Vorlesung (WS 2014/15)

Softwarekonstruktion

Prof. Dr. Jan Jürjens

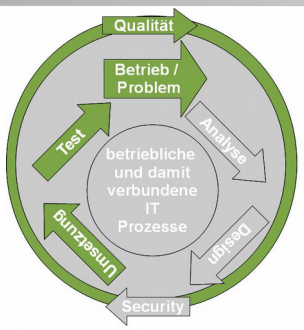
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 2.2: Softwaremetriken

v. 08.12.2014

Einordnung
Softwaremetriken

- Modellgetriebene SW-Entwicklung
- Qualitätsmanagement
- Testen
  - Grundlagen Softwareverifikation
  - Softwaremetriken
  - Black-Box-Test
  - White-Box-Test
  - Testen im Softwarelebenszyklus



[inkl. Beiträge von Prof. Martin Glinz, Universität Zürich und Prof. Ian Sommerville, Univ. St. Andrews]

Literatur (s. Vorlesungswebseite)**:**
• Andreas Spillner, Tilo Linz: Basiswissen Softwaretest.
• Eike Riedemann: Testmethoden für sequentielle und nebenläufige Software-Systeme.

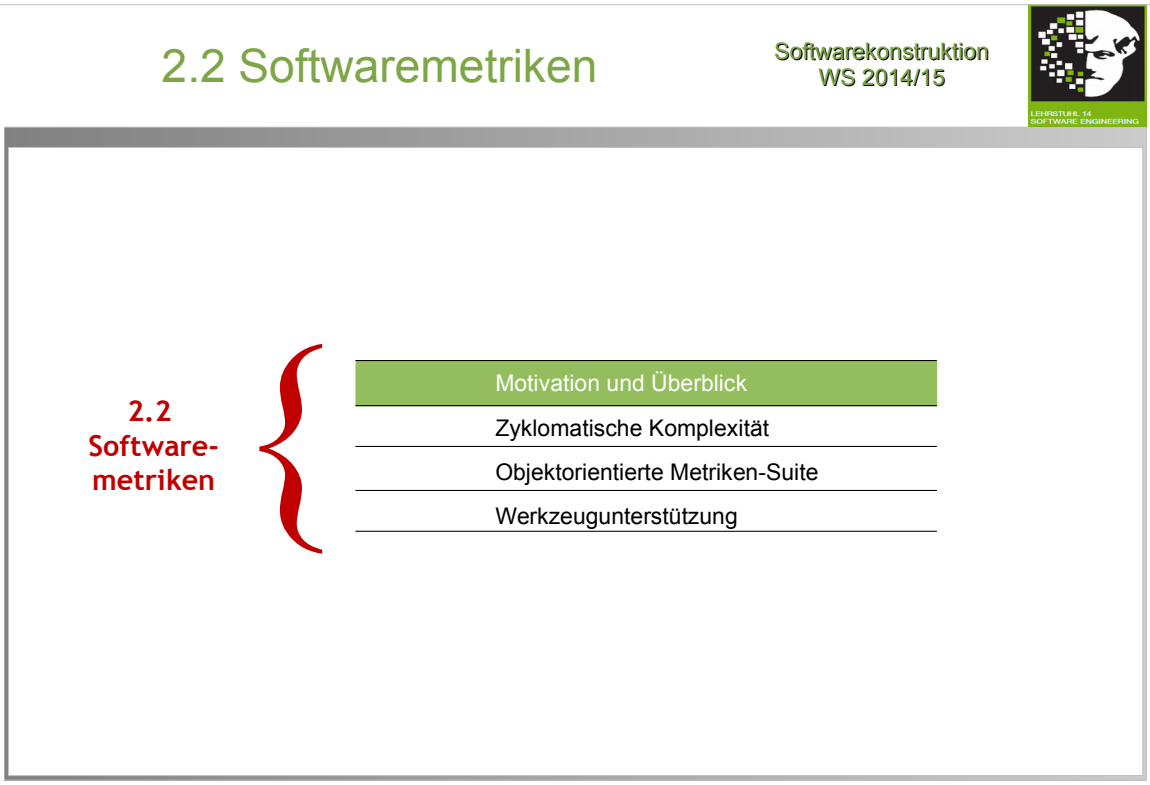
Literatur:

E. Riedemann: **Testmethoden für sequentielle und nebenläufige Software-Systeme**
http://www.ub.tu-dortmund.de/katalog/titel/687299

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

Einleitung Softwaremetriken

- **Vorheriger Abschnitt:** Einführung Softwareverifikation
- **Dieser Abschnitt:** Bestimmung der Komplexität des Programmes
  - Softwaremetriken: Zyklomatische Zahl
  - Werkzeugunterstützung



