DHBW Karlsruhe

Advanced Software Engineering

Software Metriken

Agenda

Einführung

- Sinn und Zweck der Softwaremessung
- Dimensionen der Substanz Software
- Objekte der Softwaremessung
- Ziele der Softwaremessung

Software Metriken

Praxisbeispiel

Einführung

Tom DeMarco: „You cannot control, what you cannot measure.“

Einführung | Sinn und Zweck der Softwaremessung

Zum Verständnis der Software

- Welche Bausteintypen existieren?
- Welche Beziehungen bestehen zwischen den Bausteintypen?
- Eigenschaften dieser Bausteintypen helfen diese zu klassifizieren
- Größe in Zeilen, Wörtern oder Symbolen
- Zahlen sind genauer als Sprache und Diagramme

Einführung | Sinn und Zweck der Softwaremessung

Zum Vergleich der Software

- Zwischen 2 Softwareprodukten entscheiden
- Vergleich von Versionen desselben Systems
- Für den Vergleich sind Zahlen Grundvoraussetzung

Einführung | Sinn und Zweck der Softwaremessung

Zur Vorhersage

- Softwareentwicklung und -wartung kosten Zeit und Geld
- Was kann der Käufer für sein Geld erwarten?
  - Welche Funktionalität zu welcher Qualität?
  - Dauer eines Projekts in Tagen oder Monaten
  - Anzahl der Personentagen
- Kunde braucht Informationen für seinen Entscheidungsprozess
- Zahlen geben diese Informationen am besten

Einführung | Sinn und Zweck der Softwaremessung

Zur Projektsteuerung

- Stand des Projekts feststellen *(was ist bereits abgeschlossen, was noch zu entwickeln?)*
- Qualität des fertiggestellten Anteils messen
- Aufgrund dieser Zahlen Entscheidungen treffen

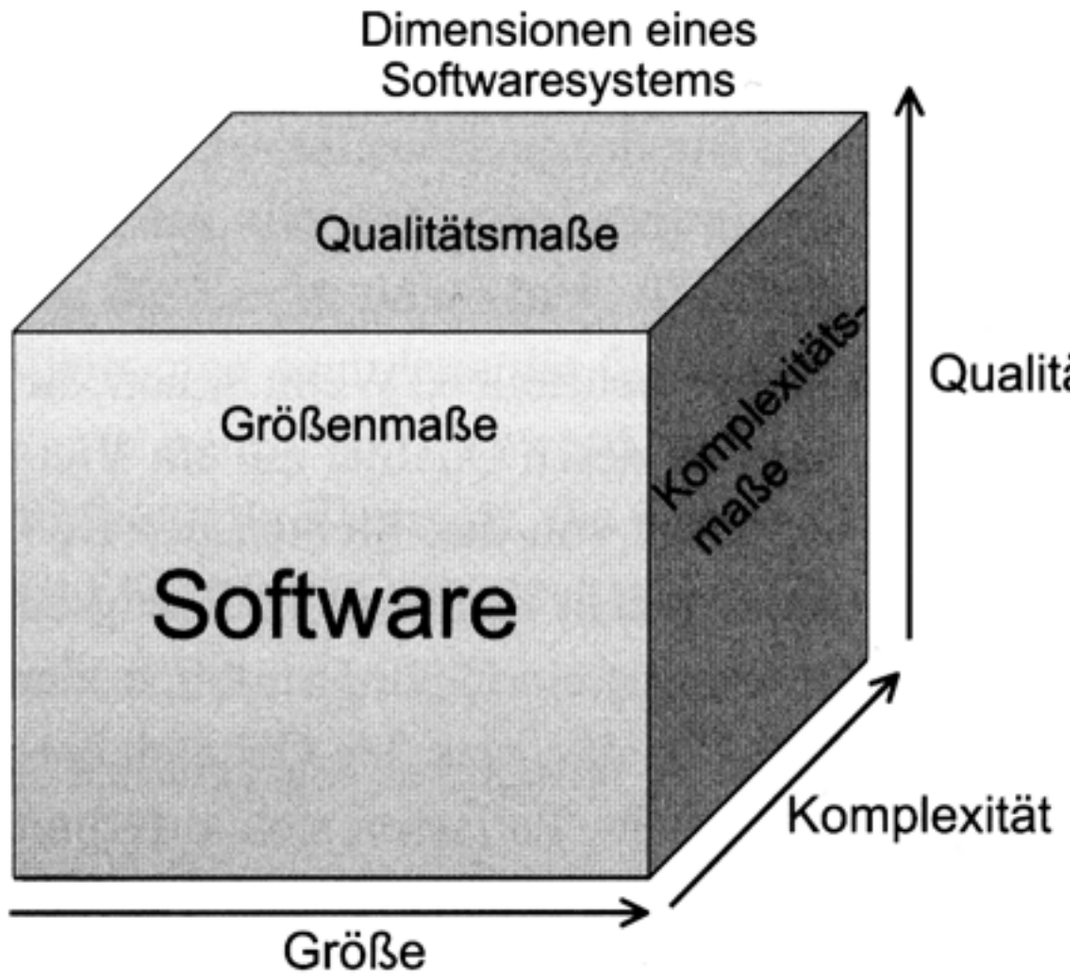
Einführung | Sinn und Zweck der Softwaremessung

