

Steve McConnell

Code Complete

Deutsche Ausgabe der Second Edition

**Microsoft'
Press**

Inhaltsverzeichnis

Vorwort	XXV
An wen wendet sich dieses Buch?	XXV
Wo können Sie solche Informationen sonst finden?	XXVI
Was dürfen Sie von diesem Buch erwarten?	XXVII
Warum dieses Buch geschrieben wurde	XXIX
Kritik und Anregungen	XXX
Technischer Support	XXXI
Danksagungen	XXXII

Teil I	Das Gießen des Fundaments	1
Kapitel 1	Willkommen auf der Softwarebaustelle	3
1.1	Was bedeutet »Software bauen«?	3
1.2	Warum ist der Bauabschnitt wichtig?	6
1.3	Wie Sie dieses Buch lesen sollten	7
	Zusammenfassung	7
Kapitel 2	Metaphern als Hilfsmittel für die Softwareentwicklung	9
2.1	Die Bedeutung von Metaphern	10
2.2	Das Verwenden von Softwaremetaphern	12
2.3	Gebräuchliche Softwaremetaphern	13
	Der Software-Autor: Quelltext schreiben	13
	Der Landwirt: Software anbauen	15
	Austernzucht: Das wachsende System	16
	Ein Haus errichten: Software bauen	17
	Softwaretechniken: Der geistige Werkzeugkasten	20
	Metaphern kombinieren	20
	Weiterführende Literatur	21
	Zusammenfassung	21
Kapitel 3	Zweimal messen, einmal sägen: Vorbereitung beim Softwarebau	23
3.1	Die Bedeutung guter Vorbereitung	24
	Ist für moderne Softwareprojekte Vorbereitung erforderlich?	24
	Ursachen für ungenügende Vorbereitung	25
	Plädoyer für eine gründliche Vorbereitung	27
3.2	Einordnen eines Softwareprojekts	30
	Vorbereitung bei iterativen Verfahren	32
	Wählen zwischen iterativen und sequentiellen Ansätzen	34
3.3	Die Problemdefinition	35
3.4	Die Anforderungen	37
	Wozu formelle Anforderungen?	37
	Der Mythos von felsenfesten Anforderungen	38
	Änderungen von Anforderungen während des Baus	39

	3.5 Die Architektur	42
	Typische Komponenten der Architektur	43
	3.6 Zeitbedarf für die Vorbereitung	55
	Weiterführende Literatur	56
	Zusammenfassung	58
Kapitel 4	Schlüsselentscheidungen für den Bau.	61
	4.1 Auswählen der Programmiersprache	61
	Sprachbeschreibungen	63
	4.2 Konventionen für die Programmierung	66
	4.3 Ihr Platz auf der Technologiewelle	67
	Beispiel für das Programmieren <i>in eine</i> Sprache	68
	4.4 Auswählen von zentralen Bautechniken	69
	Zusammenfassung	71
Teil II	Erstellen von qualitativ hochwertigem Code.	73
Kapitel 5	Entwurf beim Bau.	75
	5.1 Herausforderungen beim Entwurf	76
	Entwurf ist ein verzwicktes Problem	76
	Entwurf ist ein schludriger Prozess (auch wenn das Ergebnis stimmt)	77
	Beim Entwurf geht es um Kompromisse und Prioritäten	78
	Der Entwurf erfordert Beschränkungen	78
	Entwurf ist nicht deterministisch	78
	Der Entwurf als heuristischer Prozess	78
	Der Entwurf entwickelt sich evolutionär	79
	5.2 Schlüsselkonzepte für den Entwurf	79
	Das primäre Ziel: Zähmen der Komplexität	79
	Wünschenswerte Merkmale eines Entwurfs	82
	Entwurfsstufen	84
	5.3 Bausteine entwerfen: Heuristik	89
	Reale Objekte	90
	Konsistente Abstraktionen	91
	Kapseln von Implementierungsdetails	93
	Wann Vererbung den Entwurf vereinfacht	93
	Verbergen von Informationen	94
	Identifizieren von Bereichen, die sich wahrscheinlich ändern	100
	Lose Kopplung	103
	Entwurfsmuster	106
	Andere heuristische Verfahren	109
	Zusammenfassung der heuristischen Entwurfsverfahren	112
	Richtlinien zum Einsatz heuristischer Verfahren	112
	5.4 Entwurfstechniken	114
	Iteration	114
	Teile und herrsche	115
	Top-Down- und Bottom-Up-Entwurfsansätze	115

