

1 Einführung und Grundlagen	1
1.1 Wer verträgt wieviel Abstraktion?	1
1.2 Methoden der Software-Entwicklung	4
1.3 Prinzipien der Software-Entwicklung	6
2 Die strukturierte Systemanalyse (Definition der Anforderungen)	9
2.1 Die strukturierte Systemanalyse	9
2.2 Techniken und Hilfsmittel der strukturierten Systemanalyse	10
2.3 Das Auflösen (Konkretisieren) der Datenflussdiagramme	11
2.4 Die Berücksichtigung der Prozessdynamik	16
2.5 Themenspezifische Literaturhinweise	20
3 Der strukturierte Systementwurf	21
3.1 Herleitung von hierarchisch strukturierten Softwaresystemen	21
3.2 Anforderungen an einen guten System-Entwurf	24
4 Praxisbericht: Arbeiten mit Analyse-, Design- und Programmiertools	28
5 Praxisbericht: Strukturierte Analyse und objektorientierte Entwicklung - Eine Brücke zwischen den Methoden	40
6 Die Daten-Modellierung als Voraussetzung für die Automatisierung von Informationsprozessen	44
7 Die Datenmodellierung mit Hilfe des Entity-Relationship-Modells (ERM)	49
7.1 Klassifizierung der Objekte	50
7.2 Festlegung der relevanten Eigenschaften	52
7.3 Festlegung der Identifizierung	56
7.4 Beschreibung der sachlogischen Zusammenhänge zwischen zwei Objekttypen	60
7.5 Beschreibung der sachlogischen Zusammenhänge innerhalb eines Objekttyps	72
7.6 Modellierung in Grenzfällen des Entity-Relationship-Modells	75
7.7 Qualitätssicherung von Entity-Relationship-Modellen	86
7.8 Nutzen der Datenmodellierung	99
7.9 Themenspezifische Literaturhinweise	100
8 Grundzüge der Objektorientierung	101
8.1 Nochmals: Die Prinzipien der Software-Entwicklung	101
8.2 Grundbegriffe der objektorientierten Programmierung- ein erster Überblick	102
8.3 Die Mechanismen der objektorientierten Programmierung	104

9 Grundlagen der objektorientierten Herangehensweisen	107
9.1 Gegenüberstellung von strukturierten und objektorientierten Methoden	107
9.2 Der Lebenszyklus objektorientierter Software	108
9.3 Die objektorientierte Analysephase	108
9.4 Die Designphase	114
9.5 Test- und Wartungsphase	115
9.6 Modellierungstechniken	115
9.7 Objekte und Nachrichten	117
9.8 Klassen und Strukturen	119
9.9 Die Notwendigkeit mehrerer Modelle	120
9.10 Chancen und Risiken der Objektorientierung	122
9.11 Entwicklung der Unified Modeling Language (UML)	124
9.12 Modellierungstechniken gemäß Rumbaugh (OMT)	124
9.13 Der OMT-Entwicklungsprozeß	126
9.14 Systemmodellierung nach Coad	128
9.15 Objektorientiertes Design (OOD) nach Coad	129
9.16 Themenspezifische Literaturhinweise	129
10 Einführung in die Unified Modeling Language (UML)	131
10.1 Das Analyse- und Designinstrument UML	131
10.2 Themenspezifische Literaturhinweise	136
10.3 Die statische Klassenmodellierung	137
10.4 Assoziation	142
10.5 Aggregation	146
10.6 Generalisierung - Spezialisierung	147
10.7 Implementierungsdiagramme	148
10.8 Themenspezifische Literaturhinweise	150
10.9 Die dynamische Klassenmodellierung	150
10.10 Themenspezifische Literaturhinweise	157
10.11 Die UML und der Systementwurf	157
10.12 Vom Problembereich/Anwendungsfall zur Klassenmodellierung	158
10.13 Programmiersprachen: Java vs. C++?	160
11 Praxisbericht: Das Arbeiten mit problembereichsneutralen Komponenten	163
11.1 Von Assoziationen zu problembereichs-neutralen Komponenten	163
11.2 Bereichsneutrale Komponenten	167
12 Vorgehensmodelle für Software-Entwicklungsprojekte	172
12.1 Traditionelle Vorgehensweisen	172
12.2 Modernere Vorgehensweisen	173
12.3 Einführung in das Vorgehensmodell OEP	175
12.4 Das Siemens-Vorgehensmodell	184
13 Praxisbericht: Vorgehensmodell plus Case-Tool reduziert Projektlaufzeiten	196
14 Praxisbericht: Das objektorientierte CASE-Tool ObjectiF	198
14.1 Einführung	198
14.2 Die Systemarchitektur bewusst gestalten	202
14.3 Ein Projekt mit objectiF®	205

15 Anhang 1: Aktuelle CASE Produkte	221
16 Anhang 2: Abbildungsverzeichnis	222
17 Anhang 3: Index	225