
Inhaltsverzeichnis

1	Was ist Software-Engineering?	1
2	Prozessmodellierung	7
2.1	Unternehmensprozesse	8
2.2	Prozessmodellierung mit Aktivitätsdiagrammen	11
2.3	Risikomanagement	19
2.4	Risikoanalyse Prozessmodellierung	21
2.5	Aufgaben	23
3	Vorgehensmodelle	25
3.1	Phasen der Software-Entwicklung	27
3.2	Wasserfallmodell	28
3.3	Prototypische Entwicklung	30
3.4	Iterative Entwicklung	31
3.5	Iterativ-inkrementelle Entwicklung	32
3.6	Allgemeines V-Modell	33
3.7	Das V-Modell der Bundesrepublik Deutschland	35
3.8	Rational Unified Process	37
3.9	Agile Vorgehensmodelle	39
3.10	Scrum	42
3.11	Extreme Programming	45
3.12	Test Driven und Behaviour Driven Development	47
3.13	Risikoanalyse Vorgehensmodell	49
3.14	Aufgaben	50
4	Anforderungsanalyse	53
4.1	Stakeholder und Ziele	54
4.2	Klärung der Hauptfunktionalität (Use Cases)	61
4.3	Beschreibung typischer und alternativer Abläufe	70
4.4	Ableitung funktionaler Anforderungen	73
4.5	Nicht-funktionale Anforderungen	81

4.6	Lasten- und Pflichtenheft	84
4.7	Varianten und ergänzende Ansätze der Anforderungsanalyse	86
4.8	Risikoanalyse Anforderungsanalyse	88
4.9	Aufgaben	89
5	Grobdesign.	93
5.1	Systemarchitektur	94
5.2	Ableitung von grundlegenden Klassen	96
5.3	Ableitung von Methoden und Kontrollklassen	102
5.4	Validierung mit Sequenzdiagrammen	108
5.5	Überlegungen zur Oberflächenentwicklung.	118
5.6	Anforderungsverfolgung	121
5.7	Risikoanalyse Grobdesign	122
5.8	Aufgaben	124
6	Vom Klassendiagramm zum Programm.	127
6.1	CASE-Werkzeuge	128
6.2	Übersetzung einzelner Klassen	129
6.3	Übersetzung von Assoziationen.	133
6.4	Spezielle Arten der Objektzugehörigkeit.	139
6.5	Aufbau einer Software-Architektur	144
6.6	Weitere Schritte zum lauffähigen Programm.	150
6.7	Fallstudie: Von Anforderungen zum Code	154
6.8	Risikoanalyse Klassendiagrammübersetzung	161
6.9	Aufgaben	163
7	Konkretisierungen im Feindesign	167
7.1	Zustandsdiagramme	168
7.2	Object Constraint Language	178
7.3	Risikoanalyse Feindesign.	184
7.4	Aufgaben	185
8	Optimierung des Designmodells	189
8.1	Design im Kleinen	190
8.2	Model View Controller.	199
8.3	Vorstellung einiger GoF-Pattern	205
8.4	Fallstudie zur patternbasierten Architektur: Redux	234
8.5	Abschlussbemerkungen zu Pattern	243
8.6	Risikoanalyse Design-Optimierungen	246
8.7	Aufgaben	247
9	Implementierungsaspekte	253
9.1	Einfluss nicht-funktionaler Anforderungen	254
9.2	Verteilte Systeme	255

9.3	Grundideen von XML und JSON	260
9.4	Programmbibliotheken	262
9.5	Komponenten	264
9.6	Frameworks	268
9.7	Persistente Datenhaltung	274
9.8	Annotationen	277
9.9	Domain Specific Languages	281
9.10	Model Driven Architecture	283
9.11	Refactoring	285
9.12	Konkretes Beispiel: JSF, JPA, CDI und EJB	288
9.13	Risikoanalyse Implementierungsaspekte	302
9.14	Aufgaben	304
10	Oberflächengestaltung	309
10.1	Hintergründe der Oberflächengestaltung	309
10.2	Konkretisierung des Nutzbarkeitsbegriffs	311
10.3	Berücksichtigung der Ergonomie im Software- Entwicklungsprozess	316
10.4	Prüfung der Nutzbarkeit	318
10.5	Risikoanalyse Oberflächengestaltung	322
10.6	Aufgaben	323
11	Qualitätssicherung	325
11.1	Formale Korrektheit	326
11.2	Zusicherungen	328
11.3	Unit-Tests	331
11.4	Testbarkeit von Systemen herstellen	340
11.5	Äquivalenzklassentests	344
11.6	Kontrollflussbezogene Testverfahren	352
11.7	Testarten und Testumfeld	359
11.8	Metriken	364
11.9	Konstruktive Qualitätssicherung	369
11.10	Manuelle Prüfverfahren	371
11.11	Risikoanalyse Qualitätssicherung	374
11.12	Aufgaben	376
12	Umfeld der Software-Entwicklung	383
12.1	Versionsmanagement	384
12.2	Build-Management	389
12.3	Grundlagen der Projektplanung und -verfolgung	394
12.4	Aufwandsschätzung	404
12.5	Qualitätsmanagement	415
12.6	Der Mensch im Projekt	422

12.7	Risikoanalyse Projektumfeld	428
12.8	Aufgaben	431
13	Ausblick	435
13.1	Cloud, Low- und No-Code.	435
13.2	Künstliche Intelligenz	438
A	UML-Überblick	447
Literatur.		455
Stichwortverzeichnis.		463