Freie Universität Berlin

Objektbasierte Softwaremaße

- Stephan Hagendorf
  Institut für Informatik
  FU Berlin

- 03.04.2006

Gliederung

Freie Universität Berlin

- Begrifflichkeiten
- Motivation
- „Metrics Suite“ von C&K
- Empirische Studien
- Schlussfolgerungen

Themenbezogene Begriffe

Freie Universität Berlin

- Objektorientierte Konzepte
- OOD
- Messtheorie, Maße
- Kopplung
- Bindung

Objektbasierte Maße – warum, wie?

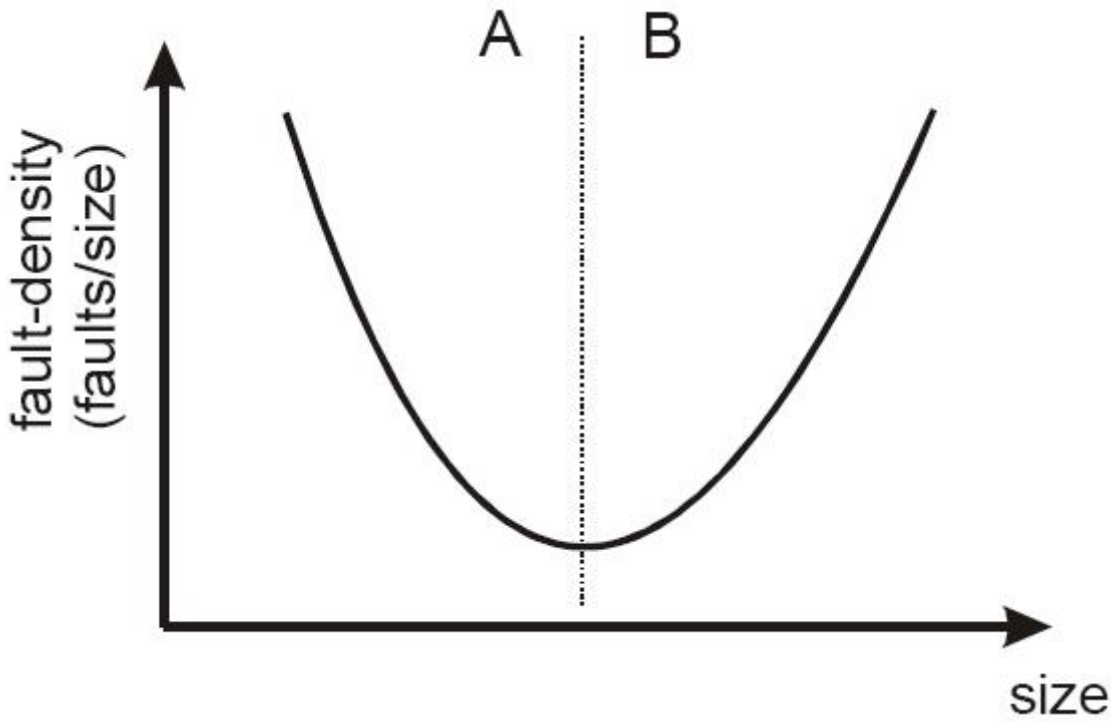
Freie Universität Berlin

- Traditionelle Maße evtl. Nicht geeignet für OO-Software
  - Beispiel: Umfang in LOC
- → Berücksichtigung der OO-Konzepte
- „Klassifikation“ der Maße:
  - Methodenebene
  - Klassenebene
  - Vererbungshierarchien
  - Aggregationshierarchien
  - Zeitpunkt der Anwendung

    - OOD-Methode → Testen

Vorsicht bei Maßen!

- Beispiel optimal Class Size
  - Genaue Definition gefragt
  - Bei der Auswertung darauf achten, welche Werte gegenüber stehen



Bedeutende Forscher auf dem Gebiet

Freie Universität Berlin

Shyam R. Chidamber

BSEE Abschluß der U. Bombay; MSEE Abschluß der U. Pittsburgh; PhD Abschluß vom MIT. Zur Zeit ist er „Founding Director of the Center for Information Technology and the Global Economy“ an der „Kogod School of Business of American University“