Literatur:

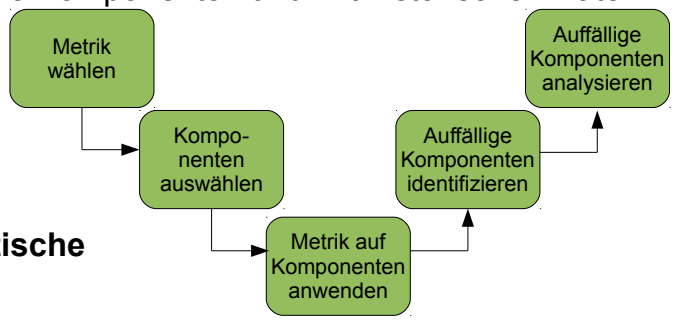
V. Gruhn: **MDA - Effektives Software-Engineering**

http://www.ub.tu-dortmund.de/katalog/titel/1223129

- Kapitel 2

Softwareverifikation vs. Softwaremetriken

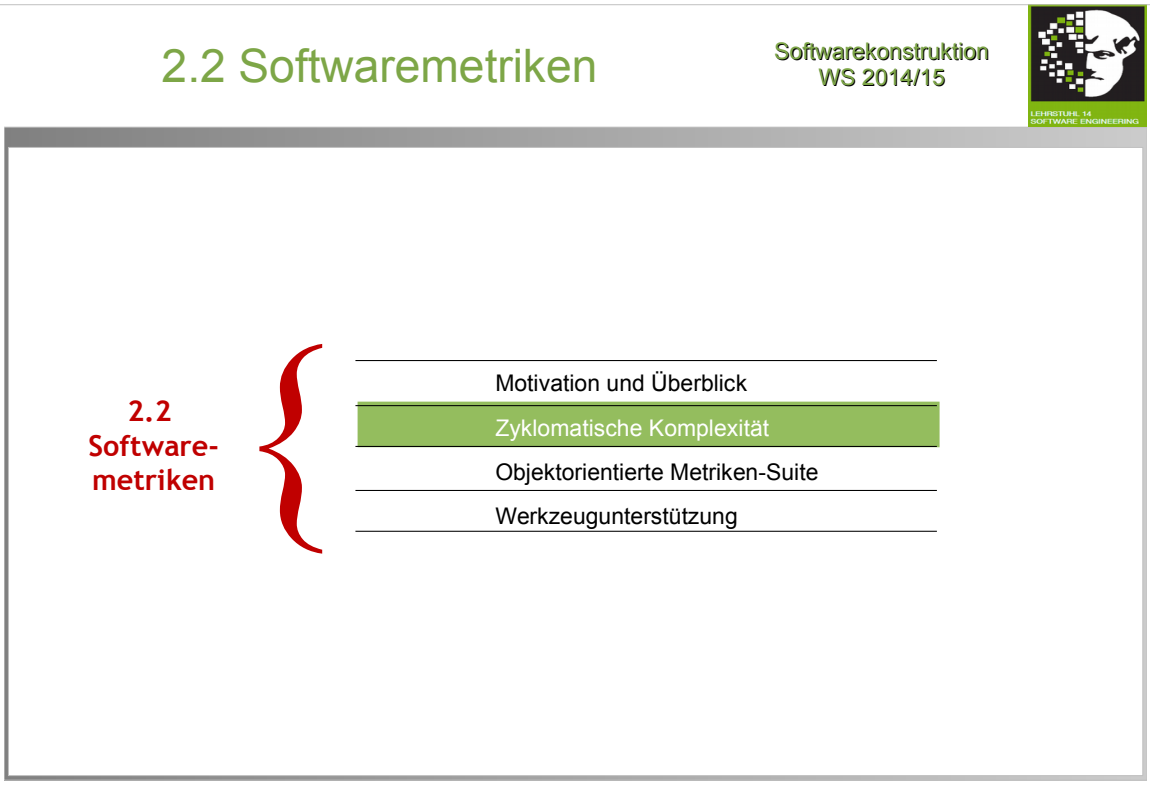
- Volllständiges Austesten i.A. unmöglich => **Prioritäten** setzen.
- Schwerpunkt auf besonders **fehleranfällige** Teile der Software legen !
- Wie diese Teile effizient identifizieren ?
- Fehleranfälligkeit korreliert mit **Komplexität** von Softwarekomponente.
- Komplexität der Softwarekomponenten mit **Metriken** ermitteln.
- Metrikwerte für verschiedene Komponenten und mit historischen Daten **vergleichen**.
- Anomale Werte können auf Qualitätsprobleme hinweisen
  => intensiver testen.
- Hier ein Beispiel: **zyklomatische Komplexität**.



Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11 (S.478-480)



Literatur:

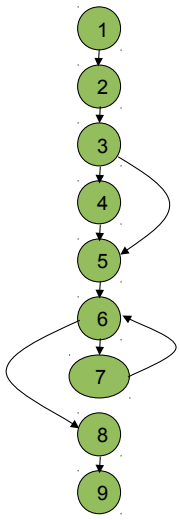
V. Gruhn: **MDA - Effektives Software-Engineering**

http://www.ub.tu-dortmund.de/katalog/titel/1223129

- Kapitel 2

Zyklomatische Komplexität eines Programmes

- **Zyklomatische Komplexität** („McCabe-Metrik“):
  Misst Komplexität eines Programmes auf Basis des Kontrollflussgraphen.
- **Zyklomatische Komplexität** des Kontrollflussgraphen G: $\nu(G) = e - v + 2$
- **Beispiel:** $\nu(G) = ?$



Literatur:

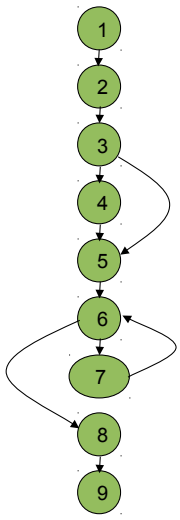
H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

T.J. McCabe: A Complexity Measure, IEEE Transactions on Software Engineering Vol. 2, No. 4, p. 308 (1976)

Zyklomatische Komplexität eines Programmes

- **Zyklomatische Komplexität** („McCabe-Metrik“):
  Misst Komplexität eines Programmes auf Basis des Kontrollflussgraphen.
- **Zyklomatische Komplexität** des Kontrollflussgraphen G: $\nu(G) = e - v + 2$
- **Beispiel:** $\nu(G) = 10 - 9 + 2 = 3$.



Literatur:

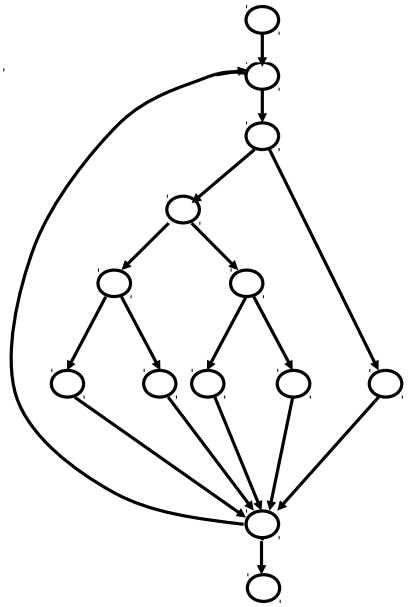
H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

T.J. McCabe: A Complexity Measure, IEEE Transactions on Software Engineering Vol. 2, No. 4, p. 308 (1976)

Beispiel: Zyklomatische Komplexität

- **Bemerkung:** Berechnung von ν(G) in Programmiersprachen mit geschlossenen Ablaufkonstrukten:
  - Zähle alle Verzweigungen und Schleifen (**if**, **while**, **for**, etc.).
  - Addiere für jede Auswahl-Anweisung (**switch**, **CASE**) die Zahl der Fälle - 1.
  - Addiere 1.
- **Hier:** ? + 1
- **Entspricht McCabe-Formel:**
  v = ? , e = ?
  ν(G) = e - v + 2 = ?



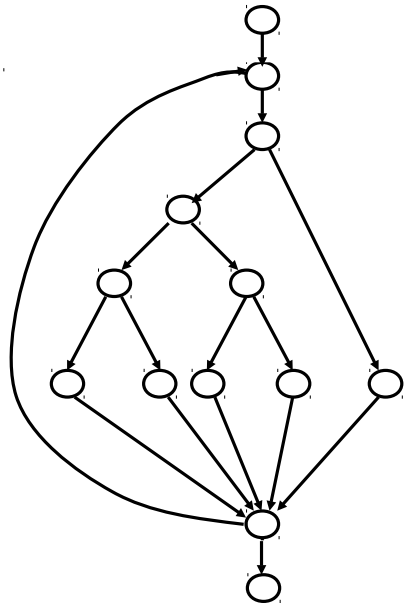
Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Beispiel: Zyklomatische Komplexität

- **Bemerkung:** Berechnung von ν(G) in Programmiersprachen mit geschlossenen Ablaufkonstrukten:
  - Zähle alle Verzweigungen und Schleifen (**if**, **while**, **for**, etc.).
  - Addiere für jede Auswahl-Anweisung (**switch**, **CASE**) die Zahl der Fälle - 1.
  - Addiere 1.
- **Hier:** 5 + 1.
- **Entspricht McCabe-Formel:**
  v = 13, e = 17
  ν(G) = e - v + 2 = 17 - 13 + 2 = 6



Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Mehrere Endpunkte

Bislang Annahme: Programm hat nur ein Endpunkt.