Experimentelle Prototypen	118
Teamwork beim Entwurf	119
Wie viel Entwurf ist genug?	120
Dokumentieren des Entwurfs.	121
5.5 Anmerkungen zu verbreiteten Methoden	123
Weiterführende Literatur.	124
Grundlagen des Softwareentwurfs.	124
Theorie des Softwareentwurfs.	125
Entwurfsmuster.	125
Allgemeiner Entwurf.	125
Standards.	126
Zusammenfassung	127

Kapitel 6

Vom Umgang mit der Arbeiterklasse.	129
6.1 Grundlagen für Klassen: abstrakte Datentypen.	130
Wann benötigen Sie einen abstrakten Datentyp? Ein Beispiel.	130
Die Vorteile abstrakter Datentypen.	131
Weitere Beispiele für ADTs.	133
Mehrfache Dateninstanzen mit ADTs auch außerhalb objektorientierter Umgebungen.	135
ADTs und Klassen.	137
6.2 Gute Klassenschnittstellen.	137
Gute Abstraktionen.	137
Gute Kapselung.	143
6.3 Entwurfs- und Implementierungsfragen.	148
Einbettung (»Hat ein«-Beziehungen).	148
Vererbung (»Ist ein«-Beziehungen).	149
Memberfunktionen und -daten.	155
Konstruktoren.	156
6.4 Gründe für das Erstellen einer Klasse.	158
Klassen, die Sie vermeiden sollten.	161
Zusammenfassung der Gründe für das Erstellen einer Klasse.	162
6.5 Sprachspezifische Unterschiede.	162
6.6 Jenseits von Klassen: Pakete.	163
Weiterführende Literatur.	165
Zusammenfassung.	166

Kapitel 7

Qualitätsmerkmale guter Routinen.	167
7.1 Gründe für das Erstellen von Routinen.	170
Operationen, die für eine eigene Routine zu einfach erscheinen.	172
Zusammenfassung der Gründe für das Erstellen einer Routine.	173
7.2 Entwurf auf der Routinenebene.	174
7.3 Gute Routinennamen.	177
7.4 Wie lang darf eine Routine sein?	179
7.5 Routinenparameter.	181

7.6	Besonderheiten beim Einsatz von Funktionen	187
	Sollten Sie eine Funktion oder eine Prozedur verwenden?	187
	Zurückgeben des Ergebniswerts bei Funktionen	188
7.7	Makro- und Inlineroutinen	189
	Alternativen zum Einsatz von Makroroutinen	190
	Inlineroutinen	191
	Zusammenfassung	192

: Kapitel 8

	Defensive Programmierung	195
8.1	Schutz vor ungültigen Eingabedaten	196
8.2	Sicherheitsabfragen	197
	Einen eigenen Mechanismus für Sicherheitsabfragen erstellen	199
	Richtlinien für den Einsatz von Sicherheitsabfragen	199
8.3	Techniken für die Fehlerbehandlung	202
	Robustheit und Korrektheit	205
	Berücksichtigen der Fehlerbehandlung im Entwurf	206
8.4	Ausnahmen	206
8.5	Dämmen Sie die Auswirkungen von Fehlern ein	212
	Die Beziehung zwischen Barrikaden und Sicherheitsabfragen	214
8.6	Fehlersuchhilfen	214
	Für die Entwicklungsversion gelten kaum Einschränkungen	214
	Fehlersuchhilfen - von Anfang an	215
	Angriff ist die beste Verteidigung: Offensive Programmierung	215
	Das Entfernen der Fehlersuchhilfen einplanen	216
8.7	Wie viel defensive Programmierung soll in der fertigen Version verbleiben?	218
8.8	Vorsicht bei der defensiven Programmierung!	220
	Weiterführende Literatur	221
	Zusammenfassung	222

Kapitel 9

	Pseudocode-Programmierung.	223
9.1	Bauphasen für Klassen und Routinen	223
	Schritte beim Bauen einer Klasse	224
	Schritte beim Bauen einer Routine	225
9.2	Pseudocode für Profis	225
9.3	Bauen von Routinen mithilfe des PPP	228
	Entwerfen der Routine	228
	Schreiben des Codes	233
	Überprüfen des Quelltextes	238
	Aufräumarbeiten	240
	Wiederholen Sie nötigenfalls einzelne Schritte	241
9.4	Alternativen zum PPP	241
	Zusammenfassung	243