Chris F. Kemerer

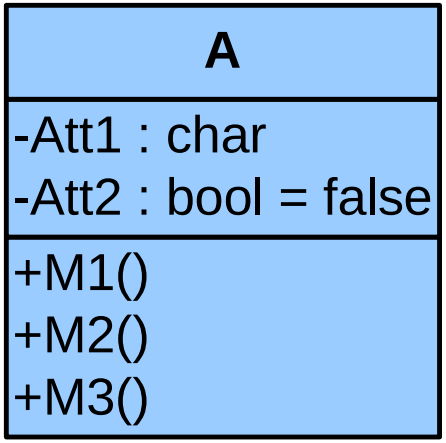
BS Abschluß an der „Wharton School“ an der Universität von Pennsylvania - PhD Abschluß an der „Carnegie Mellon University“. Er hat den „David Roderick Chair in Information Systems“ an der Katz School an der Universität Pittsburgh

- 6 OOD-Maße mit fester Basis in der Messtheorie
- Empirische Daten um Charakteristiken der Maße zu zeigen und Wege der Benutzung vorzuschlagen

WMC – Weighted Methods per Class

Freie Universität Berlin

- Gewichtete Summe der Anzahl der Methoden einer Klasse
- WMC (A) = $k1(M_1) + k2(M_2) + \ldots + k3(M_n)$



Klasse *A* besitzt drei Methoden. Damit gilt:

$WMC(A) = k1(M_1) + k2(M_2) + k3(M_3) = 3$

sofern *kx* die Gewichtung der einzelnen Methoden angibt und überall 1 ist

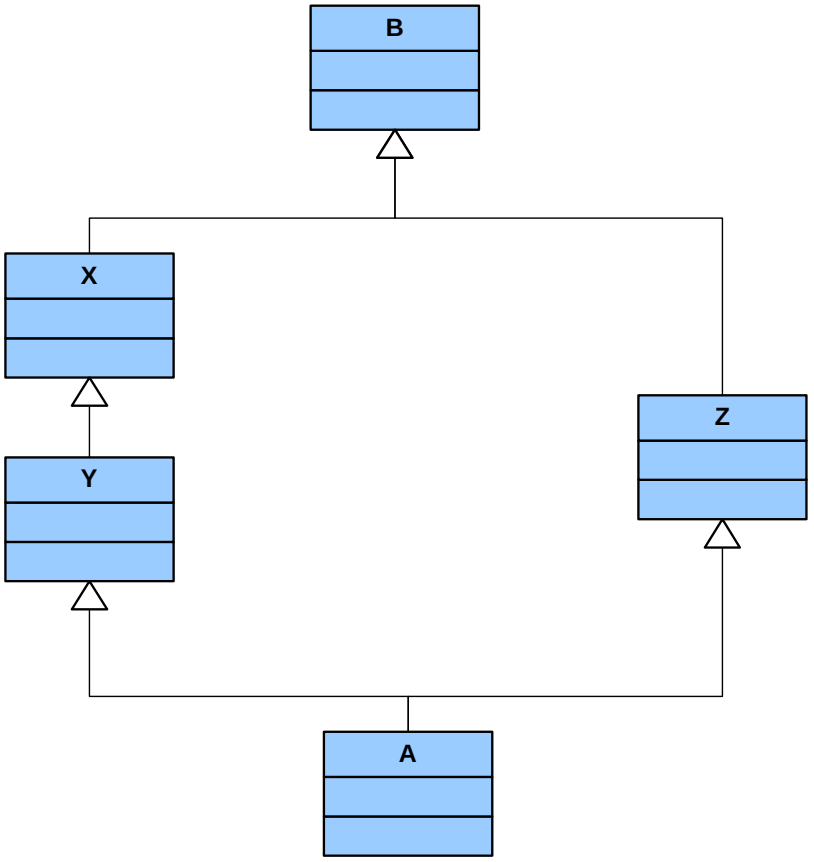
WMC - Interpretation

Freie Universität Berlin

- zeigt Schwierigkeit eine Klasse zu verstehen und weiterzuentwickeln
- mehr Methoden → größere Auswirkungen auf die Unterklassen
- viele Methoden → sehr anwendungsspezifisch also begrenzt wiederverwendbar
- Methoden → Defektwahrscheinlichkeit

DIT - Depth of Inheritence Tree

- Länge des maximalen Weges in der Klassenhierarchie von der Wurzel bis zur betrachteten Klasse



DIT(*B*) = 0
DIT(*X*) = DIT(*Z*) = 1, DIT(*Y*) = 2
Klasse *A:* DIT(*A*) = 3,
da der maximale Weg von *B* zu *A* über *X* und *Y* läuft
vollständige Hierarchie *H:* DIT(*H*) = 3

