komplexes Maß und wird nach folgender Gleichung berechnet: AVGS=(N1+N2+1) / (STMT+1) wobei N1 = Gesamtanzahl der Operatoren (Halstead) N2 = Gesamtanzahl der Operanden. Je länger eine Anweisung ist, um so größer ist der Aufwand sie zu verstehen.
COMF	**Kommentarfrequenz.** Das ist das Verhältnis von Kommentaren und Anweisungen.
FAN_IN	**Zahl der Datenflüsse, die zu einem Modul führen**. Je größer diese Zahl für einen Modul ist, um so stärker wird der Modul durch seine Umgebung (die rufenden Module) beeinflußt.
FAN_OUT	**Zahl der Datenflüsse, die von einem Modul wegführen.** Je größer diese Zahl für einen Modul ist, um so größer ist der Einfluß dieses Moduls auf seine Umgebung. Dieser Modul spielt also eine sehr wichtige Rolle in einem System und muß besonders genau überprüft werden.
	Aus diesen Metriken wird ein Wert für die Analysierbarkeit berechnet. **Analysierbarkeit** = f1*STMT + f2*VG + f3*AVGS + f4*COMF + f5*FAN_IN + f6*FAN_OUT

Darstellung der Messergebnisse



Aufrufgraph (heraus-zoomen möglich!)

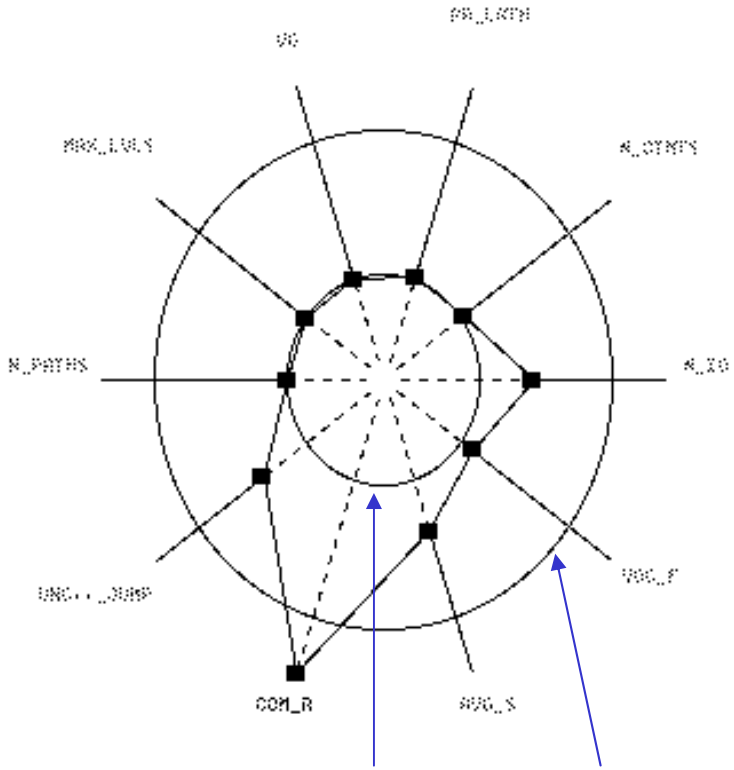
Darstellung der Messergebnisse



Kontrollflussgraph (gleiche Information wie PAP, Aktivitätsdiagramm)