Teil III**Kapitel 10****Variablen. 245****Der Einsatz von Variablen. 247**

10.1	Kleine Datenkunde.	248
	Datentypen - der Wissenstest	248
	Weiterführende Literatur zu Datentypen.	249
10.2	Variablendeklarationen vereinfachen.	249
	Implizite Deklarationen.	249
10.3	Richtlinien für das Initialisieren von Variablen.	250
10.4	Der Gültigkeitsbereich	254
	Referenzen auf Variablen lokalisieren.	254
	Nur eine tote Variable ist eine gute Variable: Halten Sie die	
	Lebensdauer von Variablen so kurz wie möglich	255
	Allgemeine Richtlinien zum Verkleinern des Gültigkeitsbereichs	258
	Anmerkungen zum Verkleinern des Gültigkeitsbereichs.	260
10.5	Persistenz	261
10.6	Zeitpunkt der Bindung	262
10.7	Beziehungen zwischen Datentypen und Steuerstrukturen.	263
10.8	Variablen nur für einen einzigen Zweck verwenden.	265
	Zusammenfassung	267

Kapiteln**Die Macht guter Variablennamen 269**

11.1	Was ist ein guter Name?.	269
	Das Wichtigste zuerst	270
	Ausrichtung auf das Problem	271
	Optimale Länge von Namen	272
	Der Einfluss des Gültigkeitsbereichs auf Variablennamen.	272
	Variablennamen, die für berechnete Werte stehen.	273
	Gegensatzpaare in Variablennamen.	274
11.2	Namen für bestimmte Datentypen.	274
	Namen für Schleifenindexe	274
	Namen für Statusvariablen.	276
	Namen für temporäre Variablen.	277
	Namen für boolesche Variablen.	278
	Namen für Aufzählungstypen.	279
	Namen für Konstanten.	280
11.3	Die Vorteile von Namenskonventionen.	280
	Konventionen - wozu?.	280
	Wann brauchen Sie eine Namenskonvention?.	281
	Stufen der Formalität	281
11.4	Informelle Namenskonventionen.	282
	Richtlinien für sprachunabhängige Konventionen.	282
	Richtlinien für sprachabhängige Konventionen.	285
	Mischen von Programmiersprachen	286
	Beispiele für Namenskonventionen.	287
11.5	Standardisierte Präfixe.	289
	Abkürzungen für benutzerdefinierte Typen.	289

Semantikpräfixe	290
Vorzüge von standardisierten Präfixen	291
11.6 Kurze, lesbare Namen	292
Allgemeine Richtlinien für Abkürzungen	292
Phonetische Abkürzungen	292
Einige Anmerkungen zu Abkürzungen	293
11.7 Schlechte Namen	295
Zusammenfassung	299

Kapitel 12

Grundlegende Datentypen	301
12.1 Grundlegendes zu Zahlen	301
12.2 Integer	303
12.3 Gleitkommazahlen	305
12.4 Zeichen und Strings	307
Strings in C	309
12.5 Boolesche Variablen	311
12.6 Aufzählungstypen	312
Was aber, wenn Ihre Sprache keine Aufzählungstypen kennt?	316
12.7 Benannte Konstanten	317
12.8 Arrays	319
12.9 Erstellen eigener Typen	321
Warum sind die Beispiele für selbstdefinierte Typen in Pascal und Ada geschrieben?	324
Richtlinien für das Erstellen selbstdefinierter Typen	324
Zusammenfassung	327

Kapitel 13

Ungewöhnliche Datentypen	329
13.1 Strukturen	329
13.2 Zeiger	333
Die Funktionsweise von Zeigern	333
Allgemeine Hinweise zu Zeigern	335
Zeiger in C++.	343
Zeiger in C	344
13.3 Globale Daten	345
Häufige Probleme mit globalen Daten	346
Einsatzzwecke für globale Daten	348
Benutzen Sie globale Daten nur als letzte Möglichkeit	349
Zugriffsroutinen anstelle globaler Daten	350
Vermindern der mit globalen Variablen verbundenen Risiken	353
Weiterführende Literatur	354
Zusammenfassung	355