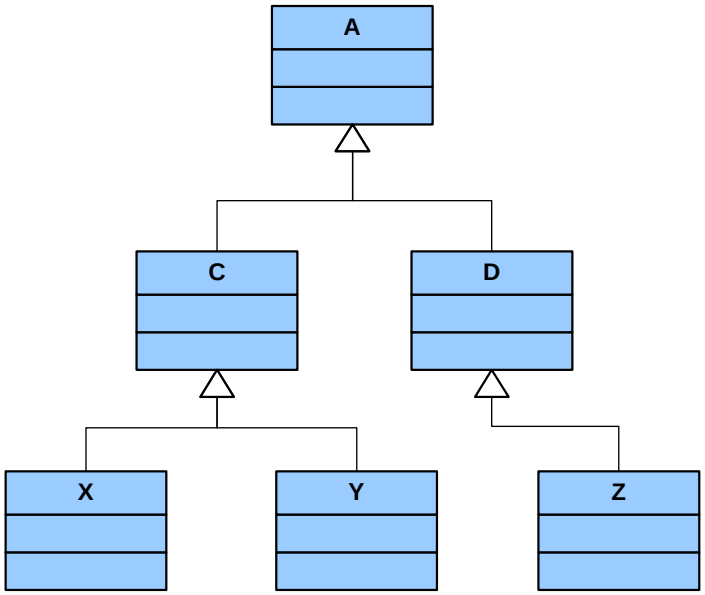
DIT - Interpretation

Freie Universität Berlin

- Je mehr Oberklassen existieren, desto komplexer ist die Klasse
- Änderungen in den Oberklassen → Auswirkung Klasse
- Tiefe Vererbungshierarchien gliedern Klassen im Sinne der Abstraktion
- Tiefe Hierarchien sind schwer wartbar

NOC - Number of (immediate) Children

- Anzahl der direkten Unterklassen
- Klassenmaß, da nur die Ebene unter der gegebenen Klasse betrachtet wird



Für die Basisklasse *A* in dieser Abbildung gilt: NOC(*A*) = 2
Analog ergeben sich die folgenden Werte für die verbleibenden Klassen:
NOC(*C*) = 2
NOC(*D*) = 1
NOC(*X*) = NOC(*Y*) = NOC(*Z*) = 0

NOC - Interpretation

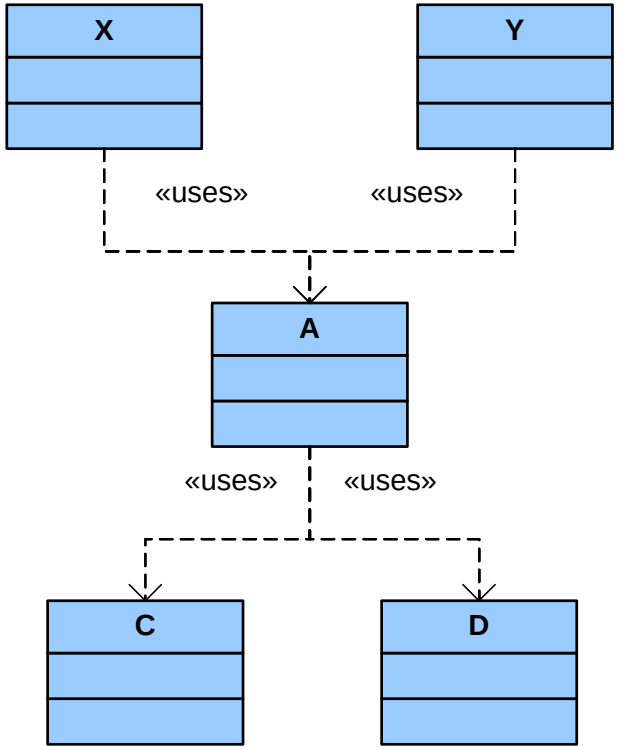
Freie Universität Berlin

- Je mehr Unterklassen existieren, desto höher ist der Grad der Wiederverwendung
- Die Anzahl der Unterklassen → Wichtigkeit der Klasse im gesamten Entwurf. → z.B. intensiveres Testen

CBO - Coupling Between Object Classes

Anzahl der Klassen, mit denen eine Klasse kommuniziert

- ohne Klassen in der Vererbungshierarchie
- Zwei Klassen sind gekoppelt, wenn eine Klasse Instanzvariablen und/oder Methoden der anderen Klasse benutzt → +1 bei beiden



Klasse *A*:

CBO(*A*) = 4

Für die anderen Klassen gilt:

CBO(*X*) = CBO(*Y*) =
CBO(*C*) = CBO(*D*) = 1

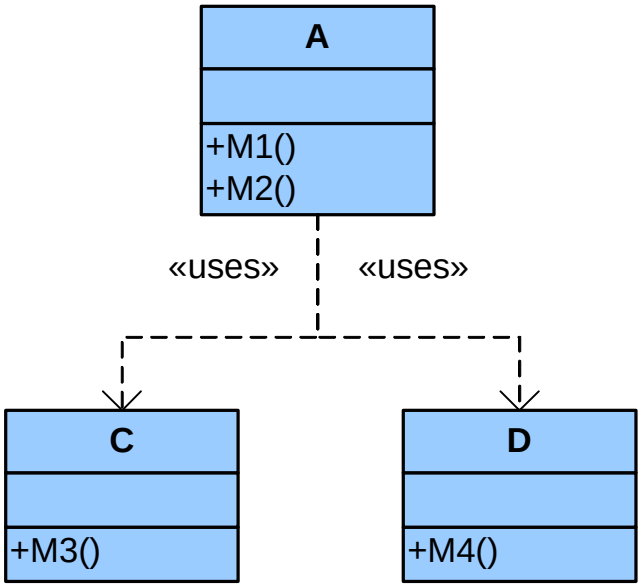
CBO - Interpretation

Freie Universität Berlin

- Je höher die Kopplung, desto schwieriger ist die Klasse zu verstehen
  - Schlechtere Wiederverwendbarkeit
  - Schwierigere Testbarkeit
  - Sie ist defektanfälliger bei Veränderungen

RFC - Response For a Class

- RFC(A) = Methoden, die potentiell ausgeführt werden können, wenn ein Objekt der Klasse A auf eingegangene Nachrichten reagiert
- Response Set einer Klasse:
  - Alle Methoden, die direkt aufgerufen werden können.
  - Alle Methoden, die durch Kopplung mit anderen Klassen erreichbar sind



Klasse *A* benutzt *C* und *D*. Angenommen, Methode M_1 ruft M_4 und Methode M_2 ruft M_3 auf. Damit ist das Response Set für *A:*

$RS_A = \{M_1, M_2, M_3, M_4\}$

und entsprechend

$\mathrm{RFC}(A) = |RS_A| = 4$

RFC - Interpretation

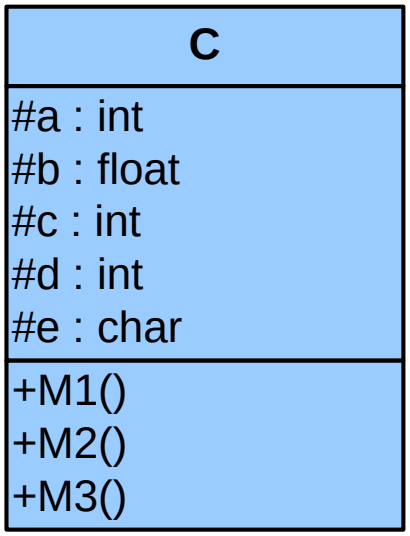
Freie Universität Berlin

- Je größer der RFC–Wert einer Klasse, desto komplexer ist sie
- Auswirkungen auf Testen der Methoden → Finden von geeigneten Testfällen?

LCOM - Lack of Cohesion in Methods

Freie Universität Berlin

- LCOM(A) = Anzahl der Paare von Methoden von A ohne gemeinsame Instanzvariablen minus der Anzahl der Paare von Methoden von A mit gemeinsamen Instanzvariablen



Methoden M_1, M_2 und M_3 der Klasse C operieren auf den Instanzvariablen $\{a, b, c\}$, $\{c, d\}$ und $\{e\}$.

paarweise Schnittmenge:

$M_1 \cap M_2 = \{c\}$

$M_1 \cap M_3 = \varnothing$

$M_2 \cap M_3 = \varnothing$

LCOM (C) = 2 - 1 = 1

LCOM kann nicht kleiner 0 werden

LCOM - Interpretation

Freie Universität Berlin

- „Zusammenhalt“, Bindung wird über die Verwendung von Instanzvariablen definiert
- Je kleiner LCOM, desto größer der Zusammenhalt der Methoden
- steigender LCOM-Wert → Klasse versucht viele verschiedene Ziele zu verfolgen → fehleranfälliger.