Zyklomatische Komplexität des Kontrollflussgraphen G eines Programms **mit mehreren Endpunkten**:
$\nu(G) = e - n + p + 1$

- e Zahl der Kanten
- n Zahl der Knoten
- p Zahl der Endpunkte des Programms

[NB: Annahme weiterhin: Nur ein Startpunkt.]

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Bewertung der zyklomatischen Komplexität (1)

Zyklomatische Komplexität **> 10** nach McCabe **nicht tolerabel.**

- Überarbeitung des Programmteils !
- Messwert 6 im Beispiel liegt im Bereich → nach McCabe akzeptabel
- Z.T. Messwerte bis 15 ausnahmsweise akzeptiert (dokumentierte Begründung).

Für Wartbarkeit: Verständlichkeit eines Programmstücks wichtig.

- Ermittelte **zyklomatische Komplexität hoch:**

  → Nachvollziehen des Ablaufs des Programmstücks schwierig.

  → **Schlechte Verständlichkeit.**

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Bewertung der zyklomatischen Komplexität (2)

Auskunft über Testaufwand:

- Zykl. Komplexität = **Anzahl unabhängiger Pfade**.
- Zykl. Komplexität – 1 = **Anzahl Entscheidungen** im Kontrollflussgraph.
- **100%-ige Ausführung aller Anweisungen** und Verzweigungsmöglichkeiten eines Programms verlangt.
  → Einmaliger Durchlauf unabhängiger Pfade durch Kontrollfluss-graphen.
- Zykl. Komplexität: **Obere Grenze für Anzahl benötigter Testfälle** zur Erreichung dieses Kriteriums.

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Bewertung der zyklomatischen Komplexität (3)

- Problem der **Validität:**
  - Komplexität Spaghetti-Programm = Komplexität wohlstrukturiertes Programm gleichen Problems.
  - Auch intuitiv gleich komplex ?
- Eignet sich Maß als **Indikator für Fehleranfälligkeit ?**

  → Nicht besser als NCSS (Kafura und Canning, 1985).
- Skala: **Verhältnisskala**, aber **nicht additiv:**

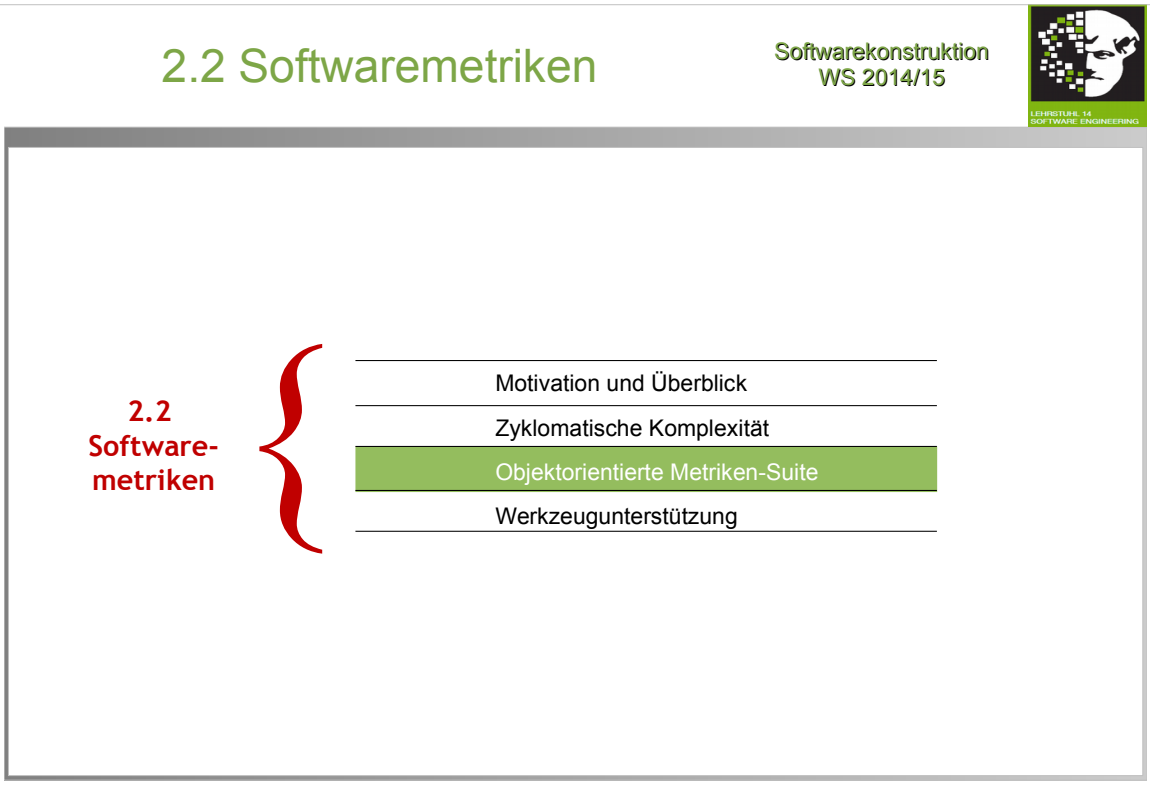
  Aneinanderreihung zweier Programmstücke mit Komplexitäten ν1 und ν2: Komplexität ν1+ν2-1

