

---

# Software Due Diligence

---

Christian Demant

# Software Due Diligence

Softwareentwicklung als Asset bewertet

Dipl.-Ing. Christian Demant  
Stuttgart  
Deutschland

ISBN 978-3-662-53061-0      ISBN 978-3-662-53062-7 (eBook)  
<https://doi.org/10.1007/978-3-662-53062-7>

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.d-nb.de> abrufbar.

Springer Gabler

© Springer-Verlag GmbH Deutschland 2018

Das Werk einschließlich aller seiner Teile ist urheberrechtlich geschützt. Jede Verwertung, die nicht ausdrücklich vom Urheberrechtsgesetz zugelassen ist, bedarf der vorherigen Zustimmung des Verlags. Das gilt insbesondere für Vervielfältigungen, Bearbeitungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften. Der Verlag, die Autoren und die Herausgeber gehen davon aus, dass die Angaben und Informationen in diesem Werk zum Zeitpunkt der Veröffentlichung vollständig und korrekt sind. Weder der Verlag, noch die Autoren oder die Herausgeber übernehmen, ausdrücklich oder implizit, Gewähr für den Inhalt des Werkes, etwaige Fehler oder Äußerungen. Der Verlag bleibt im Hinblick auf geografische Zuordnungen und Gebietsbezeichnungen in veröffentlichten Karten und Institutionsadressen neutral.

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer Gabler ist Teil von Springer Nature

Die eingetragene Gesellschaft ist Springer-Verlag GmbH Deutschland

Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

---

## Vorwort

„Pi mal Daumen“ lautete die Antwort eines renommierten Experten für „Mergers & Acquisitions“ auf meine Frage, wie beim Verkauf von Industriebetrieben die im Unternehmen vorhandene Software-Entwicklungskapazität bewertet wird. Diese Aussage hat mich entsetzt. Wie kann man nur bei der Bewertung eines Unternehmens die Softwareentwicklung ausklammern? Für viele Unternehmen in den unterschiedlichsten Branchen erfüllt die produzierte Software schon heute oftmals (über-)lebensnotwendige Zwecke. Wie kann ein Fachmann darüber hinwegsehen? Man stelle sich vor, der mit der Bewertung eines Gebrauchtwagens beauftragte Gutachter prüfte den Zustand von Motor und Getriebe ebenfalls nur Pi mal Daumen. Oder der mit einem routinemäßigen Check-up beauftragte Kardiologe gäbe sich bei der Beurteilung der Pumpleistung des Herzens mit einem Pi mal Daumen zufrieden?

Was mögen die Gründe für diese Gleichgültigkeit gegenüber der Softwareentwicklung sein? Hält man die aus den Jahresabschlüssen abgeleiteten Kennzahlen für ausreichend? Oder ist es schlichtweg Unwissenheit oder Arroganz? Wahrscheinlich ist es eine Mischung aus vielen Faktoren. Auf jeden Fall ist diese Ignoranz im Rahmen einer Unternehmensanalyse aus meiner Sicht in höchstem Maße fahrlässig. Es ist öffentlich bekannt und im Rahmen zahlreicher Studien dokumentiert, dass viele Softwareprojekte regelmäßig scheitern. Zusätzlich warnen Brancheninsider seit langem davor, dass aufgrund des Fachkräftemangels viele Unternehmen in ihren Entwicklungsabteilungen einen Berg an veraltetem und kaum noch wartbarem Code vor sich herschieben. Die zukünftigen Resultate der Softwareentwicklung sind bei diesem morschen Fundament in mehrfacher Hinsicht ungewiss. Durch Pi mal Daumen sind daher für einen angehenden Inhaber oder die zukünftigen Verantwortlichen Enttäuschungen vorprogrammiert. Wer die sorgfältige Analyse eines zentralen Unternehmenswertes unterlässt, muss sich im Klaren darüber sein, dass das überaus riskant ist.

