Software-Qualitätsmanagement

Vorlesung im Modul 10-202-2319 Software-Management

Sommersemester 2013

Prof. Dr. Hans-Gert Gräbe

http://bis.informatik.uni-leipzig.de/HansGertGraebe

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Quellcode-Analyse

Ansatz: Qualität von Systemkomponenten besteht nicht nur in deren **funktioneller Qualität** (Q.-Z. Funktionalität und Effizienz; Fokus der bisher besprochenen Qualitätssicherungs-Methoden Test und Verifikation), sondern auch in der **Qualität des Quellcodes** selbst (Q.-Z. Änderbarkeit, Übertragbarkeit sowie teilweise Benutzbarkeit).

Relevante Parameter:

- sinnvolle **Granularität** der Komponenten längs **funktioneller Grenzen**.
- sinnvolle **Schnittstellengestaltung** für die Zusammenarbeit der Komponenten untereinander.

Kann in quantitativen Parametern der **Bindung** (innerhalb einer Komponente) und **Kopplung** (zwischen Komponenten) erfasst werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bindung und Kopplung

Die Bindung innerhalb einer Systemkomponente und die Kopplung der Systemkomponenten untereinander bestimmen die Struktur eines Software-Systems.

Bindung (*cohesion*) ist ein qualitatives Maß für die Kompaktheit einer Systemkomponente. Es werden dazu die Beziehungen zwischen den Elementen einer Systemkomponente betrachtet.

Kopplung (*coupling*) ist ein qualitatives Maß für die Schnittstellen zwischen den Systemkomponenten. Es werden der Kopplungs-mechanismus, die Schnittstellenbreite und die Art der Kommunikation betrachtet.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Je stärker die Bindungen der Systemkomponenten im Vergleich zu den Kopplungen, desto ausgeprägter ist die Struktur und Modularität eines Systems.
  - Bindung auf der Ebene der Funktionen (Methoden): Wie weit ist abgrenzbare Funktionalität an einer Stelle zusammengefasst?
  - Bindung auf der Ebene der Datenabstraktionen (Module und Klassen): Wie weit ist datenmäßig zusammengehörende Funktionalität zusammengefasst?
  - Informale Bindung (Pakete): Wie sind Datenabstraktionskonzepte an Datenstrukturen gebunden?
- Die Forderung nach guter Modularität wird erfüllt, wenn die Kopplungen minimiert und die Bindungen maximiert werden.
- Für **Komponenten** spielt der Bindungsgrad eine qualitätsrelevante Rolle, für **Systeme** die Ausgestaltung der Kopplung zwischen den Komponenten.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bindung auf der Ebene der Funktionen und Methoden

- Gute Bindung liegt vor, wenn nur solche Elemente zu einer Einheit (Funktion bzw. Methode) zusammengefasst werden, die auch zusammen gehören.
- Bindung von Funktionen wird nur qualitativ erfasst.

Ziel: Erreichen einer guten funktionalen Bindung.

- Alle Elemente sind an der Verwirklichung einer einzigen, abgeschlossenen Funktion beteiligt.
- Komplexe Funktionen werden realisiert, indem Teilaufgaben an andere Funktionen delegiert werden, die selbst funktional gebunden sind (Manager-Funktionen).

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Kennzeichen einer guten funktionalen Bindung:

- Alle Teile tragen dazu bei, ein einzelnes spezifisches Ziel zu erreichen.
- Es gibt keine überflüssigen Teile.
- Die Aufgabe kann mit genau einem Verb und genau einem Objekt beschrieben werden.
- Austausch gegen andere Funktion oder Methode, welche denselben Zweck erfüllt, leicht möglich.
- Hohe Kontextunabhängigkeit, d.h. einfache Beziehungen zur Umwelt.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorteile einer funktionalen Bindung:

- Hohe Kontextunabhängigkeit (die Bindungen befinden sich innerhalb der Prozedur, nicht zwischen Prozeduren).
- Geringe Fehleranfälligkeit bei Änderungen,
- Hoher Grad der Wiederverwendbarkeit,
- Leichte Erweiterbarkeit und Wartbarkeit, da sich Änderungen auf isolierte, kleine Teile beschränken.