Teil IV

Anweisungen	357
------------------------------	------------

Kapitel 14

Sequentieller Ablauf	359
14.1 Anweisungen mit zwingender Reihenfolge	359
14.2 Anweisungen, deren Reihenfolge keine Rolle spielt	362

	Quelltext von oben nach unten lesen	363
	Verwandte Anweisungen zusammenfassen	364
	Zusammenfassung	365
Kapitel 15	Bedingungsausdrücke	367
	15.1 if-Anweisungen	367
	Einfache /f-t/ien-Anweisungen	367
	if-then-else-Ketten	370
	15.2 case-Anweisungen	373
	Die sinnvollste Fallreihenfolge	373
	Tipps für den Einsatz von case-Anweisungen	374
	Zusammenfassung	378
Kapitel 16	Schleifensteuerung	379
	16.1 Auswählen des Schleifentyps	379
	Einsatzzwecke für w/zi/e-Schleifen	380
	Einsatzzwecke für Schleifen mit <i>Exit</i> -Anweisung	381
	Einsatzzwecke für/or-Schleifen	384
	Einsatzzwecke für/oreac/i-Schleifen	384
	16.2 Steuern der Schleife	384
	Einsprung in die Schleife	385
	Die Schleifenmitte	387
	Verlassen der Schleife	388
	Überprüfen der Endpunkte	393
	Schleifenindexe	394
	Wie lang sollte eine Schleife sein?	396
	16.3 Schleifen einfach von innen nach außen anlegen	397
	16.4 Zusammenhänge zwischen Schleifen und Arrays	399
	Zusammenfassung	401
Kapitel 17	Ungewöhnliche Steueranweisungen	403
	17.1 Mehrere Rückkehranweisungen in einer Routine	403
	17.2 Rekursion	405
	Ein Beispiel für Rekursion	406
	Tipps für den Einsatz der Rekursion	408
	17.3 <i>goto</i>	410
	Was spricht gegen <i>goto</i> ?	410
	Was spricht für <i>goto</i> ?	411
	Dieser Glaubenskrieg ist sinnlos	412
	Fehlerbehandlung und <i>goto</i>	413
	<i>goto</i> und gemeinsam genutzter Code im <i>else-Zweig</i>	418
	Zusammenfassung der Richtlinien für den Einsatz von <i>goto</i> -Anweisungen	420
	17.4 Bewertung der ungewöhnlichen Steuerstrukturen	421
	Weiterführende Literatur	421
	Zusammenfassung	423

Kapitel 18	Tabellengesteuerte Methoden.	425
18.1	Allgemeine Überlegungen zum Einsatz tabellengesteuerter Methoden	425
	Voraussetzungen für den Einsatz tabellengesteuerter Methoden . . .	426
18.2	Direkter Zugriff.	427
	Ein Beispiel: Anzahl der Tage pro Monat	427
	Beispiel: Versicherungsbeiträge.	428
	Beispiel für ein flexibles Nachrichtenformat	430
	Konstruieren eines Suchschlüssels.	437
18.3	Indizierter Zugriff	438
18.4	Stufenweiser Zugriff.	440
18.5	Andere Beispiele für die Tabellensuche	442
	Zusammenfassung	443
Kapitel 19	Allgemeine Gesichtspunkte der Ablaufsteuerung.	445
19.1	Boolesche Ausdrücke.	445
	<i>true</i> und <i>false</i> in booleschen Abfragen.	445
	Vereinfachen komplizierter Ausdrücke.	447
	Formulieren Sie boolesche Ausdrücke positiv.	449
	Boolesche Ausdrücke durch Klammern strukturieren.	450
	So werden boolesche Ausdrücke ausgewertet	452
	Schreiben Sie numerische Vergleiche im Sinn der Zahlengeraden . . .	453
	Richtlinien für den Vergleich mit 0.	454
	Häufige Probleme mit booleschen Ausdrücken.	455
19.2	Verbundanweisungen (Blöcke).	456
19.3	Null-Anweisungen.	457
19.4	Tiefe Verschachtelungen beherrschen	458
	Zusammenfassung: Techniken zum Auflösen tiefer Verschachtelung .	467
19.5	Strukturierte Programmierung.	467
	Die drei Komponenten der strukturierten Programmierung	468
19.6	Steuerstrukturen und Komplexität	470
	Welche Bedeutung hat die Komplexität?.	470
	Allgemeine Richtlinien für das Verringern der Komplexität	471
	Andere Arten von Komplexität	472
	Zusammenfassung	473
Teil V	Verbessern des Programmcodes.	475
Kapitel 20	Die Qualität der Software.	477
20.1	Merkmale der Softwarequalität	477
20.2	Techniken zur Verbesserung der Softwarequalität	480
	Der Entwicklungsprozess.	481
	Zielsetzung.	482
20.3	Relative Effektivität der Techniken.	483
	Prozentsatz der gefundenen Fehler.	483
	Kosten der Fehlersuche.	485
	Kosten der Fehlerbeseitigung	485