Je mehr ich über die Angelegenheit nachdachte, desto klarer sah ich die hier benannte Achtlosigkeit bei der Durchführung einer Unternehmens-Analyse als Ausprägung einer Problematik, die sich seit Jahren durch die ganze Industrie zieht. Es sind nicht nur Wirtschaftsprüfer, die bei ihrem Blick von außen in die Unternehmen hinein Schwierigkeiten mit dem Thema Softwareentwicklung haben. Selbst innerhalb der Unternehmen gibt es

zwischen dem Management und den Softwareentwicklern häufig umfangreiche Kommunikationsprobleme und ein erhebliches Misstrauen. In den nun über fünfundzwanzig Jahren meiner beruflichen Tätigkeit im Bereich der professionellen Softwareentwicklung wurde ich immer wieder überrascht, wie mangels Verständnis für die Herausforderungen und sehr speziellen Problemstellungen der Softwareentwicklung in vielen Unternehmen Führung und Softwareentwickler komplett aneinander vorbeireden.

Ich hatte schon mit Unternehmen zu tun, in denen die Mitarbeiter in der Softwareentwicklung machten, wozu sie Lust hatten, weil letztlich niemand ihre Arbeit angemessen überprüfen konnte. Manchmal entwickeln sich in den Unternehmen sogar wahre Parallelwelten. Software wird dann eher gebastelt und weniger professionell entwickelt. Führung und Inhaber spüren zwar mit einem gewissen Unwohlsein im Bauch, dass die Softwareentwicklung aus dem Ruder gelaufen ist, trauen sich aber nicht an das Problem heran. Dies führt regelmäßig zu verhängnisvollen Fehleinschätzungen und riskanten unternehmerischen Entscheidungen.

Um mögliche Gründe für diese brisante Lage aufzuzeigen, möchte ich einen Blick auf die tragende Säule unserer Wirtschaft werfen, den auch international so vielgelobten Mittelstand. In Unternehmen des Mittelstandes wissen Inhaber und Führung im Normalfall recht gut Bescheid, welchen Wert das Unternehmen und seine Geschäftsbereiche haben und wie das Unternehmen im Vergleich zum Wettbewerber am Markt positioniert ist. Der Grund dafür ist einfach und stellt eine typische Stärke des deutschen Mittelstandes dar: Inhaber und Führung sind meistens vom Fach. Eine Vielzahl von Unternehmen in der Metallverarbeitung haben als Geschäftsführer z. B. einen studierten Maschinenbauer oder Feinwerktechniker. Bei Bauträgern ist oftmals ein Architekt oder ein Bauingenieur am Ruder. Auch wenn sich so mancher Mittelständler im Zuge der internationalen Expansion einen Betriebswirt oder MBA mit internationaler Erfahrung als Geschäftsführer an Bord geholt hat, so ist in der Regel im Management-Board bzw. in der zweiten Führungsebene eine hohe Fachkompetenz und Expertise hinsichtlich des traditionellen Kerngeschäfts vorhanden.

Doch die Situation hat sich in den letzten Jahren verändert.

Softwareentwicklung war in vielen Unternehmen bis vor zehn Jahren eher selten und bis vor 20 Jahren so gut wie gar kein Thema. Als Microsoft im Jahr 1990 die Version 3.0 des PC-Betriebssystems Windows einführte, wurden weltweit jährlich gerade einmal 20 Millionen PCs verkauft. Durch die zunehmende Digitalisierung und den Einzug von Software in die Unternehmen haben sich jedoch die Gewichte verschoben. Software durchdringt alle Bereiche und ist allgegenwärtig. Das bekannte englische Zitat „Software is everywhere“ liefert derzeit bereits über fünf Millionen Treffer bei Google. Aktuell sind Industrie 4.0 und das (Industrial) Internet of Things zentrale Themen. Im Zuge dieser „digitalen Transformation“ müssen sich immer mehr Betriebe mit Fragen der Softwareentwicklung auseinandersetzen.