- Konzept der Bindung verallgemeinert die (konzeptuellen) Regeln für „guten Code“ zu Regeln für „guten Software-Entwurf“.
- Die Bindungsart einer Prozedur lässt sich nicht automatisch ermitteln, sondern nur durch manuelle Prüfmethoden.
- Entsprechende Untersuchungen sind noch im experimentellen Stadium und haben weitgehend informellen Charakter.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bindung auf der Ebene von Datenabstraktionen und Klassen

Beschreibt das Zusammenwirken verschiedener Funktionen, welche derselben Datenabstraktion oder Klasse zuzuordnen sind.

- Voraussetzung: Alle Methoden sind gut funktional gebunden

Gute Bindung (*model cohesion*) liegt vor, wenn

- die Klasse ein einzelnes semantisch bedeutungsvolles Konzept repräsentiert,
- die Klasse keine verborgenen Klassen enthält und
- keine Operationen enthält, die an andere Klassen delegiert werden können.

Wird in der Literatur auch als Kohärenz bezeichnet.

Für Klassen ist weiter die Bindung innerhalb von Vererbungsstrukturen wesentlich.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Informale Bindung auf der Ebene von Paketen

Datenabstraktionen sollen dem Prinzip der guten informalen Bindung genügen.

- liegt vor, wenn mehrere, in sich abgeschlossene, funktional gebundene Zugriffsoperatoren, die zu einer Datenabstraktion gehören, auch nur auf einer einzigen Datenstruktur operieren.
- **Idee**: hinter der gemeinsamen Funktionalität liegt auch ein gemeinsames Datenmodell

Merkmale:

- Unterstützt das Geheimnisprinzip, d.h. die Datenstruktur gehört nur zu einer Datenabstraktion,
- Änderungen der Datenstruktur tangieren nur eine Datenabstraktion,
- Problem der Vermischung von Zugriffsoperationen, die alle auf derselben Datenstruktur operieren, wird vermieden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Bindung in Vererbungsstrukturen

- Die ganze Vererbungshierarchie muss untersucht werden.
- **Starke Vererbungsbindung** liegt vor, wenn die Hierarchie eine Generalisierungs-/Spezialisierungshierarchie im Sinne der konzeptuellen Modellierung ist.
- **Schwache Vererbungsbindung** liegt vor, wenn die Hierarchie nur zum "*code sharing*" verwendet wird.
- Das **Ziel** jeder neu definierten Unterklasse muss sein, ein einzelnes semantisches Konzept auszudrücken.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Analyse der Kopplung

Typische Ansätze für Kopplungsmetriken

- **fan-in**: Gewichtete Summe der Komponenten, deren Funktionalität in der zu vermessenden Komponente verwendet wird.
- **fan-out**: Gewichtete Summe der Komponenten, welche die Funktionalität der zu vermessenden Komponente verwenden.

Erfahrungen legen folgende Zielgrößen nahe:

geringe fan-in-Werte

- Grund: Delegierungsprinzip sinnvoll einsetzen

hohe fan-out-Werte

- Grund: hohe Verwendbarkeit deutet auf gute Struktur hin

geht nicht global, da Summe fan-in = Summe fan-out

Lösung: Manager-Komponenten identifizieren.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

OO-Spezifik: Vererbung als Bindung oder Kopplung?

Vererbung als Kopplung:

- gute Vererbungsstruktur hat enge Kopplung, gute Systemstruktur möglichst lose Kopplung

Vererbung als Bindung:

- gute Systemstruktur hat enge Bindung

Vererbung hat Bindungscharakter, also möglichst alle Klassen einer Hierarchie in *einer* Komponente versammeln.

- Implementations-Vererbung darf nicht über Komponenten-grenzen hinweg erfolgen, da die Korrumpierungsmöglichkeiten der Kopplungseffekte kaum zu überschauen sind.

Vererbungsmetriken werden deshalb den Komponentenmetriken zugerechnet.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einführung

Quantitative Aussagen über die Produktqualität einer Systemkomponente können mit Hilfe von **Metriken** ermittelt werden.