20.4	Zeitpunkt für die Qualitätssicherung	486
20.5	Das Grundprinzip der Softwarequalität	487
	Weiterführende Literatur	489
	Wichtige Standards	489
	Zusammenfassung	490

Kapitel 21

	Teamwork beim Bau	491
21.1	Teamworkentwicklungsverfahren	491
	Teamwork beim Bau ergänzt andere Techniken der Qualitätssicherung	492
	Teamwork beim Bau verbreitet Firmenkultur und Programmiererfahrung	494
	Gemeinsame Verantwortung betrifft alle Arten des Teamworks beim Bau	494
	Teamwork hilft vor und nach dem Bau	495
21.2	Paarprogrammierung	495
	Faktoren für erfolgreiche Paarprogrammierung	495
	Vorteile der Paarprogrammierung	496
21.3	Formelle Inspektionen	497
	Welche Ergebnisse können Sie von Inspektionen erwarten?	497
	Die Rollen bei der Inspektion	498
	Prinzipieller Ablauf einer Inspektion	499
	Das Ego und die Inspektion	502
	Inspektionen bei diesem Buch	503
	Zusammenfassung zum Thema Inspektionen	503
21.4	Andere Teamworktechniken	504
	Walk-throughs	504
	Lesen des Quellcodes	506
	Vorfürhungen	507
	Vergleich von Teamworktechniken beim Softwarebau	507
	Weiterführende Literatur	508
	Paarprogrammierung	508
	Inspektionen	509
	Wichtige Standards	509
	Zusammenfassung	509

Kapitel 22

	Entwicklertests	511
22.1	Entwicklertests und die Softwarequalität	512
	Testen während des Baus	514
22.2	Empfohlene Ansätze für Entwicklertests	515
	Tests zuerst oder zuletzt?	516
	Einschränkungen von Entwicklertests	516
22.3	Tipps und Tricks zum Testen	517
	Unvollständiges Testen	517
	Strukturierter Basistest	518
	Test des Datenflusses	521
	Äquivalenzaufteilung	524

	Fehler wittern	525
	Randbedingungen analysieren	525
	Bösartige Daten.	526
	Gutartige Daten	527
	Verwenden Sie Testfälle, die sich leicht von Hand nachvollziehen lassen.	528
22.4	Typische Fehler.	528
	Welche Klassen enthalten die meisten Fehler?.	528
	Eine Klassifizierung der Fehler.	529
	Anteil der beim Bau entstehenden Fehler.	531
	Wie viele Fehler sind zu erwarten?.	532
	Fehler beim Testen.	533
22.5	Testwerkzeuge	534
	Ein Gerüst zum Testen einzelner Klassen.	534
	Ergebnisvergleiche.	536
	Datengeneratoren für Tests.	536
	Coveragemonitore.	537
	Datenrekorder/Protokollierung	537
	Symbolische Debugger.	538
	System Perturbers - Störenfriede im System.	538
	Fehlerdatenbanken	539
22.6	So verbessern Sie Ihre Testmethoden.	539
	Testplanung	539
	Wiederholte Tests (Regressionstests).	540
	Automatisierte Tests.	540
22.7	Testdaten speichern.	541
	Persönliche Testaufzeichnungen.	541
	Weiterführende Literatur.	542
	Testen.	542
	Testgerüste.	543
	Erstellen von Testfällen.	543
	Wichtige Standards.	543
	Zusammenfassung	544
	Debugging.	547
23.1	Debugging im Überblick.	547
	Debugging und die Qualität von Software.	548
	Das Debuggingtempo.	548
	Sehen Sie Fehler als Chancen.	549
	Eine ungeeignete Methode.	550
23.2	So finden Sie einen Fehler.	552
	Die wissenschaftliche Methode.	552
	Tipps für das Auffinden von Fehlern.	556
	Syntaxfehler.	561
23.3	Beseitigen von Fehlern.	562
23.4	Debugging und Psychologie.	566