Blicken wir exemplarisch auf den Maschinenbau: Nach den festverdrahteten Relais Steuerungen folgte ab 1968 mit Vorstellung des „Modular Digital Controllers“ (Modicon 084) seitens General Motors der Siegeszug der Speicherprogrammierbaren Steuerungen

(SPS) in den 80er Jahren. Wer als Hersteller auf den SPS-Zug damals nicht aufsprang, war spätestens Mitte der 90er Jahre vom Markt verschwunden. Heute wird erwartet, dass selbst einfache Geräte und Maschinen die Prozessdaten in der Cloud abspeichern, statistisch auswerten und der Maschinenbediener die generierten Daten und Key Performance Indicators im Vorbeilaufen auf einem Tablet-Computer visualisieren kann. Wenn die Daten schon in der Cloud liegen, liefert der Maschinen-Lieferant idealerweise über intelligente „Machine Learning“ Ansätze gleich noch Vorschläge zur Reduzierung der Stillstands-Zeiten mit. Um all diese Funktionalität bieten zu können, muss eine Menge an Software entwickelt werden.

Aber auch im Consumer-Bereich werden immer mehr „smarte“ Geräte verkauft, deren Betriebsgrundlage die Software darstellt. Wer vom Smartphone aus per App die Heizung reguliert oder sich ein Bild von der über dem Hauseingang montierten Webcam anzeigen lässt, nutzt die Arbeitsleistung cleverer Softwareentwickler. Näher betrachtet stellt Software in vielen Unternehmen bereits heute das zentrale Know-how („Intellectual Property“) dar, auch wenn vielen Inhabern das noch nicht so klar ist. Der US-Elektroauto-bauer Tesla hat die große Bedeutung des Software-Know-hows erkannt und versteht sich daher heute schon primär als Software- und nicht als Automobilhersteller. Für deutsche Automobil-Vorstände scheint diese Sichtweise noch auf Jahre undenkbar. Warum halten gerade in Deutschland so viele Führungskräfte und Inhaber an der traditionellen Sichtweise auf die Unternehmen fest?

Das Management der meisten Betriebe in Deutschland wurde vor der umfassenden Anwendung der Computer-Technik ausgebildet und hat daher, unabhängig vom Studiengang, zwangsläufig Schwierigkeiten mit der Beurteilung der Software-Thematik. Viele Inhaber, Geschäftsführer und Führungskräfte sind Anfang Mitte 50 oder älter und zählen wie ein Großteil unseres politischen Führungspersonals nicht zu den „Digital Natives“. Insbesondere auch in dem für die deutsche Wirtschaft so wichtigen Maschinenbau haben viele Führungskräfte einen anderen beruflichen Hintergrund und es fehlt aufgrund des fortgeschrittenen Alters der fachliche Zugang zum Thema Software. Ältere Chefs neigen auch dazu weniger zu investieren und die Substanz der Betriebe zu verbrauchen. Als Folge unterbleiben turnusmäßig erforderliche Konsolidierungen und Aktualisierungen des Codes und der Software-Architektur und die im Unternehmen bereits vorhandene Software „verrottet“ mehr und mehr.

Neben den durch demografische Effekte forcierten Problemen innerhalb der Unternehmen spielen auch gesellschaftliche Einflussfaktoren eine wichtige Rolle. In Deutschland existiert auch heute noch in breiten gesellschaftlichen Schichten eine latente Technik-Feindlichkeit. Ärzte, Richter und Hochschulprofessoren genießen hohes gesellschaftliches Ansehen, Softwareentwickler und IT-Experten hingegen kommen bestenfalls auf mittlere Ränge und werden nicht selten für ihre Technikaffinität belächelt. Statt die wirtschaftlichen Chancen neuer Technologien ergebnisoffen und sachlich auszuloten, dominieren in der öffentlichen Diskussion häufig die Bedenken und Ängste. Die bestenfalls durchschnittliche Reputation der IT-Berufe führt dazu, dass viele Schulabgänger Abstand zu dem Thema halten. So studieren leider immer noch mehr junge Leute Rechtswissenschaften

als Informatik. Offensichtlich ist bei den Abiturienten noch nicht angekommen, dass ein Industrieland an der Schwelle zur Komplett-Digitalisierung auf Jahrzehnte hin einen großen Bedarf an Informatikern und Ingenieuren mit Software-Know-how haben wird und die beruflichen Aussichten für Absolventen aus diesen Bereichen auch langfristig entsprechend gut sein müssten.

