

Software-Metriken

Die Vermessung von Applikationen

DAS INHALTS- VERZEICHNIS

» Hier geht's
direkt
zum Buch

Inhalt

Vorwort	XV
Geleitwort zur 1. Auflage	XVII
Die Autoren	XIX
1 Softwaremessung	1
1.1 Das Wesen von Software	1
1.2 Sinn und Zweck der Softwaremessung	6
1.2.1 Zum Verständnis (Comprehension) der Software	7
1.2.2 Zum Vergleich der Software	7
1.2.3 Zur Vorhersage	7
1.2.4 Zur Projektsteuerung	8
1.2.5 Zur zwischenmenschlichen Verständigung	8
1.3 Dimensionen der Substanz Software	8
1.3.1 Quantitätsmetrik von Software	9
1.3.2 Komplexitätsmetrik von Software	9
1.3.3 Qualitätsmetrik von Software	10
1.4 Sichten auf die Substanz Software	10
1.5 Objekte der Softwaremessung	12
1.6 Ziele einer Softwaremessung	14
1.7 Zur Gliederung dieses Buches	17
2 Softwarequantität	19
2.1 Quantitätsmaße	19
2.2 Codegrößen	21
2.2.1 Codedateien	23
2.2.2 Codezeilen	23
2.2.3 Anweisungen	23
2.2.4 Prozeduren bzw. Methoden	23
2.2.5 Module bzw. Klassen	24
2.2.6 Entscheidungen	24
2.2.7 Logikzweige	24

2.2.8	Aufrufe	24
2.2.9	Vereinbarte Datenelemente	24
2.2.10	Benutzte Datenelemente bzw. Operanden	25
2.2.11	Datenobjekte	25
2.2.12	Datenzugriffe	25
2.2.13	Benutzeroberflächen	25
2.2.14	Systemnachrichten	26
2.3	Entwurfsgrößen	26
2.3.1	Strukturierte Entwurfsgrößen	26
2.3.2	Datenmodellgrößen	26
2.3.3	Objektmodellgrößen	27
2.3.3.1	Komponenten	28
2.3.3.2	Klassen	28
2.3.3.3	Klassenmethoden	28
2.3.3.4	Klassenattribute	28
2.3.3.5	Klasseninteraktionen	28
2.3.3.6	Objekte	28
2.3.3.7	Objektzustände	29
2.3.3.8	Objektinteraktionen	29
2.3.3.9	Aktivitäten	29
2.3.3.10	Entscheidungen	29
2.3.3.11	Verarbeitungsregel	29
2.3.3.12	Systemschnittstellen	29
2.3.3.13	Anwendungsfälle und Systemakteure	30
2.4	Anforderungsgrößen	30
2.4.1	Anforderungen	32
2.4.2	Abnahmekriterien	32
2.4.3	Anwendungsfälle	32
2.4.4	Verarbeitungsschritte	33
2.4.5	Oberflächen	33
2.4.6	Systemschnittstellen	33
2.4.7	Systemakteure	33
2.4.8	Relevante Objekte	33
2.4.9	Objektzustände	34
2.4.10	Bedingungen	34
2.4.11	Aktionen	34
2.4.12	Testfälle	34
2.5	Testgrößen	35
2.5.1	Testfälle	36
2.5.2	Testfallattribute	36
2.5.3	Testläufe	36
2.5.4	Testskripte bzw. Testprozeduren	36
2.5.5	Testskriptzeilen	37
2.5.6	Testskriptanweisungen	37
2.5.7	Fehlermeldungen	37

