

Inhaltsverzeichnis

1 Einführung	1
2 Grundlagen	5
2.1 Software-Wartung	5
2.1.1 Gibt es ein Wartungsproblem?	8
2.2 Begriffe	9
2.3 Imperative Programmiersprachen	15
2.3.1 Variablen	16
2.3.2 Kontrollstrukturen	19
2.4 Übungsaufgaben	26
3 Programmverstehen	27
3.1 Die kognitiven Aspekte des Programmverstehens	28
3.1.1 Bottom-Up Verstehen	29
3.1.2 Top-Down Verstehen	29
3.1.3 Opportunistisches Verstehen	30
3.2 Die Aufbereitung des Programmtextes	30
3.3 Alternative Darstellungsarten	33
3.3.1 Systemüberblick	34
3.3.2 Daten und Datenstrukturdarstellungen	34
3.3.3 Der Kontrollfluß	36
3.3.4 Der Datenfluß	40
3.3.5 Darstellung anderer Eigenschaften	40
3.4 Übungsaufgaben	43
4 Metriken	45
4.1 Die Größe von Software-Systemen	46

4.1.1	Die Größe eines Programms in Zeilen	46
4.1.2	Die Größe eines Programms nach Halstead	46
4.1.3	Das Vokabular eines Programms	48
4.1.4	Das Volumen eines Programms	48
4.2	Metriken zur Erfassung von Daten und Datenstrukturen	49
4.3	Metriken der Ablauflogik	50
4.3.1	Zyklomatische Komplexität	50
4.3.2	GOTO-Metriken	55
4.4	Function-Points	56
4.4.1	Die Berechnung der Function-Points	58
4.4.2	Alternative Verwendung der Function-Points	60
4.5	Kann man Wartbarkeit messen?	61
4.5.1	Der Reifegrad	63
4.5.2	Source-Code	64
4.5.3	Dokumentation	67
4.5.4	Erfahrungen	68
4.6	Übungsaufgaben	70
5	Sprachkonversion	73
5.1	Anweisungslokale Konversion	73
5.2	Konversion durch Modellbildung	75
5.3	Repräsentationsformalismen	76
5.3.1	Spezifikationssprachen	77
5.3.2	Narrow-spectrum Sprachen	77
5.3.3	Wide-spectrum Sprachen	78
5.3.4	Broad-spectrum Sprachen	79
5.4	Ein Beispielsystem	79
5.5	Eine allgemeinere Sichtweise der Sprachkonversion	82
5.6	Übungsaufgaben	83
6	Restrukturierung	85
6.1	Primprogramme	85
6.2	Zusammengesetzte und strukturierte Programme	88
6.2.1	Einfache Erzeugung strukturierter Programme — Satz der Strukturierten Programmierung	88

6.2.2	Erzeugung effizienter strukturierter Programme	92
6.3	Übungsaufgaben	94
7	Wiederverwendung	97
7.1	Motivation	97
7.2	Das Umfeld	99
7.3	Ausgewählte Aspekte der Wiederverwendung	100
7.3.1	Wiederverwendung im Software-Entwicklungsprozeß	100
7.3.2	Qualitätskriterien für Bausteine	102
7.3.3	Beschreibungs- und Suchverfahren	103
7.4	Beispiele für erfolgreiche Wiederverwendung	106
7.4.1	Funktionsbibliotheken	106
7.4.2	Software-Schablonen	106
7.4.3	Generatoren	107
7.4.4	Abstrakte Datentypen	108
7.4.5	Objektorientierung	108
7.5	Übungsaufgaben	109
8	Migration in die objektorientierte Welt	111
8.1	Wrapper-Technik	112
8.1.1	Kommunikationsmöglichkeiten der Basissprachen	112
8.1.2	Garbage-Collection und Speicherkompaktierung	113
8.1.3	Synchronisation der Objektlebenszeiten	113
8.1.4	Cross-System-Konsistenz	114
8.2	Transformation über syntaktische Muster	114
8.3	Kombinierter Top-Down/Bottom-Up-Ansatz	117
8.3.1	Design Recovery	119
8.3.2	Object Mapping	123
8.3.3	System Transformation	124
8.4	Vergleich und Bewertung	125
8.5	Übungsaufgaben	126
9	Bestandsanalyse, Kosten/Nutzen, Risiken	129
9.1	Bestandsanalyse	129
9.1.1	Informationserhebung	130

9.1.2	Informationsverdichtung	134
9.1.3	Informationsauswertung	136
9.1.4	Eine alternative Methode der Bestandsanalyse	141
9.2	Kosten/Nutzen-Analyse	144
9.2.1	Der Geschäftswert	144
9.2.2	Kosten	145
9.2.3	Kosten/Nutzen-Vergleiche	146
9.3	Risiken des Reengineering	150
9.4	Übungsaufgaben	151
10	Der Jahrtausendwechsel	153
10.1	Ursachen und Auswirkungen	154
10.2	Das Jahr-2000-Projekt	155
10.2.1	Bestandsanalyse	156
10.2.2	Problemanalyse	158
10.2.3	Design	159
10.2.4	Abhängigkeiten und Umstellungsreihenfolge	161
10.2.5	Automatisierungspotential	165
10.3	Weitere Informationen	166
10.4	Übungsaufgaben	167
A	Lösungshinweise zu den Übungsaufgaben	169
B	Glossar	173
C	Grundlegende und weiterführende Bücher	177
	Literaturverzeichnis	181
	Sachverzeichnis	193