Diejenigen, die dennoch den Schritt in die IT-Welt wagen und in ihrer täglichen Arbeit dann mit der Bearbeitung komplexer Fragestellungen konfrontiert werden, fühlen sich oft nicht ausreichend wertgeschätzt. Außerdem sind die üblichen, stark an der Unternehmens-Zugehörigkeit ausgerichteten Vergütungssysteme für junge Spezialisten selten ausreichend motivierend. Top-Entwickler und Spitzen-Programmierer haben daher eigentlich nur über die Selbständigkeit die Chance, dass ihre Leistungen und Fähigkeiten angemessen entlohnt werden.

Aber auch auf Seiten der betroffenen Softwareentwickler wird oft unglücklich agiert: Den Programmierern ist Führung und Management häufig suspekt und notwendige wirtschaftliche Erwägungen und Fragestellungen – beispielsweise in Bezug auf Marketing und Vertrieb – bekommen nicht die Beachtung, die sie aus Sicht verantwortungsvoller Mitarbeiter haben sollten. Zusätzlich fehlt vielen Technikern und Ingenieuren die Empathie. So genannten „Nerds“ unter den Softwareentwicklern wird nachgesagt, dass sie kein Gespür dafür haben, wie andere Stakeholder die Software sehen und dass sie sich schwer tun technische Zusammenhänge verständlich zu erklären.

Die unmittelbaren Auswirkungen der zu geringen gesellschaftlichen Wertschätzung von Software-Themen und das Ergrauen der Führungsriege in den Unternehmen führen insgesamt zu einer zunehmend schwierigen Konstellation. Ob insbesondere der mittelständisch strukturierte Maschinenbau vor diesem Hintergrund den nun anstehenden Wandel zu einer umfassenden Digitalisierung rechtzeitig erfolgreich schaffen kann, sehe ich mit Skepsis. Zu groß ist mittlerweile der Abstand zu den führenden US-amerikanischen IT-Unternehmen, die (Cloud-)Infrastruktur, Entwicklungs-Werkzeuge und zunehmend auch die kreativen Anwendungen und Geschäftsmodelle vorgeben. Während hierzulande noch mühsam versucht wird, Hunderte von Kommunen und Gewerbegebieten mit leistungsfähigen Breitband-Anschlüssen auszustatten, wird in den USA bereits diskutiert, wie sich mithilfe von Drohnen und neuartigen Robotern zukünftig auch die Computer-Hardware automatisch updaten kann. Regelmäßige Software-Updates per Internet-Anbindung werden bereits als Stand der Technik vorausgesetzt.

Um im Zeitalter von Mobile- und Cloud-Computing nicht komplett den Anschluss zu verlieren und mögliche gewinnbringende Anwendungen und Geschäftsmodelle rechtzeitig erkennen und implementieren zu können, benötigen viele Unternehmen schnell mehr professionelles Software-Know-how und bessere Entwicklungs-Prozesse. Außerdem müssen Management und Entwickler enger zusammenrücken und in gegenseitiger Wertschätzung intensiver miteinander kommunizieren.

In einem ersten Schritt ist hierzu eine kritische Bestandsaufnahme erforderlich, um die bestehenden Defizite bei der Softwareentwicklung zu erkennen und daraus geeignete operative und strategische Maßnahmen abzuleiten. Selbsterkenntnis ist bekanntlich der erste

Weg zur Besserung. Ich halte es für eine elementare Notwendigkeit, dass Inhaber und Management ein Gefühl dafür bekommen, welchen Wert die im Unternehmen angesiedelte Softwareentwicklung aktuell hat. „Wert“ betrachte ich auch im übertragenen Sinn: Hat es überhaupt einen Wert, macht es Sinn, ist es zukunftsfähig, so wie aktuell im Unternehmen Software entwickelt wird? Was sind die Stärken, wo gibt es Schwächen und in welchen Bereichen lauern Risiken für das Unternehmen? Mit den Ergebnissen einer resoluten Bewertung der Softwareentwicklung im Unternehmen lassen sich dann (Investitions-)Entscheidungen ableiten und entsprechende Mittel und Ressourcen zur Verfügung stellen.