2.6	Abgeleitete Größenmaße	38
2.6.1	Function-Points	38
2.6.2	Data-Points	39
2.6.3	Object-Points	40
2.6.4	Use-Case-Points	41
2.6.5	Testfall-Points	41
3	Softwarekomplexität	43
3.1.1	Softwarekomplexität nach dem IEEE-Standard	46
3.1.2	Softwarekomplexität aus der Sicht von Zuse	47
3.1.3	Softwarekomplexität nach Fenton	47
3.1.4	Komplexität als Krankheit der Softwareentwicklung	48
3.1.5	Komplexitätsmessung nach Ebert und Dumke	50
3.1.6	Die Alpha-Komplexitätsmetrik	51
3.2	Steigende Softwarekomplexität	54
3.2.1	Codekomplexität – Warum Java komplexer als COBOL ist	55
3.2.2	Entwurfskomplexität – warum verschiedene Entwurfsansätze im Endeffekt gleich komplex sind	58
3.2.3	Anforderungskomplexität – warum die zu lösenden Aufgaben immer komplexer werden	60
3.3	Allgemeingültige Maße für die Softwarekomplexität	61
3.3.1	Sprachkomplexität	61
3.3.2	Strukturkomplexität	62
3.3.3	Algorithmische Komplexität	62
4	Die Messung der Softwarequalität	63
4.1	Qualitätseigenschaften nach Boehm	64
4.1.1	Verständlichkeit nach Boehm	65
4.1.2	Vollständigkeit nach Boehm	66
4.1.3	Portabilität nach Boehm	66
4.1.4	Änderbarkeit nach Boehm	66
4.1.5	Testbarkeit nach Boehm	66
4.1.6	Benutzbarkeit nach Boehm	67
4.1.7	Zuverlässigkeit nach Boehm	67
4.1.8	Effizienz nach Boehm	68
4.2	Gilb und die Quantifizierung der Qualität	68
4.2.1	Funktionalitätsmessung nach Gilb	69
4.2.2	Performanz-Messung nach Gilb	69
4.2.3	Zuverlässigkeitsmessung nach Gilb	70
4.2.4	Datensicherungsmessung nach Gilb	70
4.2.5	Effizienzmessung nach Gilb	70
4.2.6	Verfügbarkeitsmessung nach Gilb	71
4.2.7	Wartbarkeitsmessung nach Gilb	71
4.3	McCalls Qualitätsbaum	71
4.4	Eine deutsche Sicht auf Softwarequalität	74
4.4.1	Qualitätsbegriff	74

4.4.2	Qualitätsklassifizierung	74
4.4.3	Qualitätsmaße	75
4.4.4	Qualitätsgrößen	75
4.5	IEEE- und ISO/IEC-Standards für Softwarequalität	76
4.5.1	Funktionalität nach ISO 25010	77
4.5.2	Effiziente Performanz nach ISO 25010	77
4.5.3	Kompatibilität nach ISO 25010	77
4.5.4	Benutzbarkeit nach ISO 25010	77
4.5.5	Zuverlässigkeit nach ISO 25010	78
4.5.6	Sicherheit nach ISO 25010	78
4.5.7	Wartbarkeit nach ISO 25010	78
4.5.8	Portabilität nach ISO 25010	79
4.6	Zielgerichtete Softwarequalitätssicherung	79
4.6.1	Qualitätszielbestimmung	79
4.6.2	Qualitätszielbefragung	80
4.6.3	Qualitätszielbemessung	80
4.7	Automatisierte Softwarequalitätssicherung	81
4.7.1	Automatisierte Messung der Anforderungsqualität	82
4.7.2	Automatisierte Messung der Entwurfsqualität	83
4.7.3	Automatisierte Messung der Codequalität	84
4.7.4	Automatisierte Messung der Testqualität	86
4.8	Folgen fehlender Qualitätsmessung	87
5	Anforderungsmessung	89
5.1	Tom Gilbs Anstoß der Anforderungsmessung	91
5.2	Weitere Ansätze zur Anforderungsmessung	93
5.2.1	Der Boehm-Ansatz	93
5.2.1.1	Vollständigkeit	93
5.2.1.2	Konsistenz	94
5.2.1.3	Machbarkeit	94
5.2.1.4	Testbarkeit	94
5.2.2	N-Fold Inspektion	95
5.2.3	Parnas & Weis Anforderungsprüfung	95
5.2.4	Ableich der Anforderungen nach Fraser und Vaishnavi (Anforderungsprüfung)	96
5.2.5	Verfolgung der Anforderungen nach Hayes	96
5.2.6	Bewertung der Anforderungen nach Glinz	98
5.2.7	ISO-Standard 25030	99
5.2.8	Das V-Modell-XT als Referenzmodell für die Anforderungsmessung	99
5.3	Eine Metrik für Anforderungen von C. Ebert	100
5.3.1	Zahl aller Anforderungen in einem Projekt	101
5.3.2	Fertigstellungsgrad der Anforderungen	101
5.3.3	Änderungsrate der Anforderungen	102
5.3.4	Zahl der Änderungsursachen	102
5.3.5	Vollständigkeit des Anforderungsmodells	102
5.3.6	Anzahl der Anforderungsmängel	102