Zur zwischenmenschlichen Verständigung

- *„Die Software ist groß.“* - relativ zu was?
- *„Die Aufgabe ist komplex.“* - relativ zu was?
- Zahlen geben genauere Auskunft darüber
  - *„Die Software hat 16.668 Anweisungen.“*
  - *„Die Software weist 100 Fehler auf.“*

Einführung | Dimensionen der Substanz Software

- Quantitätsmetrik
- Komplexitätsmetrik
- Qualitätsmetrik



Quantitätsmetrik

- Mengenzahlen (Menge aller Wörter, Anforderungen, Modelltypen, Anweisungen)
- Umfang von Software
- Relevante Mengen erkennen

Komplexitätsmetrik

- Verhältniszahlen für die Beziehungen zwischen Mengen und deren Elementen
- Die Zahl der Beziehungen ist eine Aussage über Komplexität
- Relevante Beziehungen erkennen

Einführung | Dimensionen der Substanz Software

Qualitätsmetrik

- Güte einer Software beurteilen
- Qualität ist der Grad, zu dem eine vereinbarte Norm eingehalten wird. —> Distanz zwischen Soll und Ist
- Qualität verursacht Kosten
  - Zu geringe Qualität verursacht Kosten bei Wartung und Weiterentwicklung
  - Zu hohe Qualität verursacht Mehrkosten bei der Entwicklung des Systems

Aus Sicht der Elementtypen

- Diagramme, Tabellen, Natursprachliche Texte
- Codestrukturen, Codeelemente
- Anforderungselemente
- Entwurfselemente
- Testelemente

Aus Sicht des Systembenutzers

- System/Benutzer-Interaktionen
- Systemkommunikation
- Systemausgabe
- Benutzerdokumentation

Aus Sicht des Systemintegrators

- Programme
- Daten
- Schnittstellen
- Systemdokumentation
- Fehlermeldungen

Einführung | Ziele der Softwaremessung

Einmalige Messungen

- Anwender hat doppelte Anwendungssysteme —> Welche der Anwendungssysteme sollen erhalten bleiben?

- Anwender übernimmt ein System in Wartung —> Auf was lässt er sich ein?

- Anwender möchte bestehende Systeme migrieren —> Um welchen Umfang handelt es sich

- Anwender will seine Anwendungen auslagern —> Was soll die Erhaltung und Weiterentwicklung kosten?

- Anwender steht vor einer Neuentwicklung —> Wie groß und wie komplex war die bestehende Anwendung?