  → **Kontraintuitive Eigenschaft.**

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)



Literatur:

V. Gruhn: **MDA - Effektives Software-Engineering**

http://www.ub.tu-dortmund.de/katalog/titel/1223129

- Kapitel 2

Die objektorientierte Metriken-Suite „CK“ (1)

Softwaremetrik	Beschreibung
Gewichtete Methoden pro Klasse (Weighted Method Complexity, WMC)	**Anzahl Methoden in jeder Klasse**, gewichtet durch Komplexität: WMC = $\Sigma C(i)$ mit C(i) = Komplexität von Methode i **Komplexe Objekte:** Verständnis schwer.
Tiefe des Vererbungs-baums (Depth of Inheritance Tree, DIT)	**Maximale Tiefe der Generalisierungshierarchie**. → Anzahl Ebenen im Vererbungsbaum. Je tiefer der Baum, desto komplexer das Design: Viele Klassen verstehen, um unterstes Blatt des Vererbungsbaums nachvoll zu ziehen.
Zahl der Kinder (Number of Children, NOC)	**Anzahl direkter Unterklassen.** • **Hoher NOC-Wert = hohe Wiederverwendung** → Validierung der Basisklassen aufwändig, wegen großer Zahl abhängiger Unterklassen.

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.3 – Metriken für objektorierntierte Komponenten (S.482-486)
- WMC (S.484)
- DIT (S.483)
- NOC (S.483)

Die objektorientierte Metriken-Suite „CK“ (2)

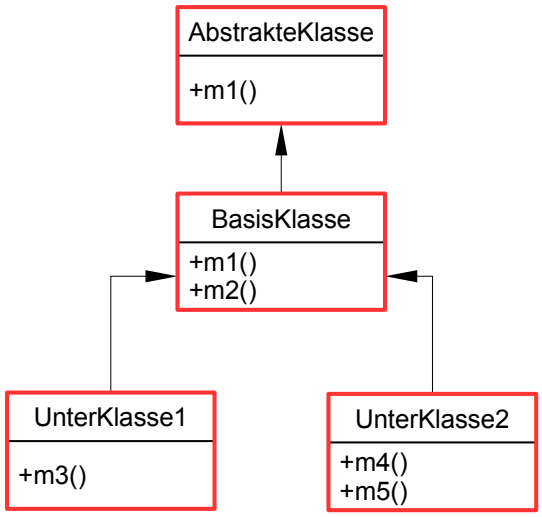
Softwaremetrik	Beschreibung
Kopplung von Klassen (CBO)	Methoden von C benutzen Methoden/Variablen von D → Klassen C und D gekoppelt. **CBO: Maß** für Anzahl der **Kopplungen.** Hoher CBO-Wert: Klassen voneinander sehr abhängig → **Änderung einer Klasse betrifft viele Klassen im Programm.**
Reaktion einer Klasse (RFC)	**Maß für Anzahl Methoden:** als **Antwort auf Nachricht** an umgebene Klasse ausführbar. Hoher RFC-Wert: Klasse komplexer und fehleranfälliger.
Mangel an Zusammenhalt in Methoden (Lack of Cohesion of Methods, LCOM)	**Anzahl** gemeinsam benutzter **Instanzvariablen von Methoden** einer Klasse. Verschiedene Arten von Metriken → Informationen zu anderen Metriken lieferbar.