5.3.7	Anzahl der Mängelarten	103
5.3.8	Nutzwert der Anforderungen	103
5.4	Die Sophist-Anforderungsmetrik	103
5.4.1	Eindeutigkeit der Anforderungen	104
5.4.2	Ausschluss der Passivform bei den Anforderungen	104
5.4.3	Klassifizierbarkeit der Anforderungen	105
5.4.4	Identifizierbarkeit der Anforderungen	105
5.4.5	Lesbarkeit	105
5.4.6	Selektierbarkeit	105
5.5	Agile Anforderungsmetrik	106
5.6	Werkzeuge für die Anforderungsmessung	107
5.6.1	Anforderungsmessung in den früheren CASE-Werkzeugen	107
5.6.2	Anforderungsmessung im CASE-Tool SoftSpec	107
5.6.3	Anforderungsmessung in den gegenwärtigen Requirements Management Tools	109
5.6.4	Anforderungsmetrik aus dem Werkzeug TextAudit	109
5.6.4.1	Anforderungsgrößen	110
5.6.4.2	Anforderungskomplexitäten	111
5.6.4.3	Anforderungsqualitäten	111
5.6.4.4	Prüfung der Rupp-Regeln	111
5.6.4.5	Implementierung der Sophist-Metrik	112
5.6.5	Darstellung der Anforderungsmetrik	112
5.7	Gründe für die Anforderungsmessung	113
6	Entwurfsmessung	115
6.1	Erste Ansätze zu einer Entwurfsmetrik	116
6.1.1	Der MECCA-Ansatz von Tom Gilb	116
6.1.2	Der Structured-Design-Ansatz von Yourdon und Constantine	116
6.1.3	Der Datenflussansatz von Henry und Kafura	118
6.1.4	Der Systemgliederungsansatz von Belady und Evangelisti	119
6.2	Entwurfsmessung nach Card und Glass	120
6.2.1	Entwurfsqualitätsmaße	121
6.2.1.1	Modulgröße	122
6.2.1.2	Modulkohäsion	122
6.2.1.3	Modulkopplung	122
6.2.1.4	Modulkontrollspanne	122
6.2.1.5	Konsequenzen der Modularisierung	123
6.2.2	Entwurfskomplexitätsmaße	123
6.2.2.1	Relative Systemkomplexität	123
6.2.2.2	Strukturelle Systemkomplexität	124
6.2.2.3	Verarbeitungskomplexität	125
6.2.2.4	Entscheidungskomplexität	125
6.2.2.5	Prozedurale Komplexität	126
6.2.3	Erfahrung mit der ersten Entwurfsmetrik	126
6.3	Die SOFTCON Entwurfsmetrik	127
6.3.1	Formale Vollständigkeits- und Konsistenzprüfung	128