Kapitel 23

Psychologische Scheuklappen und die Blindheit beim Debugging . . .	567
Der Nutzen des »Psychologischen Abstands«	568
23.5 Debuggingtools	569
Quelltextvergleicher	569
Compilerwarnungen	569
Erweiterte Syntax- und Logikprüfung	570
Profiler	570
Testgerüste	570
Debugger	570
Weiterführende Literatur	573
Zusammenfassung	574

Kapitel 24

Refaktorisieren	575
24.1 Erscheinungsformen der Evolution von Software	576
Philosophie der Softwareevolution	576
24.2 Einführung in das Refaktorisieren	577
Anlässe zum Refaktorisieren	577
Was »Refaktorisieren« nicht bedeutet	583
24.3 Konkrete Refaktorisierungsverfahren	583
Refaktorisierungsverfahren auf Datenebene	584
Refaktorisierungsverfahren auf Anweisungsebene	585
Refaktorisierungsverfahren auf Routinenebene	586
Refaktorisierungsverfahren für die Klassenimplementierung	587
Refaktorisierungsverfahren für die Klassenschnittstelle	588
Refaktorisierungsverfahren auf Systemebene	589
24.4 Sicher refaktorisieren	592
Gefahren beim Refaktorisieren	594
24.5 Refaktorisierungsstrategien	595
Weiterführende Literatur	598
Zusammenfassung	598

Kapitel 25

Strategien zur Codeoptimierung	599
25.1 Leistung im Überblick	600
Qualitätsmerkmale und Leistung	600
Leistung und Codetuning	600
25.2 Einführung in das Codetuning	603
Das Pareto-Prinzip	604
Ammenmärchen	605
Wann sollte Tuning durchgeführt werden?	608
Compileroptimierungen	609
25.3 Schnecken und Bleigewichte	610
Häufige Leistungsbremsen	610
Relativer Aufwand häufig vorkommender Operationen	613
25.4 Messungen	615
Messungen müssen genau sein	616
25.5 Iteration	617
25.6 Zusammenfassung: Ansätze bei der Codeoptimierung	618

	Weiterführende Literatur	618
	Leistung	619
	Algorithmen und Datentypen	619
	Zusammenfassung	620
Kapitel 26	Techniken zur Codeoptimierung.	623
26.1	Logik	624
	Beenden Sie Abfragen, wenn das Ergebnis klar ist	624
	Abfragen nach Häufigkeit anordnen	625
	Leistung der Vergleichsoperationen in ähnlichen Logikstrukturen .. .	627
	Ersetzen komplizierter Ausdrücke durch Suchtabellen.	628
	Späte Auswertung	629
26.2	Schleifen	629
	Verzweigungen beseitigen	630
	Fusion von Schleifen	631
	Aufrollen	632
	Die Arbeit in Schleifen verringern	634
	Wächterklauseln	635
	Vielbeschäftigte Schleifen gehören an die innerste Position	636
	Verringern der Stärke	637
26.3	Datenumwandlungen	638
	Verwenden Sie Integer anstelle von Gleitkommazahlen	638
	Verringern Sie die Anzahl der Dimensionen eines Arrays so weit wie möglich	639
	Verringern Sie die Verweise auf Arrays.	640
	Einsatz zusätzlicher Indexe	640
	Caching	641
26.4	Ausdrücke	643
	Ausnutzen algebraischer Identität	643
	Verringern der Stärke	643
	Initialisierung während der Kompilierung	645
	Haben Sie ein wachsames Auge auf Systemroutinen.	646
	Verwenden Sie für Konstanten den korrekten Datentyp.	647
	Ergebnisse vorab berechnen	648
	Beseitigen wiederholter Berechnungen.	651
26.5	Routinen	652
	Inlineroutinen	652
26.6	Maschinennahe Sprachen	653
26.7	Je mehr sich die Dinge ändern, desto mehr bleiben sie gleich	656
	Weiterführende Literatur.	657
	Zusammenfassung	658