In meiner Rolle als Unternehmer und Geschäftsführer sind mir die Interessen und die Argumentation des Managements in Bezug auf Softwareprojekte sehr wohl bekannt. Aber auch die andere Seite ist mir bestens vertraut: Fast zwei Jahrzehnte als Leiter eines Teams von Softwareentwicklern haben mir die besondere Denkweise und Motivation der Programmierer in den unterschiedlichsten Arbeitssituationen nahegebracht.

Mit dem vorliegenden Buch möchte ich einen Beitrag leisten, die fachliche und teils auch psychologische Distanz zwischen Management und Softwareentwicklern zu verringern und die Zusammenarbeit beider Gruppen durch einen Zugewinn an Transparenz zu verbessern. Das Buch „Software Due Diligence“ soll Außenstehenden ohne tiefere IT-Kenntnisse einen Einblick in die Besonderheiten und vermeintlichen Geheimnisse der Softwareentwicklung geben, immer unter der Prämisse, dabei zu einer Bewertung der betrieblichen Gegebenheiten zu kommen. Wer nicht weiß, wo er aktuell steht, kann kaum den Weg zum Ziel finden.

Stuttgart, im Frühjahr 2017

Christian Demant  
Dipl.-Ing. Technische Kybernetik  
[www.software-due-diligence.de](http://www.software-due-diligence.de)



---

## Danksagung

Für die konstruktive und angenehme Zusammenarbeit bedanke ich mich bei Frau Hestermann-Beyerle und Frau Kollmar-Thoni beim Springer-Verlag.

Für die langjährige vertrauensvolle und überaus erfolgreiche Zusammenarbeit möchte ich mich bei meinen beiden englischen Geschäftspartnern Earl Yardley und Andrew Waller bedanken.

Ein besonderer Dank gilt meinem Personal Trainer Robin Müller-Schober, der mit Sachverstand und Humor dafür sorgt, dass ich physisch fit bleibe und damit den Anforderungen meines Berufsalltags gewachsen bin.

Für langjährige, wertvolle Freundschaften und viele anregende Gespräche bedanke ich mich bei Beate Bauer, Michael Braun, Marcellus Buchheit, Frank Hüfner, Stefan Kast, Dietmar Lude, Christoph Sturm, Oliver Vietze und Thomas Walter.

Besonders hervorheben möchte ich meinen in Seattle lebenden langjährigen Freund Marcellus Buchheit, der mir im Rahmen eines gemeinsamen Abendessens bei einer ausgedehnten Diskussion des Themas „Software Due Diligence“ den entscheidenden Anstoß gab, meine gesamten Erfahrungen, Gedanken und Ideen dazu am besten in einem weiteren Buch zu dokumentieren.

Herzlich bedanken möchte ich mich bei meiner Ehefrau Simone für ihre umfassende Unterstützung bei meinem beruflichen Engagement und ihr wertvolles Feedback beim Schreiben dieses Buches.

---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b> .....	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
<b>2</b>	<b>Wert der Software</b> .....	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
<b>3</b>	<b>Ablauf einer Software Due Diligence .....</b>	<b>51</b>
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
<b>4</b>	<b>Das Team</b>	<b>87</b>
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
<b>5</b>	<b>Entwicklertools</b>	<b>125</b>
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
<b>6</b>	<b>Prozesse für die Softwareentwicklung . . . . .</b>	<b>161</b>
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 Softwarelizenzierung	240
6.11	Fragenkatalog zur Vorbereitung	241
	Literatur	242
<b>7</b>	<b>Der Quellcode</b>	<b>245</b>
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
<b>8</b>	<b>Die Anwendung . . . . .</b>	<b>281</b>
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

---

<b>9</b>	<b>Ausblick</b> .....	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
	<b>Stichwortverzeichnis</b> .....	305