6.3.2	Technische Qualitätsmaße für den Systementwurf	129
6.3.2.1	Modularitätsmessung	129
6.3.2.2	Wiederverwendbarkeitsmessung	130
6.3.2.3	Portabilitätsmessung	130
6.3.2.4	Entwurfskomplexitätsmessung	130
6.3.2.5	Systemintegritätsmessung	131
6.3.2.6	Zeiteffizienz	131
6.3.2.7	Speichereffizienzmessung	131
6.4	Objektorientierte Entwurfsmetrik	132
6.4.1	Die OO-Metrik von Chidamer und Kemerer	133
6.4.1.1	Anzahl gewichteter Methoden pro Klasse (WMC)	134
6.4.1.2	Tiefe der Vererbungshierarchie (DIH)	134
6.4.1.3	Anzahl der Unterklassen (SUB)	134
6.4.1.4	Kopplung der Klassen (CBO)	135
6.4.1.5	Anzahl potenzieller Zielmethoden (RFC)	135
6.4.1.6	Zusammenhalt der Methoden (CBO)	135
6.4.1.7	Kritik der Chidamer/Kemerer-Metrik	136
6.4.2	MOOD-Entwurfsmetrik	136
6.4.2.1	Messung des Kapselungsgrades	137
6.4.2.2	Messung des Vererbungsgrades	138
6.4.2.3	Messung des Kopplungsgrades	138
6.4.2.4	Messung des Bindungsgrades	138
6.5	Entwurfsmetrik in UMLAudit	139
6.5.1	Entwurfsquantitätsmetrik	140
6.5.2	Entwurfskomplexitätsmetrik	142
6.5.2.1	Objektinteraktionskomplexität	143
6.5.2.2	Klassenhierarchiekomplexität	143
6.5.2.3	Klassen/Attributkomplexität	143
6.5.2.4	Klassen/Methodenkomplexität	143
6.5.2.5	Objektzustandskomplexität	144
6.5.2.6	Zustandsübergangskomplexität	144
6.5.2.7	Aktivitätenflusskomplexität	145
6.5.2.8	Anwendungsfallkomplexität	145
6.5.2.9	Akteurinteraktionskomplexität	145
6.5.2.10	Allgemeine Entwurfskomplexität	146
6.5.2.11	Mittlere Entwurfskomplexität	146
6.5.3	Entwurfsqualitätsmetrik	146
6.5.3.1	Klassenkopplungsgrad	147
6.5.3.2	Klassenkohäsionsgrad	147
6.5.3.3	Modularitätsgrad	148
6.5.3.4	Portabilitätsgrad	148
6.5.3.5	Wiederverwendbarkeitsgrad	149
6.5.3.6	Testbarkeitsgrad	149
6.5.3.7	Konformitätsgrad	149
6.5.3.8	Konsistenzgrad	150
6.5.3.9	Vollständigkeitsgrad	150

6.5.3.10	Erfüllungsgrad	151
6.5.3.11	Mittlere Entwurfsqualität	151
6.5.4	Entwurfsgößenmetrik	152
6.5.4.1	Data-Points	153
6.5.4.2	Function-Points	153
6.5.4.3	Object-Points	153
6.5.4.4	Use-Case-Points	154
6.5.4.5	Test-Points	154
6.6	Entwurfsmetrik für Webapplikationen	155
7	Codemetrik	157
7.1	Programmaufbau	157
7.2	Ansätze zur Messung von Codekomplexität	160
7.2.1	Halsteads Software Science	160
7.2.2	McCabes Zyklomatische Komplexität	162
7.2.3	Chapins Q-Komplexität	164
7.2.4	Elshofs Referenzkomplexität	165
7.2.5	Prathers Verschachtelungskomplexität	166
7.2.6	Weitere Codekomplexitätsmaße	167
7.3	Ansätze zur Messung von Codequalität	168
7.3.1	Der Codequalitätsindex von Simon	168
7.3.2	Der Maintainability-Index von Oman	169
7.3.3	Zielorientierte Codequalitätsmessung	171
7.3.3.1	Codeverständlichkeit	171
7.3.3.2	Codeportierbarkeit	172
7.3.3.3	Codekonvertierbarkeit	174
7.3.3.4	Codewiederverwendbarkeit	174
7.3.3.5	Codesicherheit	175
7.3.3.6	Codetestbarkeit	176
7.3.3.7	Codewartbarkeit	178
7.4	Codemetrik nach SoftAudit	179
7.4.1	Codequantitätsmetrik	179
7.4.2	Codekomplexität	180
7.4.2.1	Datenkomplexität	180
7.4.2.2	Datenflusskomplexität	180
7.4.2.3	Zugriffskomplexität	180
7.4.2.4	Schnittstellenkomplexität	181
7.4.2.5	Ablaufkomplexität	181
7.4.2.6	Entscheidungskomplexität	181
7.4.2.7	Verschachtelungskomplexität	182
7.4.2.8	Sprachkomplexität	182
7.4.2.9	Beziehungskomplexität	182
7.4.3	Codequalität	183
7.4.3.1	Sicherheit (Security)	183
7.4.3.2	Konformität (Conformity)	183
7.4.3.3	Datenunabhängigkeit (Data Independency)	184