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.3 – Metriken für objektorierntierte Komponenten (S.482-486)
- CBO (S.484)
- RFC (S.483)
- LCOM (S.485)

Beispiel: WMC, DIT und NOC



- AbstrakteKlasse
  - WMC = 0; DIT = 0; NOC = 1
- BasisKlasse
  - WMC = 2; DIT = 1; NOC = 2
- UnterKlasse1
  - WMC = 1; DIT = 2; NOC = 0
- UnterKlasse2
  - WMC = 2; DIT = 0; NOC = 0

(hier wird angenommen, dass jede Methode die Komplexität C = 1 hat)

Diskussionsfrage

Komplexitätsmaße: Indikator für Fehleranfälligkeit und Pflegbarkeit.
Welches **Problem** ergibt sich, wenn Programmierer am **Komplexitätsmaß gemessen** wird, um beide Eigenschaften zu steuern ?

Diskussionsfrage

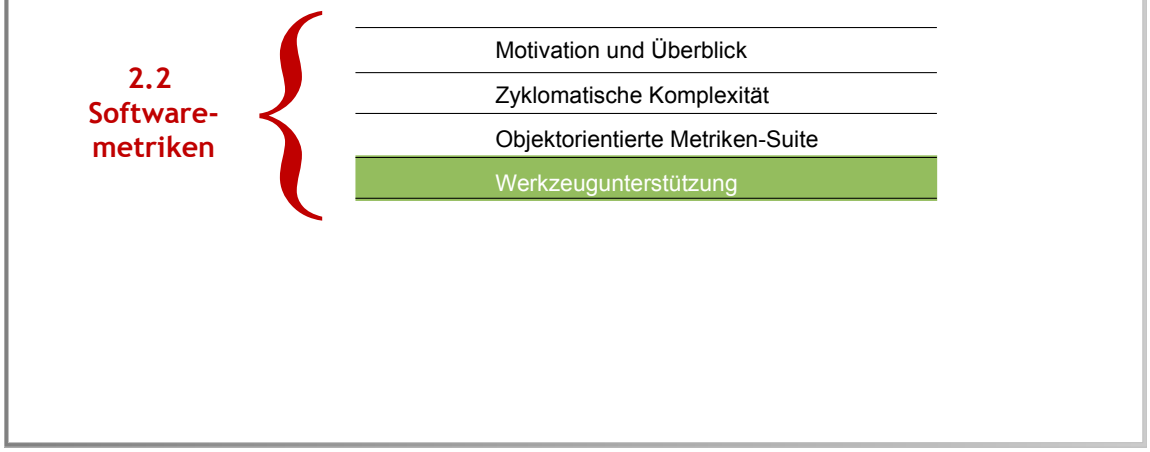
Komplexitätsmaße: Indikator für Fehleranfälligkeit und Pflegbarkeit. Welches **Problem** ergibt sich, wenn Programmierer am **Komplexitätsmaß gemessen** wird, um beide Eigenschaften zu steuern ?

Antwort:

1) Komplexität von Code = Komplexität des zu lösenden Problems
   → teilweise durch Programmierer steuerbar.

   Vergleich von **mehreren Lösungen** für das **gleiche Problem** durch Metriken → Vergleich von **Qualität** der Programmierung.

2) **Metrik bekannt:** Programmierer kann **Code optimieren,** ohne Qualität zu verbessern.



Literatur:

V. Gruhn: **MDA - Effektives Software-Engineering**

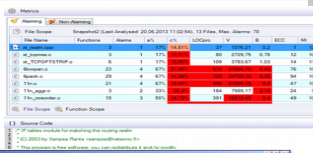
http://www.ub.tu-dortmund.de/katalog/titel/1223129

- Kapitel 2

Werkzeugunterstützung

Testwell CMTJava http://www.testwell.fi/cmtjdesc.html

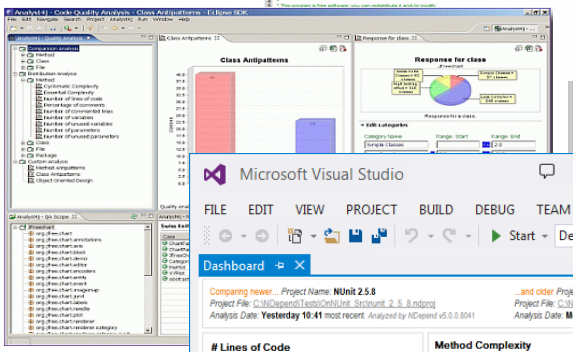
- *Unterstützte Metriken:* Zeilenmetriken, **Halstead-Metrik, McCabe Zyklomatische Komplexität,** Wartungsaufwand.