- Mit solchen Metriken sind heute nur einfache Aussagen über Eigenschaften einer Komponente möglich.
- Eine Metrik bewertet ein Software-System immer nur unter einem sehr speziellen Blickwinkel.
- Aussagekräftiger Gesamteindruck von einer Systemkomponente nur durch Auswertung einer Gruppe von Metriken, oft auch nur im Vergleich zu Parametern anderer, bereits im Einsatz befindlicher Komponenten.
- Metriken können nicht nur für bereits implementierte Komponenten, sondern auch schon entwicklungsbegleitend eingesetzt werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Metriken zum Erfassen der prozeduralen Komplexität

Ziel: Bewertung eines Produkts (Entwurfsdokuments, Grob/Fein-Entwurf, Code, Designdokumentation usw.) mittels Metriken

Schwerpunkt: Zuverlässigkeit, Änderbarkeit

Umfangsmetriken

- sind die ältesten Metriken
- stellen ab auf die textuelle Komplexität
- verwenden einfach verfügbare Informationen (Anzahl an Programmzeilen, Dateigröße, Zahl der Funktionen, ...)
- Vertreter: LOC, Halstead-Metrik, Function Points (zur Erfassung des Umfangs verbaler Anforderungen)

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Logische Strukturmetriken (Kontrollfluss-Metriken)

- Analyse des Kontrollfluss-Graphen
- Wird samt seiner Begleitobjekte (Symboltabelle) sowieso vom Compiler ausgewertet
- Vertreter: McCabe-Metrik

Datenstrukturmetriken

- messen die Anzahl an Variablen, deren Gültigkeit und Lebensdauer sowie die Referenzierung der Variablen

Stilmetriken

- messen ob die Programme richtig eingerückt wurden und ob die Namenskonventionen eingehalten wurden

Interne Bindungsmetriken

- messen die syntaktische Bindung durch Prüfen des Codes jeder Komponente

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Beispiele

Einsatzgebiet	Kriterium	Metrik
Komponenten-analyse	Umfang	lines of code
	innere Struktur	Kontrollfluss-komplexität
	Schnittstelle	# Methoden pro Klasse Schnittstellenbreite

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Einsatzgebiet	Kriterium	Metrik
Systemanalyse	Umfang	lines of code
	Kopplung	# Aufrufe in/aus Komponenten
	OO-Strukturierung	OO-Metriken
Prozessanalyse	Aufwandsoptimierung	Zeiterfassung
	Dokumentenqualität	entdeckte Fehler pro Seite
	Prüfprozessqualität	# vorab gefundener Fehler / # in der Sitzung gefundener Fehler

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Umfangsmetriken – Die Halstead-Metrik

Misst die textuelle Komplexität eines Programms, indem die Zahl der verwendeten Funktionen und der verwendeten Variablen ins Verhältnis gesetzt werden.

- Es wird jeweils die Gesamtzahl (Programmtext) und die Zahl verschiedener Objekte (Symboltabelle) bestimmt.
- $\boldsymbol{\eta_1}$, $\mathbf{N_1}$ = Zahl der (verschiedenen) Funktionen, Operatoren, Symbole oder Schlüsselwörter (z. B.: +, -, *, /, **while**, **if**, ...)
- $\boldsymbol{\eta_2}$, $\mathbf{N_2}$ = Zahl der (verschiedenen) Variablen, Operanden ...

Interpretation:

- $\boldsymbol{\eta}=\eta_1+\eta_2$: Größe des Vokabulars
- $\mathbf{N}=N_1+N_2$: Länge der Implementierung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorteile:

- einfach zu ermitteln,
- bei jeder Programmiersprache verwendbar und
- gute Eignung der Metriken für die zu messenden Größen

Nachteile:

- nur der Implementierungsaspekt betrachtet und
- Mehrdeutigkeiten im Messansatz, z. B. bei den Klassifikationsregeln für Operatoren und Operanden

abgeleitete Größen:

- **D**= $\eta_1/2 \cdot N_2/\eta_2$,
  Parameter für die Schwierigkeit, den Code zu verstehen
- Interpretation:
  N_2/η_2 = durchschnittliches Vorkommen jeder Variablen,
  η_1 = Anzahl der verwendeten Funktionen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

```
int ZaehleVokale(String s) {
    int VokalAnzahl; char Zchn; int i;
    for(i=0; i < s.length(); i++) {
        Zchn=s[i];
        if  ((Zchn == 'A') || (Zchn == 'E') || (Zchn == 'I') ||
            (Zchn == 'O') || (Zchn == 'U'))  VokalAnzahl++;
    }
    return VokalAnzahl;
}
```

