

Inhalt

1	Einleitung	1
1.1	Softwarequalität betrifft viele	1
1.2	Für wen dieses Buch gemacht ist	1
1.3	Was Sie von diesem Buch erwarten können	2
1.4	Das Abenteuer von Q	3
1.5	Themen und Anspruch	3
1.5.1	Themenauswahl und Gewichtung	4
1.5.2	Die Reihenfolge der Themen	5
1.6	Bedeutung von Softwarequalität	6
1.7	Wie Q zur Softwarequalität kam	8
2	Grundkonzepte	11
2.1	Qualitätsorganisation und Terminologie	12
2.2	Kosten und Nutzen von Softwarequalität	16
2.3	Qualitätsbeauftragte	18
2.4	Eine Vision: Total Quality Management	21
2.5	Grundbegriffe des Testens	23
2.6	Normen und Standards	27
2.7	Qualitätsaspekte, -anforderungen und Qualitätsmodelle	30
3	Erfahrungen systematisch nutzen	39
3.1	Qualitätsnetzwerke und Qualitätszirkel	40
3.2	Leichtgewichtige Dokumentation von Erfahrungen	42
3.3	Organisation der Erfahrungsverwaltung	45
3.4	Herausforderungen und Chancen für Erfahrungsnutzung	47
3.5	Networking in Organisationen und auf Tagungen	49

4	Messen von Softwarequalität	51
4.1	Wozu messen und konkretisieren?	52
4.2	Softwremetriken	54
4.2.1	Grundlagen	55
4.2.2	Was Softwremetriken messen	55
4.2.3	Bezug zwischen Metrik und Qualitätsaspekt	57
4.2.4	Skalen für die Resultate der Metriken	57
4.3	Diskussion bekannter Softwremetriken	59
4.3.1	Lines of code: Der Teufel steckt im Detail	59
4.3.2	Zyklomatische Komplexität von McCabe	61
4.3.3	Halstead Software Science	66
4.3.4	Weitere Metriken: ein Ausblick	69
4.4	Metriken nach Maß: QQM	70
4.4.1	Von Zielen zu Fragen zu Metriken – und zurück	70
4.4.2	Zielorientiertes Messen und Bewerten	72
4.4.3	Zielfacetten schärfen den Blick	73
4.4.4	Messung vorbereiten mit Abstraction Sheets	75
4.4.5	Besonderheiten bei Messung und Auswertung	79
4.5	Projektfortschritt messen mit Quality Gates	80
5	Systematisches Testen	83
5.1	Vorüberlegungen	83
5.1.1	Testvorbereitung	83
5.1.2	Vollständig testen?	85
5.1.3	Woraus ein Testfall besteht	86
5.1.4	Testfälle dokumentieren	87
5.1.5	Testfälle ermitteln: eine Strategie	88
5.1.6	Hintergrund von Fehlern	90
5.1.7	Übersicht: Black-Box-Test und Glass-Box-Test	91
5.2	Black-Box-Tests aus der Spezifikation	91
5.2.1	Minimalforderung und Effizienzprinzip	92
5.2.2	Äquivalenzklassenmethode	94
5.2.3	Grenzwertanalyse	96
5.2.4	Spezifikationsabdeckung optimieren	96
5.2.5	Klassifikationsbaummethode	99
5.2.6	Zustandsbasiertes Testen	101
5.2.7	Testablauf dokumentieren	105
5.3	Sollwerte aus der Spezifikation	106