Frühe empirische Studien

Freie Universität Berlin

- **Sharble & Cohen 1993**
  - Benutzung der 6 CK-Maße um 2 OO-Methoden einzuschätzen und die “bessere” zu bestimmen
  - Die “responsibility-based” - Methode dominierte die “Datadriven”-Methode
- **Li and Henry 1993**
  - untersuchten 3 Jahre lang Wartungs- bzw. Änderungsdaten zweier Systeme
  - Benutzte Maße: 5 von 6 CK-Maßen (nicht CBO), + 3 LH, plus 2 Umfang
  - Sehr niedrige Werte für NOC, DIT
  - WMC, RFC korrelierten mit Umfang
  - OO Maße erklären signifikante Variation über dem Umfang

Freie Universität Berlin

- **Chidamber & Kemerer, 1994**
  - Softwareverkäufer, 634 C++ Klassen aus 2 Klassenbibliotheken für graphische Benutzerschnittstellen, für „rapid prototyping“ in versch. Anwendungen
  - Bei einer High-Tech-Firma, 1459 Smalltalk Klassen, Steuerungssoftware für Roboter (Produktion von VLSI-Schaltkreisen)
  - → Praktische Erfahrungen beim Sammeln von Maßen
  - Aufdeckung des Mangels von Nutzung der Vererbung (DIT, NOC)
  - Empfehlung die Maße zum Unterscheiden und identifizieren von „Design-Ausreißern“ zu benutzen

Weitere Studien

- **Cartwright & Shepperd, 1996**
  - Subsystem eines großen britischen Telekommunikationssytems, 32 Klassen, 133000 LOC C++
  - benutzten unter anderen DIT und NOC
  - Korrelation zwischen DIT und Benutzerproblemen trotz kleiner Abweichung
  - Zweifel an der Effektivität der Benutzung von Vererbung
  - relativ wenig Gebrauch von Vererbung und Polymorphie
  - Klassen mit größter Änderungsdichte relativ weit oben

Freie Universität Berlin

- **Chidamber, Darcy, Kemerer, 1998**
  - 3 kommerzielle „Financial Service“-Applikationen
  - Produktivität von 45 Klassen C++ System
  - Wiederverwendungsaufwand von 27 Klassen Objective C
  - Design Aufwand von 25 Klassen, 40 Seiten Design
  - niedrige Varianz in NOC, DIT
  - WMC, RFC, CBO stark korreliert
  - Hohe CBO und LCOM Werte wurden assoziiert mit
    - niedrigerer Produktivität
    - höherem Wiederverwendungsaufwand
    - größerem Design-Aufwand

Grenzwerte – ein Beispiel

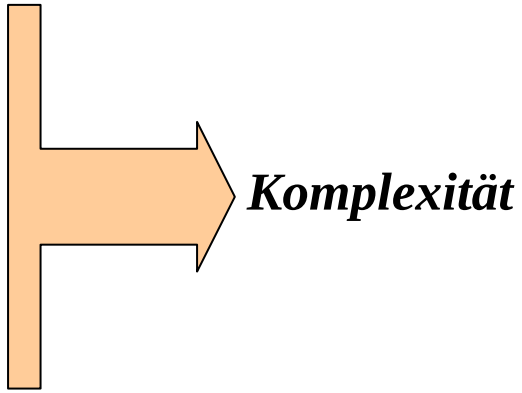
Freie Universität Berlin

- Software Assurance Technology Center, Goddard Space Flight Center NASA: „Applying Object Oriented Metrics“
- Jede Klasse, die *mindestens zwei* der folgenden Kriterien erfüllt, wird als kritisch gekennzeichnet:
  - RFC > 100
  - CBO > 5
  - RFC > 5 * WMC mit k=1 für alle Methoden
  - WMC > 100
  - WMC mit k=1 für alle Methoden > 40

Ist auf nur wenige Anwendungsarten anwendbar!!!

Fazit

- Hohe Werte von CK Maßen gehen (i.A.) einher mit:
  - niedriger Produktivität
  - Hohem Aufwand Klassen wiederzuverwenden
  - Hohem Aufwand Klassen zu designen
  - Schwierigkeiten Klassen zu implementieren
  - Der Anzahl von Wartungsbedingte Veränderungen
  - Der Anzahl der defekten Klassen
  - Problemen, die von Benutzern berichtet werden



- Entwickler gebrauchen fast keine Vererbung
- CK-Maße sind nicht programmiersprachenunabhängig
- Diese Maße sind „indirekte Anzeiger“ abstrakter Begriffe
- LCOM umstritten
- CK-Maße können helfen Aufwand zu fokussieren
- Tools: für empirischen Untersuchungen, Crocodile, PlugIn für Rational Rose, **ckjm** etc.

Ausblick

Freie Universität Berlin

- besserer Entwurf → bessere Werte von CK-Maße? → CK-Maße zu allgemein?
- Allgemeine Probleme von Maßen + wer legt Grenzen fest (fehlende Erfahrung?)
- eine Reihe weiterer objektorientierter Maße → Weitere Validierung der Maße, *bestehende* validieren
- Erfahrungen in unterschiedlichen Anwendungen und Programmiersprachen sammeln

Fragen und Antworten