Eclipse Metrics Plugin (Freeware)

http://eclipse-metrics.sourceforge.net

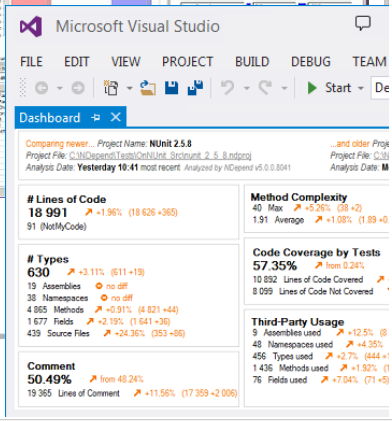
- *Unterstützte Metriken:* **McCabe Zyklomatische Komplexität.**



Ndepend (Testversion)

http://ndepend.com

- *Unterstützte Metriken:*
  - **Zyklomatische Komplexität**
  - Schätzung des **Entwicklungsaufwands**.
  - Erkennung von **großen Methoden** und **Klassen**.

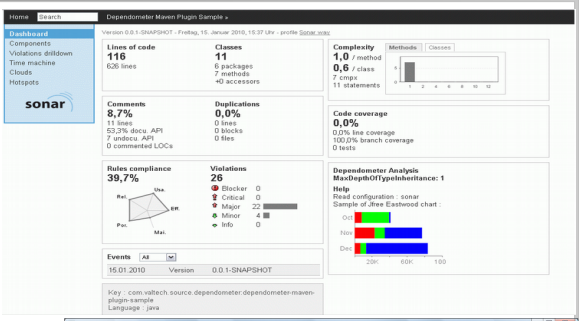


Werkzeugunterstützung

Sonar Source (Freeware)

http://sonarsource.com

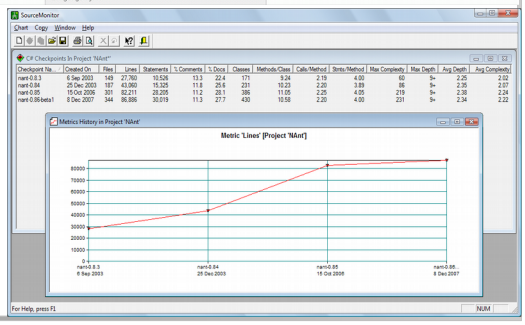
- *Unterstützte Metriken:* **McCabe Zyklomatische Komplexität**.



Source Monitor (Freeware)

http://www.campwoodsw.com/sourcemonitor.html

- *Unterstützte Metriken:* **Zyklomatische Komplexität.**



Zusammenfassung

In diesem Abschnitt:

- Motivation Softwaremetriken
- Zyklomatische Komplexität
- Werkzeugunterstützung

Im nächsten Abschnitt:

- Dynamisches Testen

Anhang
(weitere Informationen)

Zyklomatische Zahl eines zyklischen Graphen

- **Zyklischer Graph:** Auffassung als Vektorraum.
  [via Menge seiner Eulerschen Teilgraphen]
  → **Menge unabhängiger Pfade** (erzeugen in Linearkombination alle Pfade durch Graphen).
- **Zyklomatische Zahl:** Anzahl linear unabhängigen Pfade eines (zyklischen) Graphen.
- Sei **G = (V, E)** mit
  - e = |E(G)|, Anzahl Kanten (edges).
  - v = |V(G)|, Anzahl Knoten (vertices).
- Dann gilt **cn(G) = e - v + 1 (cyclomatic number).**

Literatur:

E. Riedemann: **Testmethoden für sequentielle und nebenläufige Software-Systeme**
http://www.ub.tu-dortmund.de/katalog/titel/687299

- Abschnitt 16.1 (S.439-444)
- Zyklomatische Zahl (S.440-441)

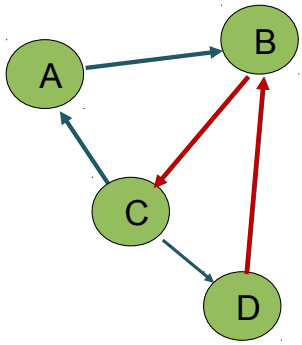
H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Zyklomatische Zahl: Alternative Definition

- **Alternative (äquivalente) Definition:** Anzahl zu entfernender Kanten stark zusammenhängenden Graphen, um **Spannbaum** zu erhalten
- **Beispiel:**
  cn(G) = 5 - 4 + 1 = 2

  (NB: **Entfernung beider Kanten** B-C und D-B ergibt einen Spannbaum des Graphen !)