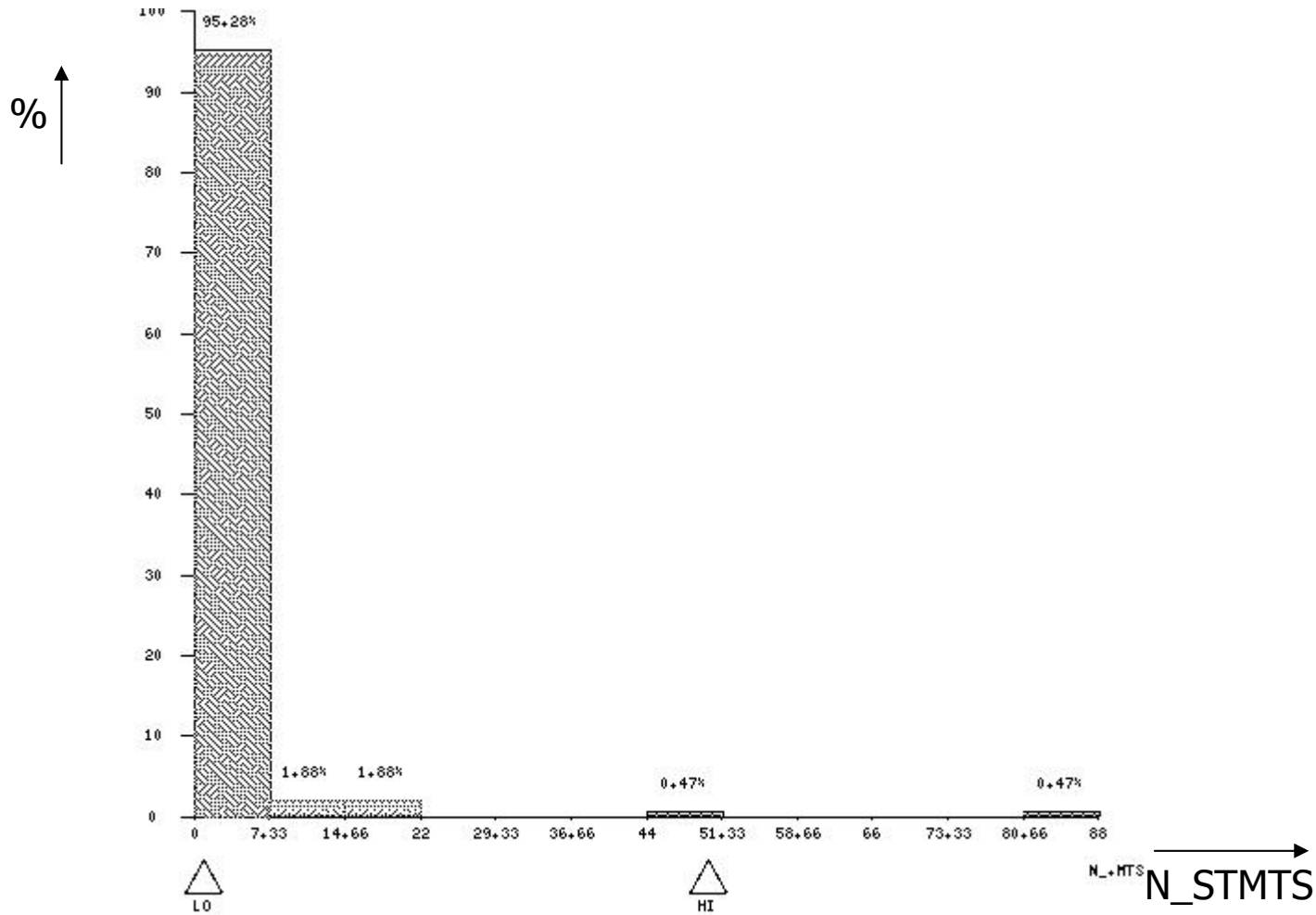
Darstellung der Messergebnisse

METRIC	LO	HI	VALUE
N_STMTS	1	50	2
PR_LGTH	3	350	9
VG	1	15	1
MAX_LVLS	1	5	1
N_PATHS	1	80	1
UNCOND_J	0	0	0
COM_R	0.20	1.00	1.50
AVG_S	3.00	7.00	4.50
VOC_F	1.00	4.00	1.28
N_IO	2	2	2



Kiviatdiagramm mit Minimum- und Maximum-Kreis

Darstellung der Messergebnisse



Statistikdiagramm