5.4	Glass-Box: Testen nach der Codestruktur	108
5.4.1	Maße für Codeüberdeckung	109
5.4.2	Interpretation von Überdeckungsmaßen	111
5.4.3	Objektorientierung und Glass-Box-Test	112
5.5	Testfälle für spezielle Qualitätsaspekte	114
5.5.1	Testfälle in Form von Code	116
5.5.2	Granularität und Reihenfolge von Prüflingen	117
5.5.3	Stresstest, Recovery und Security Tests	119
5.6	Hilfsmittel und Werkzeuge für das Testen	119
5.6.1	Debuggen ist nicht Testen	119
5.6.2	Standardhilfsmittel: Testrahmen	120
5.6.3	Werkzeuge für Glass-Box-Test	120
5.6.4	Sonstige Hilfsmittel und Werkzeuge	122
5.7	Testen von grafischen Oberflächen	122
5.7.1	Sackgasse: System als Ganzes	123
5.7.2	Capture/Replay-Tools	124
6	Usability Engineering	127
6.1	Software und Bedienbarkeit	128
6.2	Usability als Qualitätsaspekt	128
6.2.1	Gute Bedienoberflächen und Qualitätsaspekte	129
6.2.2	Usability definiert sich über Anforderungen	131
6.3	Aspekte der Benutzerfreundlichkeit nach ISO 9241	133
6.4	Bedienbarkeit messen	135
6.5	Konstruktives Usability Engineering	135
6.5.1	Aufgaben im Usability Engineering	136
6.5.2	Kernaufgaben in der Anforderungsklä rung	137
6.5.3	Aktivitäten in Entwurf und Entwicklung	139
6.5.4	Acht Goldene Regeln nach Shneiderman	140
6.6	Experten-Evaluationen	141
7	Reviews und Inspektionen	145
7.1	Rollen und Ablauf	146
7.2	Hilfsmittel	151
7.3	Aufwand und Nutzen	155
7.4	Varianten von Reviews	157

8	Formale Verfahren	159
8.1	Prädikatenkalkül und formale Beweise	160
8.1.1	Grundvorgehen und Basiselemente	161
8.1.2	Voraussetzungen aus Anforderung ableiten	162
8.1.3	Verzweigung als Anweisungsart	164
8.1.4	Schleifeninvarianten	165
8.2	Verschiedene Spezifikationsstile	168
8.3	Spezifizieren und Beweisen mit Modellen	170
8.3.1	Ampelanlage als Petrinetz-Beispiel	171
8.3.2	Beweise auf Petrinetzen	174
8.4	Diskussion formaler Techniken	176
9	Konstruktive Qualitätssicherung	177
9.1	Analytisch, organisatorisch, konstruktiv	177
9.2	Maßnahmen, bevor ein Problem auftritt	179
9.2.1	Bewährte Verfahren	180
9.2.2	Bewährte Bestandteile	181
9.2.3	Bewährte Strukturen	184
9.3	Beispiel Cleanroom: Fehler vermeiden	184
10	Agile Softwareentwicklung und Qualität	187
10.1	Die kurze Geschichte der agilen Softwareentwicklung	188
10.2	Extreme Programming im Überblick	191
10.3	Testgetriebene Entwicklung in XP	198
10.3.1	Terminologie und Testarten	198
10.3.2	Testautomatisierung ist unverzichtbar	199
10.3.3	Testcode ist seltener fehlerhaft	200
10.3.4	Test First: Testen vor Codieren	201
10.3.5	Auswirkungen von Test First	203
10.3.6	Besserer Produktionscode durch Test First	204
10.4	Die Rolle der Softwarequalität in XP	205
10.5	Scrum	206
10.6	Lean Software Development	212
10.6.1	Vom Toyota Production System zur Softwareentwicklung ..	212
10.6.2	Die Grundkonzepte von Lean	213
10.6.3	Auswirkungen auf die Softwarequalität	218

10.7	Kanban	219
10.7.1	Arbeitsabläufe visualisieren	219
10.7.2	Pull statt Push: Aufgabenvolumen begrenzen, Durchlaufzeit verkürzen	220
10.7.3	Ausblick: Kanban für Fortgeschrittene	222
10.8	Zusammenfassung	224
11	Das Abenteuer geht weiter	227
11.1	Rückblick	227
11.2	Was es noch zu erkunden gibt	229
11.3	Wann man aufhören soll	229
	Literaturverzeichnis	233
	Abkürzungen	241
	Index	243