Literatur:

E. Riedemann: **Testmethoden für sequentielle und nebenläufige Software-Systeme**
http://www.ub.tu-dortmund.de/katalog/titel/687299

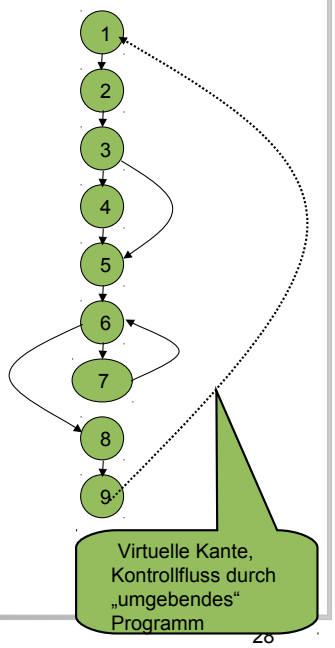
- Abschnitt 16.1 (S.439-444)
- Zyklomatische Zahl (S.440-441)

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Zyklomatische Komplexität eines Programmes

- Für **Definition dieser Zahl** als Metrik für Programme, Kontrollflussgraphen **„virtuelle“ Kante vom Endknoten zurück zum Anfangsknoten** hinzufügen, um zyklischen Graphen zu erhalten. (**Annahme:** Programm hat nur einen Endpunkt.)
- v(G) = e - v + 2: **Zyklomatische Komplexität** („McCabe-Metrik“) eines Kontrollflussgraphen G.
- **Beispiel:** v(G) = 10 - 9 + 2 = 3.



Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

T.J. McCabe: A Complexity Measure, IEEE Transactions on Software Engineering Vol. 2, No. 4, p. 308 (1976)

Mehrere Endpunkte

Bislang Annahme: Programm hat nur ein Endpunkt.

Programm mit mehreren Endpunkten: Für jeden Endpunkt „virtuelle“ Kante zum Startpunkt einfügen. → Zyklischen Graph erhalten.

Zyklomatische Komplexität des Kontrollflussgraphen G eines Programms **mit mehreren Endpunkten**:
$\nu(G) = e - n + p + 1$

- e Zahl der Kanten
- n Zahl der Knoten
- p Zahl der Endpunkte des Programms

[NB: Annahme weiterhin: Nur ein Startpunkt.]

Literatur:

H. Balzert: **Lehrbuch der Software-Technik/2 –** Software-Management, Software-Qualitätssicherung
http://www.ub.tu-dortmund.de/katalog/titel/645541

- Abschnitt 5.11.2 – Die McCabe-Metrik (S.481-482)

Größenmaße

- Größe – Wie umfangreich ist Software?
- Skala: Verhältnisskala.
- Mögliche Maße: **NCSS, Anzahl Zeichen.**

NCSS (Non-commented source statements):

- Zählung der **Codezeilen** ohne Kommentar- und Leerzeilen.
- Genaue **Zählregeln** erforderlich.
- **Programmiersprachenabhängig.**
- **Leicht messbar.**

Weitere Komplexitätsmaße

Softwaremetrik	Beschreibung
Fan-in / Fan-out	**Fan-in** einer Funktion oder Methode X: Anzahl Funktionen oder Methoden, die X aufrufen. **Fan-out:** Anzahl Funktionen oder Methoden, die von X aufgerufen werden. **Hoher Fan-in:** X eng mit Rest des System verbunden. Änderungen an X können weitreichende Folgewirkungen haben. **Hoher Fan-out:** Gesamte Komplexität von X ist hoch, da Steuerungs-logik von X aufgerufene Komponenten koordinieren muss.
Länge der Bezeichner	Maß durchschnittlicher Länge von Bezeichnern (Namen von Variablen, Klassen, Methoden, etc.) im Programm. Je länger sie sind, desto aussagekräftiger sind sie (damit Programm verständlicher).
Tiefe der Verschachte-lung	Maß der Tiefe der Verschachtelung von if-Bedingungen im Programm. Tief verschachtelte if-Bedingungen: Schwer zu verstehen und potentiell fehleranfälliger.
Fog-Index	Maß durchschnittlicher Länge von Wörtern und Sätzen im Dokument. Je höher Fog-Index eines Dokuments ist, desto schwerer ist es zu verstehen.

Weitere Metriken für objektorientierte Programme

Kürzel	Bezeichnung	Erläuterung
NOV	Number of Variables	Anzahl der Instanzvariablen (member variables) einer Klasse
NOM	Number of Methods	Anzahl der Methoden (Operationen) einer Klasse
NORM	Number of Redefined Methods	Anzahl der in einer Klasse redefinierten Methoden