7.4.3.4	Redundanzfreiheit (Non redundant)	184
7.4.3.5	Testbarkeit (Testability)	184
7.4.3.6	Wiederverwendbarkeit (Reusability)	185
7.4.3.7	Konvertierbarkeit (Convertibility)	185
7.4.3.8	Übertragbarkeit (Portability)	186
7.4.3.9	Modularität (Modularity)	186
7.4.3.10	Kommentierung (Commentation)	186
7.4.3.11	Weitere Qualitätsmerkmale	187
7.5	Beispiel einer Codemessung	187
8	Testmetrik	191
8.1	Testmessung in der früheren Projektpraxis	192
8.1.1	Das ITS-Projekt bei Siemens	192
8.1.2	Das Wella-Migrationsprojekt	193
8.2	Testmetrik nach Hetzel	195
8.3	Testmetrik bei IBM Rochester	197
8.4	Maßzahlen für den Systemtest	200
8.4.1	Testzeit	201
8.4.2	Testkosten	201
8.4.3	Testfälle	201
8.4.4	Fehlermeldungen	202
8.4.5	Systemtestüberdeckung	202
8.4.6	Empfehlungen von Hutcheson	203
8.4.7	Test-Points	203
8.5	Testmetrik im GEOS-Projekt	205
8.5.1	Messung der Testfälle	205
8.5.2	Messung der Testüberdeckung	208
8.5.3	Messung der Fehlerfindung	208
8.5.4	Auswertung der Testmetrik	210
8.6	Testmetrik nach Sneed und Jungmayr	211
8.6.1	Testbarkeitsmetrik	211
8.6.1.1	Testbarkeit auf der Unit-Test-Ebene	212
8.6.1.2	Testbarkeit auf der Integrationstestebene	212
8.6.1.3	Testbarkeit auf Systemtestebene	213
8.6.2	Testplanungsmetrik	214
8.6.3	Testfortschrittsmetrik	217
8.6.4	Testqualitätsmetrik	218
8.6.4.1	Testeffektivität	218
8.6.4.2	Testvertrauen	219
8.6.4.3	Testeffizienz	220
8.6.4.4	Restfehlerwahrscheinlichkeit	220
9	Produktivitätsmessung von Software	223
9.1	Produktivitätsmessung – Ein umstrittenes Thema	226
9.2	Softwareproduktivität im Rückblick	227