ZaehleVokale	1
VokalAnzahl	3
Zchn	7
s	3
i	5
Konstanten (6)	6

int	3
String	1
char	1
for	1
if	1
return	1

()	9	++	2
{}	2	[]	1
;	8	==	5
=	2	||	4
<	1	.	1
length	1		

η_1=17, N_1=44, η_2=11, N_2=25, D=19.32

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Kontrollfluss-Metriken – Die McCabe-Metrik

- Misst die strukturelle Komplexität eines Programms, indem eine Grapheninvariante, die **zyklomatische Zahl** V(G) des Kontrollflussgraphen, bestimmt wird.
- $V(G) = e - n + 2p$ mit
  - e = Anzahl der Kanten des Graphen
  - n = Anzahl der Knoten
  - p = Anzahl der Zusammenhangskomponenten
- Kontrollflussgraph wird für jede Prozedur aufgestellt (p=1).
- Für solchen Graphen gilt

$$\mathbf{V(G) = b + 1}$$

  mit **b** = Anzahl der Bedingungen.
- Zyklomatische Zahl ist additiv auf Komponenten.
- Lineare Teilstücke können zusammengezogen werden.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Vorteile:

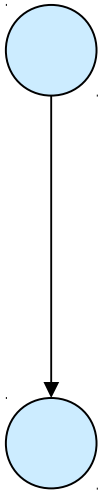
- einfach zu berechnen,
- grobes Maß für die Kontrollflusskomplexität: je größer, desto weiter weicht der Kontrollfluss vom linearen (V(G)=1) ab.

Nachteile:

- unterschiedliche Programmmerkmale werden stark vereinfacht
- Quellprogramm als zentrales Messobjekt überbetont
- Es wird nur das Programmgerüst, nicht aber die Komplexität einzelner und verschachtelter Anweisungen berücksichtigt

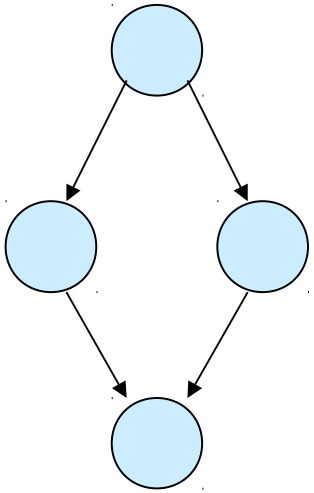
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Sequenz



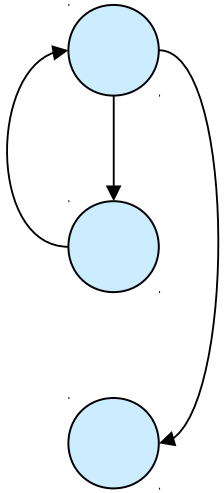
V(G) = 1 – 2 + 2
= 1

Auswahl

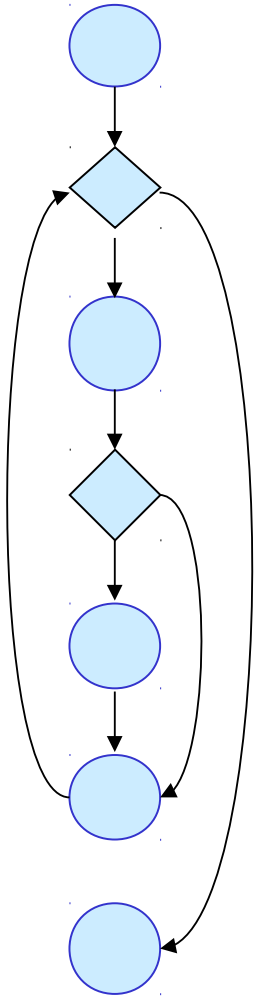


V(G) = 4 – 4 + 2
= 2

Abweisende Schleife



V(G) = 3 – 3 + 2
= 2



Das Programm ZaehleVokale hat die zyklomatische Zahl V(G) = 8 – 7 + 2 = 3

Es enthält zwei Bedingungen.