Kontinuierliche Messungen

- Kosten und Nutzen alternativen Strategien
- Vergleich verschiedener Systeme
- Vergleich mit den Industriestandards (Benchmarking)
- Informationen über den Gesundheitsstand des Systems

Kontinuierliche Messungen

- Veränderungen verfolgen
- Veränderungen der Quantität
- Reduzierung der Komplexität
- Erhöhung der Qualität
- Verbesserung der Schätzgenauigkeit

Software Metriken

Software Metriken | ANDC

ANDC (Average Number of Derived Classes)

- Durchschnittliche Anzahl von Unterklassen einer Klasse
- ANDC = Anzahl Unterklassen / Anzahl aller Klassen
- Nicht gezählt werden
  - Interfaces
  - Bibliotheksklassen

Software Metriken | AHH

AHH (Average Hierarchy Height)

- Durchschnittliche Pfadlänge des Aufrufs einer Root Klasse, aus einer Klasse
- AHH = Summe aller maximalen Pfadlängen / Gesamtanzahl aller Root Klassen
- Geringer Wert deutet auf eine allgemein flache Klassenhierarchie hin
- Hoher Wert deutet auf übermäßige Nutzung von Vererbung hin
- Nicht gezählt werden
  - Interfaces
  - Bibliotheksklassen

Software Metriken | NOP, NOC, NOM, LOC

NOP (Number of Packages)

- Anzahl von Paketen (Packages in Java oder Namespaces in C++)

NOC (Number of Classes)

- Anzahl System interner Klassen

NOM (Number of Methods)

- Anzahl von Methoden/Funktionen (Methoden oder globale Funktionen, …)

LOC (Lines of Code)

- Anzahl von Statements aller user-defined operations

Software Metriken | CYCLO

CYCLO (Cyclomatic Complexity)

- Gesamtanzahl möglicher Entscheidungspfade

Software Metriken | CALLS, FANOUT

CALLS (Number of Operation Calls)

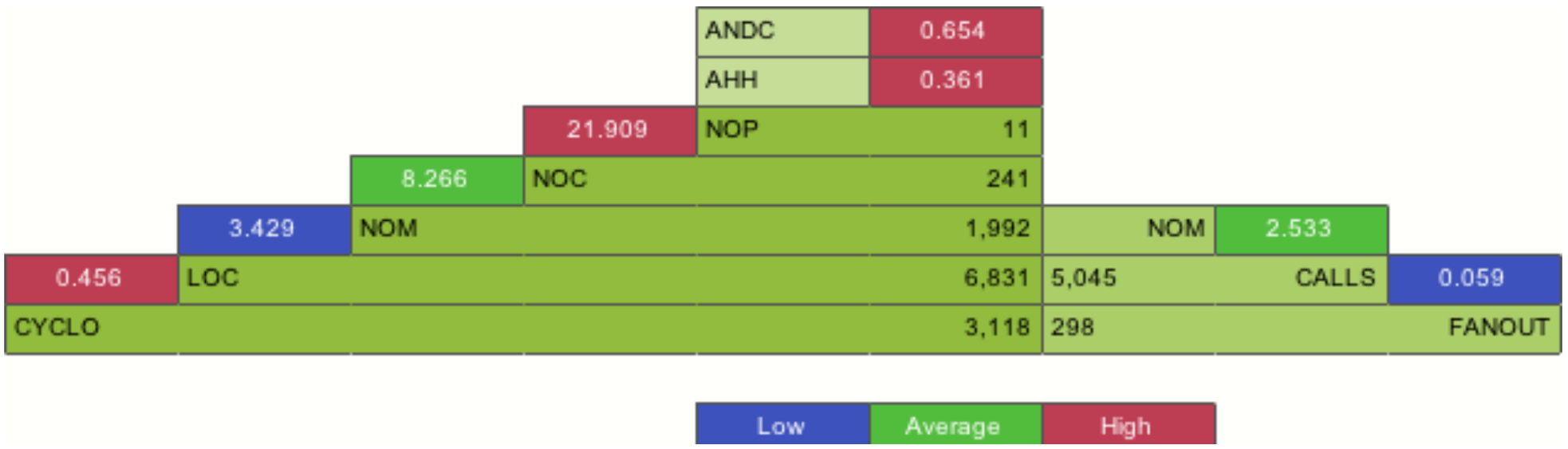
- Gesamtanzahl aller Methoden-/Funktionsaufrufe
- Aufrufe gleicher Funktionen innerhalb einer Funktion werden nur einmal gezählt