Teil VI	Faktoren für das Gesamtsystem	.659
Kapitel 27	Wie die Programmgröße den Bau beeinflusst	.661
	27.1 Kommunikation und Programmgröße	.661
	27.2 Typische Projektgrößen	.662
	27.3 Projektgröße und Fehler	.663
	27.4 Projektgröße und Produktivität	.665
	27.5 Projektgröße und Entwicklungstätigkeiten	.665
	Abhängigkeit des Anteils der Tätigkeiten von der Projektgröße	.666
	Programme, Produkte, Systeme und Systemprodukte	.667
	Methodologie und Projektgröße	.668
	Weiterführende Literatur	.670
	Zusammenfassung	.671
Kapitel 28	Entwicklungsmanagement	.673
	28.1 Fördern guter Programmierung	.674
	Überlegungen beim Festlegen von Standards	.674
	Techniken zum Fördern eines guten Programmierstils	.674
	Die Rolle dieses Buchs	.676
	28.2 Konfigurationsmanagement	.676
	Was ist Konfigurationsmanagement?	.676
	Änderungen an Anforderungen und Entwurf	.678
	Änderungen im Quellcode	.680
	Versionen der verwendeten Tools	.680
	Computerkonfigurationen	.681
	Sicherungsplan	.681
	Weiterführende Literatur zum Konfigurationsmanagement	.682
	28.3 Abschätzen eines Zeitplans für den Bau	.683
	Ansätze für die Einschätzung	.683
	Schätzen des Aufwands für den Bau	.685
	Einflüsse auf den Zeitplan	.686
	Schätzung und Kontrolle	.687
	Was tun, wenn Sie hinter dem Plan liegen?	.687
	Weiterführende Literatur zum Abschätzen von Softwareprojekten	.689
	28.4 Messungen	.689
	Weiterführende Literatur zu Softwaremessungen	.691
	28.5 Programmierer sind auch Menschen	.692
	Wie verbringen Programmierer ihre Zeit?	.692
	Schwankungen in Leistung und Qualität	.693
	Religiöse Debatten	.694
	Der Arbeitsplatz	.695
	Weiterführende Literatur zu Programmierern als Menschen	.697
	28.6 Managen Sie den Manager	.698
	Weiterführende Literatur zum Entwicklungsmanagement	.698
	Wichtige Standards	.699
	Zusammenfassung	.699

Kapitel 29	Integration.	701
27.1	Die Bedeutung der Integrationsmethode.	701
29.2	Integrationsfrequenz - synchron oder inkrementell?	703
	Synchrone Integration.	703
	Inkrementelle Integration.	704
29.3	Strategien der inkrementellen Integration.	706
	Top-Down-Integration.	706
	Bottom-Up-Integration.	709
	Sandwichintegration.	710
	Risikoorientierte Integration.	711
	Merkmalsorientierte Integration.	711
	T-förmige Integration.	713
	Zusammenfassung der Integrationsansätze.	714
29.4	Täglicher Buildvorgang und Smoketest	714
	Für welche Projektarten eignet sich der tägliche Buildvorgang?	718
	Kontinuierliche Integration.	719
	Weiterführende Literatur.	720
	Zusammenfassung.	721
Kapitel 30	Programmiertools.	723
30.1	Entwurfstools.	724
30.2	Quellcodetools.	724
	Editoren.	724
	Analyse der Codequalität.	727
	Refaktorisieren von Quellcode.	728
	Versionskontrolle.	729
	Datenverzeichnisse.	729
30.3	Tools für ausführbare Programme.	729
	Programmerstellung.	730
	Debugging.	732
	Testwerkzeuge.	733
	Codetuning.	733
30.4	Toolorientierte Umgebungen.	734
30.5	Programmiertools selbst erstellen.	734
	Projektspezifische Tools.	735
	Skripts.	735
30.6	Tools: Wunsch und Wirklichkeit.	736
	Weiterführende Literatur.	737
	Zusammenfassung.	738
Teil VII	Softwarehandwerk.	739
Kapitel 31	Layout und Stil.	741
31.1	Grundlagen des Layouts.	742
	Extreme im Layout.	742
	Der Grundsatz der Formatierung.	744

