Inhaltsverzeichnis

1 Einleitung ... 1
1.1 Sinn und Zweck einer Due Diligence ... 2
1.2 Besonderheiten bei der Analyse von Softwareunternehmen ... 3
1.3 Was dieses Buch leistet ... 5
1.4 Für wen dieses Buch geschrieben ist ... 5
1.4.1 Anredeform ... 6
1.5 Was dieses Buch nicht bietet ... 7
1.6 Wie dieses Buch strukturiert ist ... 7
1.7 Begriffe, verwendete Sprache ... 8
1.7.1 IT/ITK ... 8
Literatur ... 9

2 Wert der Software ... 11
2.1 Einführung ... 11
2.2 Immaterieller Vermögensgegenstand Software ... 13
2.3 Bewertung eines Softwareunternehmens ... 15
2.4 Substanzwertverfahren ... 16
2.4.1 Forschung versus Entwicklung ... 17
2.4.2 Leistungsunterschiede bei Programmierern ... 18
2.4.3 Fazit ... 20
2.5 Multiplikatorverfahren ... 21
2.5.1 Fazit ... 22
2.6 Ertragswertverfahren ... 23
2.7 Herausforderungen beim Ertragswertverfahren ... 26
2.7.1 Übertragbarkeit der Ertragskraft ... 27
2.7.2 Plausibilität der Prognose ... 27
2.7.3 Lebensdauer des Unternehmens ... 30
2.7.4 Schwierige Suche nach dem Betafaktor ... 35
2.8 Handlungsempfehlungen ... 37
2.8.1 Analysebedarf festlegen ... 38
2.8.2 Strategie zur Durchführung einer Software Due Diligence ... 41

2.8.3 Ergebnis einer Software Due Diligence: Wert der Software und -Entwicklung ... 45
2.8.4 Beweis durch Widerspruch ... 47
2.9 Weitere Einflussfaktoren ... 48
2.9.1 Synergie- bzw. Wertsteigerungspotenziale ... 48
Literatur ... 49

3 Ablauf einer Software Due Diligence ... 51
3.1 Informationsquellen ... 52
3.2 Exkurs: Kategorien der Softwareentwicklung ... 54
3.2.1 Kategorie 1: Hersteller von Produktsoftware ... 55
3.2.2 Kategorie 2: Hersteller von Unternehmenssoftware ... 55
3.2.3 Kategorie 3: Hersteller von Individualsoftware ... 55
3.2.4 Kategorie 4: Hersteller von Embedded Software ... 56
3.2.5 Kategorie 5: Sonderfall Hersteller von Software für in-house Anwendungen ... 57
3.2.6 Mischformen ... 57
3.2.7 Komplexität der Softwareentwicklung ... 58
3.2.8 Kommerzielle Chancen ... 59
3.3 Software Due Diligence Schwerpunkte ... 60
3.3.1 Thematische Eingrenzung der weiteren Ausführungen ... 62
3.4 Agenda einer Software Due Diligence ... 62
3.5 Kick-off Meeting ... 63
3.5.1 Ziel ... 63
3.5.2 Teilnehmer ... 64
3.5.3 Vorbereitung ... 64
3.5.4 Organigramm der Softwareentwicklung ... 64
3.5.5 Weitere vorbereitende Schritte ... 66
3.5.6 Durchführung ... 66
3.5.7 Auswertung ... 69
3.5.8 Uneinigkeit im Management ... 70
3.5.9 Neuschreiben der Software ... 70
3.6 Due Diligence Abbruchkriterien ... 72
3.7 Fragebögen und Checklisten ... 73
3.7.1 Ziel ... 74
3.7.2 Durchführung ... 74
3.7.3 Auswertung der Dokumente ... 75
3.8 Einführung in die Software und die Softwareentwicklung ... 75
3.8.1 Ziel ... 75
3.8.2 Durchführung ... 76
3.8.3 Auswertung ... 78
3.9 Iterative Detailanalyse ... 79

3.9.1 Code Review(s) ... 79
3.10 Auswertung der Ergebnisse ... 81
3.10.1 Abschlussbericht ... 81
3.10.2 Abschlussbesprechung ... 82
3.11 Sonderfälle ... 83
3.11.1 Stark limitierte Prüfungszeit ... 83
3.11.2 Kein Zutritt zum Target ... 84
3.11.3 Unerwartet hoher Prüfumfang ... 84
Literatur ... 85