FANOUT (Number of Called Classes)

- Referenzierte Klassen
- Unterschiedlicher Vererbungszweige

Software Metriken | Overview Pyramid

Overview Pyramid *(Michele Lenza)*



Software Metriken | Overview Pyramid

Top Part: System Inheritance

- Grad der Vererbung im System

Left Part: Size and Complexity

- Größe und Komplexität des Systems
- Metriken sind nach Granularität in der Pyramide angeordnet

Right Part: System Coupling

- Grad der Kopplung im System

Software Metriken | System Strukturierung

High-Level Strukturierung

- NOC / NOP
- Sind Pakete grob- oder fein granuliert

Klassen Strukturierung

- NOM / NOC
- Verteilung der Methoden auf Klassen
- Hoher Wert deutet auf zu große Klassen hin

Software Metriken | System Strukturierung

Methoden Strukturierung

- LOC / NOM
- Verteilung des System-Verhaltens auf Methoden
  - Hoher Wert deutet darauf hin, dass Methoden zu viel tun

Intrinsic Operation Complexity

- CYCLO / LOC
- Zu erwartende Komplexität innerhalb von Methoden/Funktionen

Software Metriken | Kopplung

Kopplungsintensität

- CALLS / NOM
- Durchschnittliche Anzahl von Methodenaufrufen

Kopplungsverteilung

- FANOUT / CALLS
- Durchschnittliche Anzahl an Klassen, die in Methodenaufrufen beteiligt sind

Software Metriken | Thresholds

Reference Values Java

Metrik	Low	Average	High
CYCLO/LOC	0,16	0,2	0,24
LOC/NOM	7	10	13
NOM/NOC	4	7	10
NOC/NOP	6	17	26
CALLS/NOM	2,01	2,62	3,2
FANOUT/CALLS	0,56	0,62	0,68
ANDC	0,25	0,41	0,57
AHH	0,09	0,21	0,32

Software Metriken | Thresholds

Reference Values C++

Metrik	Low	Average	High
CYCLO/LOC	0,20	0,25	0,30
LOC/NOM	5	10	16
NOM/NOC	4	9	15
NOC/NOP	3	19	35
CALLS/NOM	1,17	1,58	2
FANOUT/CALLS	0,20	0,34	0,48
ANDC	0,19	0,28	0,37
AHH	0,05	0,13	0,21

Software Metriken | Thresholds

Reference Values PHP - PDepend

Metrik	Low	Average	High
CYCLO/LOC	0,16	0,2	0,24
LOC/NOM	7	10	13
NOM/NOC	4	7	10
NOC/NOP	6	17	26
CALLS/NOM	2,01	2,62	3,2
FANOUT/CALLS	0,56	0,62	0,68
ANDC	0,25	0,41	0,57
AHH	0,09	0,21	0,32

Praxisbeispiel

Praxisbeispiel | PHP - PDepend

- http://pdepend.org

- ~ $ pdepend --summary-xml=/tmp/summary.xml
  --overview-pyramid=/tmp/pyramid.svg
  /path/to/your/project

Praxisbeispiel | Continuous Integration

- http://jenkins-ci.org
- https://www.virtualbox.org/wiki/Downloads
- http://www.vagrantup.com/downloads.html