	Interpretationen eines Programms.	744
	Was ist gutes Layout wert?	745
	Layout als Religion	747
	Ziele guten Layouts	747
31.2	Layouttechniken	748
	Zwischenräume	748
	Klammern	749
31.3	Layoutstile	750
	Reine Blöcke	750
	Simulation reiner Blöcke	751
	Begrenzen von Blöcken mit <i>begin/end</i> - oder Klammerpaaren	753
	Endlinelayout	754
	Welcher Stil ist der beste?	756
31.4	Layout von Steuerstrukturen	756
	Feinheiten der Formatierung von Anweisungsblöcken	757
	Andere Gesichtspunkte	758
31.5	Layout einzelner Anweisungen	764
	Anweisungslänge	764
	Klarheit durch Leerzeichen	764
	Formatierung umbrochener Zeilen	765
	Nur eine Anweisung pro Zeile	769
	Layout für Datendeklarationen	772
31.6	Layout von Kommentaren	774
31.7	Layout von Routinen	777
31.8	Layout von Klassen	778
	Layout von Klassenschnittstellen	779
	Layout von Klassenimplementierungen	779
	Layout von Dateien und Programmen	782
	Weiterführende Literatur	785
	Zusammenfassung	785
	Selbstdokumentierender Code.	787
32.1	Externe Dokumentation	787
32.2	Programmierstil als Dokumentation	788
32.3	Kommentieren oder nicht kommentieren, das ist hier die Frage	791
32.4	Schlüsselemente sinnvoller Kommentare	795
	Kommentartypen	796
	Sinnvolle Kommentierung	798
	Die optimale Zahl von Kommentaren	801
32.5	Techniken der Kommentierung	802
	Kommentieren einzelner Zeilen	802
	Kommentieren von Absätzen im Code	804
	Kommentieren von Datendeklarationen	811
	Kommentieren von Steuerstrukturen	813
	Kommentieren von Routinen	814
	Kommentieren von Klassen, Dateien und Programmen	819

Kapitel 32

32.6	IEEE-Standards	822
	Standards für die Softwareentwicklung	823
	Standards für die Qualitätssicherung von Software	823
	Managementstandards	823
	Überblick über Standards	823
	Weiterführende Literatur	824
	Zusammenfassung	826

Kapitel 33

	Persönlicher Stil	827
33.1	Welche Rolle spielt der persönliche Stil?	828
33.2	Intelligenz und Bescheidenheit	829
33.3	Neugier	830
33.4	Ehrlichkeit	834
33.5	Kommunikation und Zusammenarbeit	836
33.6	Kreativität und Disziplin	837
33.7	Faulheit	838
33.8	Dinge, die gar nicht so wichtig sind	838
	Beharrlichkeit	839
	Erfahrung	839
	Der Überflieger	840
33.9	Gewohnheiten	"..." 841
	Weiterführende Literatur	842
	Zusammenfassung	843

Kapitel 34

	Der Programmierer als Handwerker	845
34.1	Krieg der Komplexität!	845
34.2	Arbeitsstil - Sie haben die Wahl	847
34.3	Erst der Mensch, dann der Computer	849
34.4	Programmieren <i>in die</i> Sprache und <i>in der</i> Sprache	851
34.5	Konventionen sind Konzentrationshilfen	852
34.6	Programmieren mit der Terminologie des Praxisproblems	853
	Die Unterteilung eines Programms in Abstraktionsebenen	854
	Arbeitstechniken für die niedrigere Ebene des Problembereichs	855
34.7	Achtung, Steinschlag!	856
34.8	Iteration, wieder und wieder	858
34.9	Du sollst trennen Software und Religion	859
	Das Softwareorakel	859
	Eklektizismus	860
	Probieren geht über Studieren	861
	Zusammenfassung	862

Kapitel 35

	Weiterführende Literatur	863
35.1	Informationen über den Bau von Software	864
35.2	Weitere Aspekte der Softwareentwicklung	865
	Software im Allgemeinen	865
	Software-Engineering im Überblick	866
	Andere kommentierte Bibliografien	866

35.3	Fachzeitschriften	867
	Zeitschriften für den Massenmarkt	867
	Wissenschaftliche Zeitschriften	867
	Fachzeitschriften zu Spezialgebieten	868
35.4	Eine Leseliste für den Softwareentwickler	868
	Bücher für Anfänger	868
	Bücher für Fortgeschrittene	869
	Bücher für leitende Entwickler	869
35.5	Mitgliedschaft in einer Berufsorganisation	870
	 Bibliografie	 871
	 Index	 891
	 Der Autor	 909