9.2.1	Dokumentenmessung mit dem Fog-Index	227
9.2.2	Produktivitätsmessung bei der Standard Bank of South Africa	228
9.2.3	Die Entstehung der Function-Point-Methode	229
9.2.4	Das COCOMO-I-Modell von Boehm	231
9.2.5	Putnams Softwaregleichung	233
9.2.6	Die Data-Point-Methode	235
9.2.7	Die Object-Point-Methode	237
9.2.8	Die Use-Case-Point-Methode	240
9.3	Alternative Produktivitätsmaße	242
9.4	Produktivitätsberechnung anhand der Softwaregröße	244
9.5	Aufwandserfassung	245
9.6	Arten von Softwareproduktivität	246
9.6.1	Programmierproduktivität	246
9.6.2	Designproduktivität	247
9.6.3	Analyseproduktivität	247
9.6.4	Testproduktivität	248
9.6.5	Gesamtproduktivität	248
9.7	Produktivitätsstudien	249
9.7.1	Studien über Softwareproduktivität in den USA	249
9.7.2	Studien über Softwareproduktivität in Europa	251
9.7.3	Probleme beim Produktivitätsvergleich	253
9.8	Produktivitätsmessung nach Wertbeitrag	254
9.9	Velocity – Produktivität in agilen Projekten	255
10	Die Messung der Wartungsproduktivität	257
10.1	Frühere Ansätze zur Messung der Wartbarkeit von Software	258
10.1.1	Stabilitätsmaße von Yau und Collofello	259
10.1.2	Maintenance-Umfrage bei der U.S. Air Force	260
10.1.3	Die Wartbarkeitsstudie von Vessey und Weber	262
10.1.4	Bewertung der Softwarewartbarkeit nach Berns	263
10.1.5	Die Wartungsuntersuchung von Gremillion	264
10.1.6	Wartungsmetrik bei Hewlett-Packard	264
10.1.7	Wartungsmessung nach Rombach	266
10.1.8	Messung der Wartbarkeit kommerzieller COBOL Systeme	267
10.1.9	Der Wartbarkeitsindex von Oman	268
10.2	Ansätze zur Messung der Wartbarkeit objektorientierter Software	271
10.2.1	Erste Untersuchung der Wartbarkeit objektorientierter Programme	271
10.2.2	Chidamer/Kemerers OO-Metrik für Wartbarkeit	272
10.2.3	MOOD-Metrik als Indikator der Wartbarkeit	273
10.2.4	Eine empirische Validation der OO-Metrik für die Schätzung des Wartungsaufwands	274
10.2.5	Der Einfluss einer zentralen Steuerung auf die Wartbarkeit eines OO-Systems	275
10.2.6	Kalkulation des Wartungsaufwands aufgrund der Programm- komplexität	275

10.2.7	Vergleich der Wartbarkeit objektorientierter und prozeduraler Software	276
10.2.8	Zur Änderung der Wartbarkeit im Laufe der Softwareevolution	278
10.3	Wartungsproduktivitätsmessung	280
10.3.1	Erste Ansätze zur Messung von Wartungsproduktivität	280
10.3.2	Messung von Programmwartbarkeit im ESPRIT-Projekt MetKit	283
10.3.3	Wartungsproduktivitätsmessung in der US-Marine	285
10.3.4	Messung der Wartungsproduktivität bei Martin-Marietta	287
10.3.5	Vergleich der Wartungsproduktivität repräsentativer Schweizer Anwender	288
11	Softwaremessung in der Praxis	293
11.1	Dauerhafte Messverfahren	295
11.1.1	Beteiligung der Betroffenen	295
11.1.2	Aufbauen auf vorhandener Metrik	296
11.1.3	Transparenz des Verfahrens	296
11.2	Beispiele dauerhafter Messverfahren	297
11.2.1	Die Initiative von Hewlett-Packard zur Softwaremessung	297
11.2.2	Prozess- und Produktmessung in der Siemens AG	300
11.3	Einmalige Messverfahren	305
11.3.1	Vereinbarung der Messziele	306
11.3.2	Auswahl der Metrik	307
11.3.3	Bereitstellung der Messwerkzeuge	307
11.3.4	Übernahme der Messobjekte	307
11.3.5	Durchführung der Messung	308
11.3.6	Auswertung der Messergebnisse	308
11.4	Beispiel einer einmaligen Messung	310
	Literatur	313
	Index	329