4 Das Team ... 87
4.1 Herausforderungen mit den Mitarbeitern ... 88
4.1.1 Einführung zur Gattung der Softwareentwickler ... 88
4.1.2 Sachverhalt, Daten ... 90
4.1.3 Schlechte Kommunikations-Eigenschaften ... 93
4.1.4 Mangelnde Kritikfähigkeit ... 95
4.1.5 Mangelnde Empathie ... 96
4.1.6 Mangelndes Selbstbewusstsein ... 96
4.1.7 Berufskrankheit Optimismus ... 97
4.1.8 Mangelndes Interesse an Verantwortung ... 98
4.1.9 Technik verliebt ... 99
4.1.10 Mangelnde Konzentrationsfähigkeit und Disziplin ... 100
4.1.11 Lernfaul ... 101
4.1.12 Schlechte körperliche Fitness ... 103
4.1.13 Kreativität Fehlanzeige ... 104
4.1.14 Wenig Verständnis für wirtschaftliche Zusammenhänge ... 104
4.2 Herausforderungen in den Teams ... 105
4.2.1 Einführung zum Thema Entwicklerteams ... 105
4.2.2 Zu viele freie Mitarbeiter ... 105
4.2.3 Mangelhafte teaminterne Kommunikation ... 107
4.2.4 Sprachprobleme in internationalen Teams ... 109
4.2.5 Typische Beispiele für problematische Teams ... 111
4.3 Herausforderungen mit den Führungskräften ... 115
4.3.1 Einführung zum Thema Führung von Softwareentwicklern ... 115
4.3.2 Mangelnde Erfahrung in Softwareentwicklung ... 115
4.3.3 Zu große Toleranz gegenüber schlechter Leistung ... 120
4.4 Fragenkatalog zur Vorbereitung ... 122
Literatur ... 122

5 Entwicklertools ... 125
5.1 Einführung ... 125
5.2 Integrierte Entwicklungsumgebung ... 126
5.3 Programmiersprachen ... 128

5.4 Plattformen ... 131
5.5 Versionsverwaltung und ALMS ... 134
5.6 Weitere Tools ... 137
5.6.1 Tools zur statischen Code-Analyse ... 137
5.6.2 Profiler ... 138
5.7 3rd-Party Bibliotheken, Komponenten und Webservices ... 138
5.8 Hardwareausstattung ... 141
5.9 Herausforderungen in den Unternehmen ... 141
5.9.1 Planlose Tool- und Plattformauswahl ... 142
5.9.2 Strategische Fehlentscheidung bei der Werkzeug-Auswahl ... 145
5.9.3 Mangelnde Standardisierung ... 147
5.9.4 Zu umfangreiche Toolsets ... 148
5.9.5 Toolsets mit Update-Bedarf ... 148
5.9.6 Einsatz von zu vielen Fremdkomponenten ... 151
5.9.7 Abhängigkeit von Komponenten mit ungewisser Zukunft ... 153
5.9.8 Mangelhafte Arbeitsumgebung ... 155
5.9.9 Kein systematisches Technologie-Monitoring ... 156
5.10 Fragenkatalog zur Vorbereitung ... 159
Literatur ... 159

6 Prozesse für die Softwareentwicklung ... 161
6.1 Einführung ... 161
6.2 Projektmanagement ... 164
6.2.1 Vorgehensmodelle ... 165
6.3 Requirements-Engineering ... 167
6.3.1 Erhebung der Anforderungen ... 168
6.3.2 Analysieren und Validieren der Anforderungen ... 169
6.3.3 Anforderungsspezifikation als Ergebnis ... 170
6.3.4 Schwierigkeiten beim Management von Anforderungen ... 171
6.4 Systementwurf und -design ... 173
6.5 Implementierung ... 175
6.5.1 Programmier-Richtlinien ... 177
6.5.2 UI-Style-Guidelines ... 178
6.5.3 Regelungen für Ablageorte ... 179
6.5.4 Reviews ... 180
6.5.5 Logging ... 181
6.6 Build ... 182
6.6.1 Continuous Integration ... 185
6.6.2 Regeln für die Versionierung ... 186
6.7 Software-Test ... 188
6.7.1 Manuelle Tests ... 189
6.7.2 Automatisierte Tests ... 190

6.7.3 Testablauf ... 190
6.7.4 Schwierigkeiten beim Testen ... 194
6.7.5 Der große Nutzen von Testprozessen ... 195
6.8 (Tech) Support ... 198
6.8.1 Kommunikations-Kanäle ... 199
6.8.2 Mitarbeiter im Support ... 200
6.8.3 Bearbeiten von Bug-Meldungen ... 202
6.8.4 Schwierigkeiten im Support ... 203
6.9 Wartung/Softwarepflege ... 207
6.10 Herausforderungen in den Unternehmen ... 209
6.10.1 Mangelndes Prozessdenken ... 209
6.10.2 Zu wenig Markt-Know-how ... 212
6.10.3 Häufig sich ändernde Anforderungen ... 214
6.10.4 Keine Konsolidierungsphasen ... 215
6.10.5 Softwareentwicklung an mehreren Standorten ... 218
6.10.6 Zu großzügige Homeoffice-Regelungen ... 220
6.10.7 Zu hohe Arbeitsbelastung der Mitarbeiter ... 222
6.10.8 Ungeeignete Büros ... 223
6.10.9 Zu wenig Tests ... 224
6.10.10 Unzureichend qualifizierte Mitarbeiter im Support ... 226
6.10.11 Mehrfachverantwortlichkeiten ... 228
6.10.12 Keine Strategie zur Vermeidung von Support ... 230
6.10.13 Mangelnder Fokus auf Evolvierbarkeit ... 231
6.10.14 Falsche Einstellung zum Thema Dokumentation ... 232
6.10.15 Keine Notfallpläne ... 237
6.10.16 Zu wenig standardisierte Implementierung ... 239
6.10.17 Unsichere Softwarelizensierung ... 240
6.11 Fragenkatalog zur Vorbereitung ... 241
Literatur ... 242

7 Der Quellcode ... 245
7.1 Einführung ... 245
7.2 Durchführung eines Code Reviews ... 246
7.3 Merkmale guten Codes ... 250
7.3.1 Einhaltung von Programmier-Standards und -Richtlinien ... 251
7.3.2 Aussagekräftige Bezeichner ... 252
7.3.3 Nachvollziehbare Speicherorte ... 252
7.3.4 Verwendung von Entwurfsmustern ... 252
7.3.5 Versionierung von Beginn an ... 253
7.3.6 Keine Magic Numbers ... 253
7.3.7 Flexible Systemkonfiguration ... 254
7.3.8 Vollständige Fehlerbehandlung ... 254

7.3.9 Separate String-Verwaltung ... 255
7.3.10 Konsequente Nutzung der Plattform-Funktionalität ... 255
7.3.11 Kapselung der 3rd-Party-Komponenten ... 255
7.3.12 Exkurs Code-Metriken ... 255
7.3.13 Durchdachte Architektur ... 259
7.3.14 Exkurs Software-Architektur ... 259
7.4 Herausforderungen bei der Durchführung von Code Reviews ... 263
7.5 Ergebnisse von Code Reviews ... 266
7.6 Herausforderungen in den Unternehmen ... 268
7.6.1 Keine Architektur ... 268
7.6.2 Architektur nach Conway ... 272
7.6.3 Kontrollverlust ... 272
7.6.4 Fehlende Abstraktionen ... 275
7.6.5 Unzureichend kommentierter Code ... 276
7.6.6 Überforderung auf Seiten der Programmierer ... 277
7.6.7 Mix aus altem und neuem Code ... 277
7.6.8 Ignorieren der Programmier-Richtlinien ... 278
7.7 Fragenkatalog zur Vorbereitung ... 278
Literatur ... 279

8 Die Anwendung ... 281
8.1 Einführung ... 281
8.2 Annäherung an eine Software ... 283
8.3 User Experience als Erfolgsfaktor für Softwareprodukte ... 284
8.3.1 UX Feature #1: Dauer der Startprozedur? ... 284
8.3.2 UX Feature #2: Freie Skalierung der Anzeige im UI? ... 286
8.3.3 UX Feature #3: Ease of use? ... 286
8.3.4 UX Feature #4: mehrstufiges Rückgängig/Wiederherstellen? ... 287
8.3.5 UX Feature #5: Reaktiv durch Parallelisierung? ... 287
8.3.6 UX Feature #6: Erweiterbarkeit? ... 288
8.3.7 UX Feature #7: Robustheit/Resilienz? ... 289
8.3.8 UX Feature #8: 64-Bit? ... 289
8.3.9 UX Feature #9: Lokalisierung des UI? ... 290
8.3.10 UX Feature #10: Fehlerbehandlung? ... 290
8.4 Typische Schwachstellen in Anwendungen ... 291
8.4.1 User Interface folgt keinem Standard ... 291
8.4.2 Inkonsistentes User Interface ... 293
8.4.3 Funktional überladene Software ... 294
8.4.4 Zu schwierige Installation der Software ... 294
8.5 Fragenkatalog zur Vorbereitung ... 295
Literatur ... 295

9 Ausblick 297
9.1 Herausforderung Software-Industrie 298
9.2 Empfehlung: Fokussieren 299
9.3 Empfehlung: Komplexität reduzieren 300
9.4 Empfehlung: Know-how schützen 302
9.5 Empfehlung: Regelmäßiger Check-up 302
9.6 Digitalisierung als Chance für den Mittelstand 302

Stichwortverzeichnis 305